KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Antanas Petrauskas, Karolis Pilypaitis

XML duomenų srautų valdymas verslo komunikacinėse kilpose

Magistro baigiamasis darbas

Vadovas
doc. dr. B. Paradauskas

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Antanas Petrauskas, Karolis Pilypaitis

XML duomenų srautų valdymas verslo komunikacinėse kilpose

Magistro baigiamasis darbas

Kalbos konsultantė		Vadovas	
	I. Mickienė		doc. dr. B. Paradauskas
2006-05			
Recenzentas		Atliko	
			IFM – 0/1 gr. stud. Antanas Petrauskas Karolis Pilypaitis
2006-05		2006-05	

Kaunas, 2006

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

SUMMARY

In changing world of e-commerce and internet based communication, business transactions and data flows play a key role in successful business management. It is vital for companies' information systems to implement business transactions. The purpose of this work is to propose a model for outside business process' implementation. Basic models' steps are: business transaction specification, data structure and architectural solution selection. Communicative action loop is proposed for modeling business processes and XML is suggested as the best method for data exchange in an internet based environment. Deploying three-tier architecture completes the models' steps for successful business information system implementation. In this work we propose a multipurpose tool – DAO (Data Abstraction Object), based on PHP to control, manage and save XML document data. A PHP alternative way – Java Servlet technology is also discussed for using Java tools for XML processing and database connection. At the end of the work an example information system is implemented following the steps described above.

ĮVADAS

Verslas tobulėja ir keičiasi bemaž kiekvieną dieną. Verslo informacinės sistemos privalo greitai ir lanksčiai prisitaikyti prie šių kintančių pokyčių. Interneto ir e-komercijos veržimasis dar labiau sustiprino šią tendenciją. Kompanijos vis dažniau bendrauja tarpusavyje per internetą. Projektuojant ir atnaujinant informacines sistemas svarbu tinkamai valdyti duomenis ir jų srautus, užtikrinant jų pilnumą ir integralumą. Šiame darbe apžvelgiami XML duomenų valdymo metodai verslo transakcijose, pasiūlomas įrankis užtikrinantis saugų ir teisingą duomenų apsikeitimą.

Elektoriniame versle ypatingai svarbus tikslus verslo procesų apibrėžimas. Būtina juos tiksliai specifikuoti ir formaliai apibrėžti. Kintant funkciniams reikalavimams kinta ir duomenų struktūros, todėl lygiai taip pat svarbūs duomenų formato apibrėžimai. Verslo procesai neatsiejami nuo apsikeitimo duomenimis, todėl itin reikšmingas informacinėse sistemose yra verslo transakcijų palaikymas. Duomenys saugomi duomenų bazėse, todėl atsiranda suprantamas poreikis susieti verslo transakciją su duomenų bazės transakcija. Tačiau verslo transakcijoms vienareikšmiškai taikyti duomenų bazių transakcijų savybes ne visada įmanoma. Tai susijusią su pastarųjų žymiai trumpesnėmis trukmėmis, o taip pat su verslo transakcijų atstatymu – verslo procesai dažnai turi išorinius efektus dėl kurių atstatymas gali būti nepageidaujamas arba iš viso neįmanomas. Darbe apžvelgiami įvairūs verslo transakcijų specifikavimo modeliai, o plačiau nagrinėjamas išplėstinių komunikacinių kilpų metodas. Šio metodo pagrindas – komunikacinės binarinės kilpos, tarpusavyje susietos pragmatinėmis ir semantinėmis priklausomybėmis. Kiekviena tokia kilpa susideda iš dviejų aktorių, kurie turi tikslus, iš dviejų aktorių vykdomų procesų, bei duomenų srautų. Tokia kilpa specifikuoja binarinę verslo transakciją.

Bet koks apsikeitimas duomenimis reikalauja duomenų apibrėžties protokolo. Pastaruoju metu itin populiarus yra XML standartas, nes tai yra plačiai naudojama standartizuota kalba, nepriklausoma nuo informacinių sistemų architektūrinių sprendimų. XML yra universalus formatas – galima saugoti įvairių tipų ir formų duomenis. Yra galimybė XML priemonėmis atlikti loginius veiksmus ir t.t.

Šiame darbe nagrinėjamas verslo procesų modeliavimas, kuomet yra siekiama automatizuoti veiklos proceso valdymą; išskiriant darbų sekų procesų modeliavimą kaip vieną iš pagrindinių IS kūrimo etapų. Didžiausias dėmesys yra kreipiamas, į išorinių verslo

procesų modeliavimą t.y. į B2B (*angl. Business – To- Business*) transakcijas, kuomet verslo partneriai keičiasi duomenimis interneto erdvėje. Jų verslo procesai specifikuojami išplėstinėmis komunikacinėmis kilpomis, o architektūros realizacijai parenkamas trijų lygių sprendimas (angl. *three tier architecture*) (duomenų bazės lygis – serverio apdorojančio užklausas lygis – atvaizdavimo lygis), duomenų apsikeitimui naudojant XML formatą. Darbe pasiūlomas duomenų valdymo įrankis DAO, sukurtas PHP (*angl. Hypertext Preprocessor*) kalbos pagrindu. Kaip alternatyvą PHP siūlome ir Java Servlet technologiją – tai Sun Microsystems sukurta technologija dinaminių puslapių generavimui. Java Servlet programuojamas Java programavimo kalba. Panašiai kaip ir PHP tai yra serveryje esantis komponentas, susietas su užklausos adresu, generuojantis atsakymus. Naudodami trijų lygių architektūrą, galime užtikrinti apsikeitimą duomenimis bei įvairias operacijas su jais. Viduriniame lygyje apdorodami duomenis, galime juos paruošti saugojimui į duomenų bazę ar siuntimui.

Darbe atliekamas taikymo pavyzdys pasirinkus dalykinę sritį: e-parduotuvę ir joje esančių prekių pirkimą išsimokėtinai sudarant sutartį su lizingo kompanija. E –parduotuvė – tai paprastos prekių parduotuvės analogas internetinėje erdvėje. Galima pasirinkti norimas prekes, jų kiekius ir pan.. Už prekes yra atsiskaitoma įvairiais būdais .Standartiniu tokioms parduotuvėms būdu – kreditinėmis kortelėmis, banko pavedimu ir pan. Šiame taikymo pavyzdyje plėtojamas būdas atsiskaityti per lizingo kompaniją, sudarant su ja išperkamosios nuomos sutartį. Tai kol kas analogų neturintis atsiskaitymo būdas Lietuvoje. Taikymo pavyzdyje didžiausias dėmesys kreipiamas į išplėstinėmis komunikacinėmis kilpomis specifikuotų verslo transakcijų realizacija trijų lygių architektūroje, duomenims apibrėžti naudojant XML standartą.

Atsiradus benradarbiavimo poreikiui tarp verslo partnerių, yra apibrėžiami nauji išoriniai verslo procesai. Šių procesų kompiuterizavimas verslo partnerių informacinėms sistemoms (IS) iškelia naujus funkcinius reikalavimus. Todėl šio darbo tikslas: verslo partnerių išorinių verslo procesų kompiuterizavimo modelio sudarymas.

Buvo išskirtos tokios šio modelio dalys:

- Verslo procesų specifikavimas komunikacinėmis kilpomis
- Duomenų srautų apibrėžimas XML formatu
- E –verslo procesų projektavimas trijų lygių architektūroje
- Programinė IS procesų realizacija

Siekiant įgyvendinti tikslą, kiekvienam autoriui buvo iškelti šie tikslai:

- Atlikus alternatyvių verslo procesų specifikavimo metodų analizę, pagrįsti komunikacinių kilpų metodo tinkamumą e – verslo procesams aprašyti (Karolis Pilypaitis)
- Pagrįsti XML standarto tinkamumą duomenų mainams internetinėje erdvėje (Antanas Petrauskas)
- Sukurti duomenų abstrakcijos objektą (DAO) programiniam užklausų generavimui ir vykdymui į įvairias duomenų bazes SQL kalba; įsitikinti šio objekto naudingumu, pranašumu, lyginant su alternatyviais objektais (Antanas Petrauskas)
- Įsitikinti sukurto objekto pritaikymu Java programavimo aplinkoje (Karolis Pilypaitis)

Parinkti taikymo pavyzdį realiai egzistuojančių verslo partnerių IS. PHP aplinkoje realizuoti internetinės parduotuvės prekių katalogo atnaujinimą iš XML duomenų, sudaryti galimybę pirkėjui internetu įsigyti prekes per lizingo kompaniją, sudarant su ja išperkamosios nuomos sutartį. Prieš sudarydama tokią sutartį lizingo kompanija atlieka debitoriaus finansinės būklės nustatymą (Antanas Petrauskas, Karolis Pilypaitis)

1 VERSLO TRANSAKCIJŲ MODELIAVIMO IR DUOMENŲ VALDYMO METODŲ ANALIZĖ

1.1 Verslo transakcijų modeliavimas

Šiuolaikinis verslas vis labiau tampa priklausomas nuo informacinių technologijų. Vis svarbesni tampa informacijos mainai tarp verslo partnerių. Norint užtikrinti saugų ir patikimą duomenų perdavimą ir priėmimą, korektišką verslo informacinių sistemų veikimą, būtinas verslo transakcijų palaikymas. Tai ypatingai svarbu informacinės sistemose, kurios yra susijusios su organizacijų darbų sekų valdymu bei verslas - klientui B2C (*angl. Business-To-Customer*) ar verslas - verslui B2B (*angl. Business-To-Business*). Būtent į pastarąsias kreipiamas didžiausias dėmesys šiame darbe.

Nors nėra sutarta dėl formalaus verslo transakcijos apibrėžimo, tačiau paprastai suprantama, kad tai veiksmų tarp verslo partnerių aprašymas. Šiais veiksmais partneriai siekia realizuoti savus tikslus. Paskirstyto verslo proceso realizacija suprantama kaip tokių transakcijų visuma.

Kadangi elektroninio verslo procesas susideda iš veiksmų, kurie naudoja duomenis iš daugialypių duomenų bazių, jis turi būti siejamas su transakcijų semantika. Verslo procesams kaip ir transakcijoms gali būti taikomos ACID (*angl. Atomicity, Consistency, Isolation, Durability*), t.y. nedalumo, vientisumo, izoliavimo, išliekamumo savybės. Šios savybės gali būti taikomos ir proceso dalims. Tačiau visą verslo procesą traktuoti kaip ACID transakciją nėra tikslinga. Visų pirma, verslo procesams būdingos ilgos trukmės, todėl procesą traktuojant kaip transakciją, būtų pareikalauta daug laike įšaldytų išteklių. Antra, verslo procesai dažniausiai būna susieti su daugialypėmis ar paskirstytomis duomenų bazėmis, todėl didelių sąnaudų pareikalautų šių bazių koordinavimas. Trečia, verslo procesai turi išorinius efektus dėl kurių būsenų atstatymo (*angl. rollback*) mechanizmai gali tapti nepageidaujami, arba net visiškai semantiškai neįvykdomi. Todėl įvairūs šaltiniai pateikia įvairius išplėstinius transakcijų modelius. Pavyzdžiui Saga modelyje [1] 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. Veiklos transakcijų modelyje ATM (*angl. Activity transaction model*) transakcijos gali būti skaidomos į subtransakcijas, o šios jungiamos į grandines (*angl. chained*), arba į rinkinius (*angl. nested*). Rinkinio transakcijos vykdomos lygiagrečiai, o klaidų ir išimčių apdorojimui yra taikoma hierarchinė kontrolė. Pirmasis rinkinių modelis [2] apibrėždavo tik uždarąsias

subtransakcijas, tačiau vėliau buvo išplėstas [3] atvirtosioms subtransakcijoms palaikyti. Uždarosios subtransakcijos rezultatus patvirtina tėvinei, o tokie daliniai rezultatai visiškai patvirtinami tik tada kai yra patvirtinama viršutinė (šakninė) transakcija. Taip yra užtikrinami nedalumo ir izoliavimo principai visai transakcijai. Tuo tarpu atvirosios subtransakcijos nepaiso izoliavimo principo ir jų rezultatai patvirtinami iš karto. Klaidų apdorojimas yra hierarchinis. Jei subtransakcija nepavyksta, siunčiamas pranešimas tėvinei transakcijai, o ši jau sprendžia ar vykdyti klaidos apdorojimą ir pabandyti kartoti vaikinę subtransakciją , vykdyti kompensuojamą transakciją ar perduoti klaidos valdymą aukštesnei transakcijai. Klaidos valdymo perdavimas reikalauja kai kurių jau patvirtintų transakcijų kompensavimo. Kai kurios užduotys yra gyvybiškai svarbios, taigi klaida jose reiškia, jog visos žemesnės subtransakcijos yra klaidingos. Šaltinyje [4] pateikiami įvairūs ATM modeliai ir transakcijų semantikos formalizmai. „Pažymėtina, kad išplėstiniai transakcijų modeliai pernelyg riboja paslaugų uždavinius, išskirstytus laiko ir erdvės atžvilgiu“ [5].

Visi aukščiau paminėti modeliai yra tinkami ne tik internetiniams procesams specifikuoti. Kadangi B2B sistemoms gyvybiškai svarbu funkcionuoti elektroninėje erdvėje, yra siūloma keletas internetui skirtų transakcijų protokolų. Pavyzdžiui Interneto Transakcijų Protokolas TIP (*angl. Transaction Internet Protocol*) [6]. Tai yra paprastas dviejų patvirtinimo fazių protokolas, pagrįstas dvejais TCP/IP susijungimais. Palyginus su kitais internetinėms aplikacijoms skirtais metodais, jis turi dvi naujoves. Pirma - įtraukimo (*angl. pull*) metodas – nauja subtransakcija, gali būti įtraukiama į bendrą transakciją jos. vykdymo metu. Antra – perduodamas patvirtinimas (*angl. delegated commit*) – transakcijos apribojimai yra kontroliuojami iš kliento taikomosios programos, o tarnybinė stotis vykdo tik patvirtinimus ir atšaukimus. Toks siūlymas labai tinka internetinėms taikomosioms programoms, kur kliento pusėje nėra atsinaujinančių resursų.

1.1.1 Komerciniai standartai ir produktai

Yra vykdoma ir keletas komercinių projektų, susijusių su verslo transakcijų modeliavimu. Pvz. exotica projektas [7], [8], kuriame buvo pasiūlyti metodai ir įrankiai transakcijų modeliavimui. Jo pagrindinė užduotis – pasiūlyti vartotojams išplėstinį darbų sekų modelį, grįstą transakcijų samprata. Vartotojas specifikuoja verslo užduotis, o taip pat ir jas kompensuojančias užduotis. Procesorius pakeičia šiuos aprašus į FDL (*angl. Flowmark Definition Language*) kalbą, į reikiamas vietas (po vienos ar kelių užduočių) įterpdamas jas kompensuojančių užduočių aprašus. Pažymėtina, kad tai yra IBM kompanijos produktas ir skirtas būtent šios kompanijos platformoms ir sprendimams.

[9] pasiūlytas transakcijų modelis skirtas HP Changengine (dabar Process Manager). Šiame modelyje aprašomos virtualios transakcijos VT (*angl. Virtual Transaction*) apibendrinančios tam tikrą darbų seką. Jei įvyksta klaida vykdant, kurią nors darbų sekos užduotį, tada visa apibendrinamoji seka atstatoma atvirkštine, nei vykdymas seka, kol pasiekiamas galutinis kompensacijos taškas (*angl. compensation end point*). Tada sistema sprendžia ar kartoti vykdymą nustatytą kiekį kartų ar vykdyti alternatyvias transakcijas ar baigti viso proceso vykdymą.

1.1.1.1 BTP protokolas

Kad apibrėžti išorinių organizacijų verslo procesų transakcijų protokolą BTP (*angl. Business Transaction Protocol*), standartų ir pramonės konsorciumas OASIS suformavo Verslo transakcijų techninį komitetą. BTP [10] – tai XML pagrindu sudarytas protokolas skirtas vykdyti transakcijas internetu. Kaip ir TIP naudoja dviejų fazių patvirtinimo protokolą 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. Šiame protokole pristatomi tokios konceptualios sąvokos kaip nedalomas vienetas (*angl. atom*) ir tamprumas (*angl. cohesion*), kurios „sušvelnina“ ACID reikalavimus. Taigi, BTP nedaloma (atominė) transakcija išlaiko ACID reikalavimus, tačiau tampri transakcija leidžia į sekų formavimo uždavinį įsikišti iš išorės, patvirtinant arba atšaukiant nedalomus darbų vienetus. Šis protokolas taip pat apibrėžia ir aktorius, bei ryšius tarp jų. Yra du pagrindiniai ryšiai: valdymo ir aukštesnio – žemesnio (*angl. inferior - superior*). Valdymo ryšys – tai ryšys tarp baigties elemento (*angl. Terminator*), (elemento, kuris nustato transakcijos pabaigą) ir aktoriaus (*angl. Decider*) (BTP 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.
Jau yra nemažai komercinių produktų sukurtų BTP pagrindu. Pvz. Hewlett Packard firmos Web Services Transaction (*HP-WST*) [11] Šiam produktui nepavyko užimti didelės rinkos

dalies, kadangi jame naudojama daug ir sudėtingų pranešimų struktūrų, nėra pakankamai gerai paruoštų priemonių lankstiems būsenų atstatymo mechanizmams. Pavyzdžiui, jei atominėje transakcijoje vienas iš elementų yra neįvykdomas, atšaukiami visi kiti dalyviai. Tai yra labai svarbu, ypač turint omenyje, kad interneto aplikacijose dėl ryšio sutrikimų ar panašių veiksnių gana dažnai pasitaiko neįvykdomų užduočių.

1.1.1.2 Elektroninių ryšių valdymas ebXML

EbXML [12] naudojama metodologija verslo procesą apibrėžia kaip verslo bendradarbiavimą, tai yra, dviejų ar daugiau partnerių veiklą siekiant gauti tam tikrą rezultatą. Tai yra gan nauja technologija. EbXML projektas 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 International Standartization Organization (ISO) patvirtino keturių ebXML OASIS standartų rinkinį. ISO/TS 15000 techninės specifikacijas sudaro keturios dalys, kurių kiekviena atitinka vieną ebXML modulio standartą:

- ISO 15000-1: ebXML partnerio profilio susitarimo specifikacija (ebCPP - Collaborative Partner Profile Agreement Specification).
- ISO 15000-2: ebXML pranešimų apdorojimo specifikacija (ebMS - Messaging Service Specification).
- ISO 15000-3: ebXML registrų informacijos modelis (ebRIM - Registry Information Model).
- ISO 15000-4: ebXML registrų servisų specifikacija (ebRS - Registry Services Specification).

Svarbu suprasti kad ebXML nėra viena technologija, tai greičiau specifikacijų rinkinys skirtas kurti e-verslo struktūrą.

Projekto tikslas buvo suformuluotas ebXML projekto kūrimo pradžioje: „Suteikti atvirą XML grįstą infrastruktūrą, kuri leistų plačiai naudoti elektroninio verslo informaciją veikiančiu, saugiu bei pastoviu būdu visoms šalims“ [13]. Mažos ir vidutinio dydžio įmonės tai pat gali naudoti elektroninio B2B (Business to Business) transakcijas per ebXML.

EbXML komponentai:

- ebXML partnerio profilio susitarimo specifikacija (ebCPP).
- Šios specifikacijos tikslas yra užtikrinti veikimą tarp dviejų informacinių sistemų, net jei jos yra skirtingų kūrėjų.

- ebXML pranešimų apdorojimo specifikacija (ebMS).
- Ši specifikacija apibrėžia pranešimų formatą ir antraštinį dokumentą naudojamus ebXML pranešimų persiuntimui per tokius protokolus kaip SMTP ar HTTP. Taip pat apibrėžia programinės įrangos veiksmus, kuri siunčia ir priima ebXML pranešimus. Ši programinė įranga gali būti realizuota kaip savarankiškas pranešimų valdiklis arba gali būti e-verslo funkcionalumo integracijos produktas, arba programinis servisas.
- ebXML registrų servisų specifikacija (ebRS).
- ebXML registrai suteikia stabilų verslo subjekto įvedamų duomenų saugojimą. Tokia informacija yra naudojama, kad palengvintų ebXML grįstą B2B bendradarbiavimą bei transakcijas. Įvestas turinys gali būti XML schema ir dokumentai, procesų aprašai, ebXML branduolio komponentai, konteksto aprašymai, UML modeliai, informacija apie šalis ir programinės įrangos komponentus.. Ši paviešinta informacija yra saugoma saugykloje ir ją valdo ebXML registrų servisai. Štai tokia yra esmė.
- ebXML registrų informacijos modelis (ebRIM).
- Registro informacijos modelis yra aukšto lygio ebXML registrų schema. Ji apibrėžia registre saugomų objektų tipus ir saugomų objektų organizavimą registre.
- Verslo procesų specifikavimo schema (ebBPSS).

EbBPSS tikslas yra apjungti e-verslo procesų modeliavimą ir e-verslo programinių komponentų specifikavimą. Tiksliau tarant: ebBPSS yra specifikavimo rinkinys, būtinas apibrėžiant dviejų verslo partnerių bendradarbiavimą. Negana to, ji nustato partnerių realaus laiko sistemų parametrus, tam kad e-verslo programiniai komponentai galėtų bendradarbiauti [14].

1.1.1.3 Elektroninio verslo modelių sudarymo kalba XLANG

XLANG [15] yra vietinės reikšmės kalba. Ji naudojama Microsoft Biz Talk serveryje. Ji skirta formaliam verslo procesų specifikavimui.

XLANG kiekvieno aktoriaus servisų sąsajų specifikavimui naudoja WSDL kalbą (*angl. Web Services Description Language*). Operacijų duomenų srautų srautai nespecifikuodami. Taikoma ir ilgos trukmės kompensuojamos transakcijos samprata. Transakcijas galima rinkti į rinkinius. Kompensacijos taikomos klaidų apdorojimui. XLANG siūlo lanksčius įrankius klaidų apdorojimo ir būsenų atstatymo uždaviniams.

Verslo procesus ji apibūdina kaip duomenų apsikeitimą tarp aktorių (tinklinių paslaugų sistemų). Darbų seka apibrėžiama kaip blokinė struktūra, kur procesai gali būti dar ir dekomponuojami. Ryšiai tarp darbų sekos elementu nusakomi įvairiais nuosekliais, lygiagrečiais ar sąlyginiais veiksmais.

Nors XLANG neapibrėžia kokios veiksmų sekos gali būti prižįstamos transakcijomis, tačiau transakcijos kaip verslo proceso sudėtinės dalies samprata išlieka. Transakcijų struktūros aprašomos transakcijų blokais, kuriuos sudaro bet koks skaičius transakcijų. Su kiekvienu tokiu bloku galima susieti ir kompensuojančią transakciją. Jei įvyksta klaida, vykdomos kompensuojančios transakcijos, kurioms galima nustatyti vykdymo tvarką, tačiau pagal nutylėjimą, jos vykdomos atvirkštine tvarka.

1.1.2 Veiklos procesais grindžiamas IS modeliavimas

Darbe [16] buvo atlikta verslo transakcijų modelių lyginamoji analizė. Buvo prieita išvados, kad verslo transakcija skirtingai suprantama įvairiuose modeliuose ir neturi griežto formalaus apibrėžimo. Yra tokių modelių ir protokolų (pvz. XLANG, BTP), kurie visiškai nepaaiškina kokios veiksmų sekos gali būti laikomos transakcijomis. Dažniausiai verslo transakciją siūloma traktuoti kaip UML (*angl. Unified Modelling Language*) kalbos panaudojimo atvejis, t.y. apibrėžtų veiksmų seka, kuri duoda kažkokį reikšmingą rezultatą, tačiau [5] pažymima, kad „toks traktavimas sukelia problemas dėl panaudojimo atvejų detalizavimo laipsnio neapibrėžtumo“.

Šiuolaikinių informacinių sistemų projektavime, paprastai pradedama nuo veiklos procesų modeliavimo. Šiame darbe dėmesys sutelktas į elektroninio verslo sistemas. Dažnai jos apima kelias bendradarbiaujančias organizacijas, todėl apibrėžiant darbų sekas, modeliavimas yra orientuotas tiek į vidinius, tiek į išorinius organizacijų B2B procesų modeliavimą. Šiame darbe plėtojamas IS modeliavimas, grindžiamas paslaugų požiūriu, t.y. elektroninio veiklos procesų modeliavimo rezultatas – vykdomo proceso schema susieta su tinklo paslaugų (*angl. web services*) sistema.

Šiame darbe verslo transakcija apibrėžiama naudojant [5] suformuluotus principus. Ji suprantama kaip „nedalomas veiklos vienetas, vaizduojamas pragmatiškai motyvuota išplėstine komunikacine kilpa. Verslo transakcijos yra EM modelio elementai. EM modelį sudarys tarpusavyje susijusių veiklos procesų aibė, kurioje kiekvienas procesas sudarytas iš pragmatiškai motyvuotų komunikacinių kilpų. Įvedus tokį veiklos vienetą, galima užduoti kriterijų panaudojimo atvejamas detalizuoti“[5]. Šiame darbe didžiausias dėmesys skiriamas taip aprašytų verslo transakcijų duomenų struktūrų apibrėžimams, naudojant XML standartą.

1.2 XML duomenų standarto analizė

Esant keliems verslo partneriams, natūralu, jog jų informacinės sistemos nėra vienodos, jų realizacijos yra skirtingos. Tačiau bendradarbiavimo tikslas yra vykdyti bendrą veiklą, vykdyti informacinius mainus, keistis informacija. Štai čia ir iškyla standartizuoto duomenų formato poreikis. Šiuo metu yra labai populiarus bei plačiai naudojamas XML standartas. Pamėginsime šį standartą glaustai apibūdinti, atskleisti jo privalumus bei trūkumus.

1.2.1 Tinkamai suformuoti XML dokumentai

Tinkamai suformuoti (angl. well-formed) XML dokumentas turi teisingą XML sintaksę. Tačiau XML dokumentus tik tinkamai sutvarkyti dažniausiai nepakanka

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<priminimas>
<kam>Antanui</kam>
<nuo>Karolio</nuo>
<antraste>Svarbus priminimas</antraste>
<turinys>Nepamiršk skirti laiko poilsiui!</turinys>
</priminimas>
```

Tinkamai suformuotas XML dokumentas yra dokumentas, kuris išpildo XML sintaksės taisykles:

- Turi prasidėti XML aprašymu.
- Turi turėti vieną unikalų šakninį elementą.
- Visos pradinės žymės turi užsibaigti tomis pačiomis galinėmis.
- XML žymės yra jautrios simbolių dydžiams.
- Visi elementai turi būti uždaryti.
- Visi elementai turi būti teisingai sudėlioti lygiuose (angl.nested).
- Visos atributų reikšmės turi būti tarp kabučių.
- XML esybės turi būti naudojamos specialiems simboliams.

Netgi jei dokumentai yra tinkamai suformuoti, jie vis tiek dar gali turėti klaidų, kurios gali sukelti rimtų padarinių. Tarkime tokia situacija: užsisakote 5 lazerinių spausdintuvų dydžius, vietoje 5 lazerinių spausdintuvų. Su XML schemomis, dauguma šių klaidų gali būti išgaudytos atliekant tikrinimą.

1.2.2 Teisingi XML dokumentai

Teisingi (angl. valid) XML dokumentas yra tinkamai suformuotas XML dokumentas, kuris taip pat atitinka DTD (Dokument Type Definition – dokumento tipo apibrėžimo) taisykles:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE priminimas SYSTEM "priminimas.dtd">
<priminimas>
<kam>Antanui</kam>
<nuo>Karolio</nuo>
<antraste>Svarbus priminimas</antraste>
<turinys>Nepamiršk skirti laiko poilsiui!</turinys>
</priminimas>
```

1.2.3 XML DTD

DTD apibrėžia teisingus XML dokumento elementus. DTD tikslas yra apibūdinti XML struktūrą su tinkamų elementų sąrašu. DTD gali būti aprašytas pačiame XML dokumente arba nuoroda į išorinį šaltinį.

Vidinis dokumento tipo DOCTYPE aprašymas. Jei DTD yra įtraukiamas XML kodo dokumentą, jis turi būti apgaubiamas DOCTYPE aprašyme, kaip pateikiamame pavyzdyje:

```
<!DOCTYPE šakninis-elementas [elementu-apibrežimai]>
```

```
<?xml version="1.0"?>
<!DOCTYPE priminimas [
  <!ELEMENT priminimas (kam,nuo,antraste,turinys)>
  <!ELEMENT kam      (#PCDATA)>
  <!ELEMENT nuo      (#PCDATA)>
  <!ELEMENT antraste (#PCDATA)>
  <!ELEMENT turinys  (#PCDATA)>
]>
<priminimas>
<kam>Antanui</kam>
<nuo>Karolio</nuo>
<antraste>Svarbus priminimas</antraste>
<turinys>Nepamiršk skirti laiko poilsiui!</turinys>
</priminimas>
```

Pateiktas DTD pavyzdys interpretuojamas taip:

!DOCTYPE priminimas (eilutė 2) apibrėžia, jog tai yra **priminimas** tipo dokumentas.

!ELEMENT priminimas (eilutė 3) apibrėžia, jog elementas **priminimas** turi turėti keturis elementus: "kam, nuo, antraste, turinys".

!ELEMENT kam (eilutė 4) apibrėžia jog elementas **kam** yra "#PCDATA" tipo.

!ELEMENT nuo (eilutė 5) apibrėžia jog elementas **nuo** yra "#PCDATA" tipo ir taip toliau.....

1.2.4 XML schema

XML schema yra XML pagrindu veikianti alternatyva DTD. W3C (internetinių standartų institucija) pritaria pavadintai XML schemai, kaip alternatyvai DTD.

1.2.4.1 Kas yra XML schema?

XML Schemos tikslas yra apibrėžti teisingą blokų formavimą XML dokumente, taip kaip ir DTD:

- Apibrėžia elementus, kurie gali pasitaikyti dokumente.
- Apibrėžia atributus, kurie gali pasitaikyti dokumente.
- Apibrėžia elementus, kurie yra paveldėti elementai.
- Apibrėžia paveldėtų elementų tvarką.
- Apibrėžia paveldėtų elementų skaičių.
- Apibrėžia ar elementas yra tuščias ar gali savyje talpinti tekstą.
- Apibrėžia duomenų tipus elementams ir atributams.
- Apibrėžia numatytas ir fiksuotas elementų ir atributų reikšmes.

1.2.4.2 XML schemos yra pažangesnės už DTD

Manoma, kad labai greitai XML schemos bus naudojamos daugumoje internetinių aplikacijų vietoje DTD. Štai kelios priežastys:

- XML schemos yra pratęsiamos ateities naudojimui (pakartotinė inžinerija).
- XML schemos yra turiningesnės ir labiau panaudojamos už DTD.
- XML schemos yra aprašytos XML standartu.
- XML schemos apibrėžia duomenų tipus.
- XML schemos apibrėžia apibrėžtis (angl. namespaces).

1.2.4.3 XML schema turi duomenų tipų palaikymą

Viena iš stipriausių XML schemų pusių yra duomenų tipų palaikymas. Naudojant duomenų tipų palaikymą:

- Yra lengviau apibrėžti leistiną dokumento turinį.
- Yra lengviau tikrinti dokumento teisingumą.
- Yra lengviau dirbti su duomenimis iš duomenų bazės.
- Yra lengviau apibrėžti duomenų apribojimus.
- Yra lengviau apibrėžti duomenų formatus.
- Yra lengviau transformuoti duomenis iš vienų į kitus duomenų tipus.

1.2.4.4 XML schemos naudoja XML sintaksę

Kita XML schemų stiprybė yra tai, jog jos parašytos XML kalba. Tai suteikia tokius privalumus:

- Nereikia mokintis dar vienos kalbos.
- Galima naudoti XML redagavimo įrankius XML schemų kūrimui.
- Galima naudoti XML analizatorių XML schemų peržiūrai.
- Galima manipuliuoti schemomis su XML DOM.
- Galima transformuoti schemas su XSLT.

1.2.4.5 XML schemų saugi duomenų komunikacija

Kai duomenys yra siunčiami iš siuntėjo pas gavėją svarbiausia yra tai, kad abi pusės turėtų vienodą „supratimą“ apie turinį. Su XML schemomis, siuntėjas gali aprašyti duomenis taip, kad gavėjas tai suprastų. Data pvz., tokiu formatu: "03-11-2004", kai kuriose šalyse bus interpretuojama kaip lapkričio 3-ioji, o kitose kaip kovo 11-oji, bet XML elementas su duomenų tipu <date type="date">2004-03-11</date> užtikrina turinio suderinamumą, kadangi XML duomenų tipas „date“ yra apibrėžtas metai-mėnuo-diena (YYYY-MM-DD) formatu.

1.2.4.6 XML schemos yra praplečiamos

XML schemos yra praplečiamos, lygiai kaip ir pats XML, kadangi jos parašytos XML kalba. Su praplečiamais schemų aprašymais galima:

- Pakartotinai panaudoti schemas kitose schemose.
- Kurti savo pačių tipus iš standartinių tipų.
- Įtraukti kelias schemas į vieną dokumentą.

1.2.5 XSL ir XSLT

XSL – praplečiama stilių kalba (Extensible Stylesheet Language)

World Wide Web Consortium (W3C) pradėjo kurti XSL, kadangi buvo poreikis XML grįstai stilių aprašymo kalbai.

XSLT reiškia XSL transformacijas. Apibūdinsime, kaip naudoti XSLT norint transformuoti XML dokumentus į kitus formatus, kaip pvz. XHTML.

1.2.5.1 Kas yra XSLT?

- XSLT reiškia XSL transformacijas.
- XSLT yra svarbiausioji XSL dalis.
- XSLT transformuoja XML dokumentą į kitus XML dokumentus.
- XSLT naudoja XPath navigacijai po XML dokumentus.
- XSLT yra W3C rekomendacija.

Per paskutiniuosius kelerius metus XSLT (Extensible Style Sheet Language Transformation) tapo plačiai naudojama galinga kalba skirta transliuoti ir transformuoti XML dokumentus į kitus XML, tekstinius, HTML ir panašius dokumentus. XSLT yra žymėmis (angl. tags) grįsta aprašymo kalba, kuri padeda išrinkti specialias žymes/duomenis XML dokumento viduje. Vis dėlto, dabartinėje XSLT 1.0 versijoje yra apribojimai, kurie gali stilių aprašymą padaryti sudėtingu ir komplikuotu. XSLT 2.0 savyje laiko kelias naujoves, kurios padaro XSL algoritmus paprastesniais įdiegti ir taip pat padaro juos nepriklausomus nuo kūrėjo.

Oracle XML Developer's Kit 10g palaiko pagrindines XSLT 2.0 savybes. Oracle XDK taip pat palaiko ir DOM 3.0 bei XPATH 2.0 specifikacijas.

Normaliai XSLT paverčia XML suprantamą naršyklėms transformuodama kiekvieną XML elementą į (X)HTML elementą. O kaip žinia, interneto naršyklių pirminė užduotis ir yra (X)HTML dokumentus interpretuoti ir suinterpretuotą informaciją pateikti intertneto svetainių lankytojams.

Su XSLT galima įtraukti/išmesti elementus, atributus norint suformuoti išeigos failą. Taip pat galima pertvarkyti ir surūšiuoti elementus, atlikti testus ir padaryti išvadas – kuriuos elementus geriau paslėpti, kuriuos geriau rodyti, ir kur kas daugiau.

Transformacijos procesą galima apibūdinti sakant, kad XSLT transformuoja XML pradinių duomenų medį į XML rezultatų medį.

1.2.5.2 XSLT naudoja XPath

XSLT naudoja XPath informacijos radimui XML dokumente. XPath yra naudojamas navigacijai per elementus ir atributus XML dokumentuose.

1.2.5.3 Kaip tai veikia?

Transformavimo proceso metu, XSLT naudoja XPath pradinio dokumento dalių apibrėžimui kad tai turėtų atitikti vieną ar daugiau iš anksto apibrėžtų šablonų. Kai yra surandamas atitikimas, XSLT transformuoja atitinkančią pradinio dokumento dalį į rezultato dokumentą.

1.3 RDBVS aplinkų įrankiai XML duomenų valdymui

1.3.1 XML formato duomenų mainiai Oracle duomenų bazėje

1.3.1.1 Ryšiais ir objektiškai grįstų duomenų pavertimas XML

XML greitai tapo pagrindiniu duomenų dalinimosi tarp atskirų informacinių sistemų mechanizmu. IT (informacinių technologijų) įmonės sprendžia ar įsigyti prigimtines XML duomenų bazes ir perversti turimus duomenis iš reliacinių ir objektiškai grįstų talpų į XML modelį, kuriuo galima dalintis su verslo partneriais. Šiame skyrelyje apžvelgsime, kaip naudoti Oracle Database 10g vietoj jau turimų reliacinių ir objektiškai grįstų duomenų ir paruošti juos tokiu XML, kokio reikia.

1.3.1.2 Oracle istorijos atgarsiai

Naujų duomenų formato pritaikymas ar naujų saugojimo mechanizmų suderinimas nėra nauji dalykai Oracle duomenų bazėje. Devintajame dešimtmetyje, objektiškai orientuoto programavimo (OO) specialistai nusprendė, kad nesutapimai tarp objekto su duomenų atributų ir reliacinių lentelių su stulpeliais, buvo paprasčiausiai per daug painu kūrėjų komandoms. Tai buvo objektinės revoliucijos priežastis, kai objektiškai orientuota duomenų bazė pradėjo varžytis su didelių reliacinių duomenų bazių valdymo sistemomis (RDBMS).

Oracle pastebėjo objektiškumo palaikymo reikalingumą savo duomenų bazėse, taigi pateikė kelias iniciatyvas tam įdiegti. Oracle pristatė Java Virtual Machine (Java virtualiąją mašiną) duomenų bazės branduolyje; naujas duomenų konstrukcijas, tokias kaip Object Type (objektų tipai) arba User-Defined Type (UDT – vartotojo aprašyti tipai); ir, ko gero svarbiausia, objektų vaizdus (object view). Objektų vaizdai davė duomenų bazių administratoriams ir programuotojams galimybę kurti duomenų bazės vaizdus iš UDT, kuriuose rezultatų rinkinys yra objektų rinkinys vietoje reliacinių lentelių rinkinio.

Ši objektų vaizdų koncepcija buvo svarbiausia, todėl kad ji suteikė Oracle parduotuvėms galimybę pakeisti jų naudotas programas, raportavimo įrangą ir sandėlius.

Objektai vis dar yra didelė Oracle strategijos dalis, jie yra visiškai palaikomi Oracle Database 10g programiniame pakete. Šiandien, industrija mažiau sukoncentravusi dėmesį į objektų naudojimą (pagrindinė priežastis – ta, kad tai jau padaryta). Taip pat industrija

reikalavo iš duomenų bazių objektų palaikymo devintam dešimtmety, kaip dabar reikalauja XML galimybių.

Aprašysime Oracle duomenų bazės sugebėjimą palaikyti XML galimybes, pvz.: naujus XML duomenų tipus, XML programavimo palaikymą interpretatoriuose (angl. parsers) ir XSL transformacijas, XML Schemos naudojimą bei XML saugyklų nustatymus. Sukoncentruojame dėmesį į naują duomenų bazės funkcionalumą: XML generavimą. Per Objektinę revoliuciją, duomenų bazės buvo pritaikytos reliacinius duomenis pateikti vartotojo apibrėžtų tipų formatu. Šiandien duomenų bazė gali duomenis paversti XML. Pateikiame konkretų pavyzdį, kaip paruošti reliacinius ir objektiškai orientuotus duomenis XML formatu.

Pažvelkime, kaip galime paimti duomenų modelį, sukurti XML vaizdą ir pateiktį jį internete kaip RSS (Rich Site Summary – turtinga tinklalapio santrauka) naujienų šaltinį. Tai geras pavyzdys, kadangi pamatinis duomenų modelis gali būti labai paprastas. XML formatas nėra labai sudėtingas. Žinojimas kaip pateikti RSS iš Oracle duomenų bazės yra puikus pirmas žingsnis link XML generavimo galimybių supratimo Oracle Database 10g.

1 lentelė. Užklausa, išrenkanti nurodytame skyriuje esančių asmenų duomenis

```
select e.vardas, e.pavarde,
       e.idarbinimo_data
  from (select vardas,
               pavarde,
               idarbinimo_data,
               skyriaus_id
          from darbuotojai
         order by idarbinimo_data desc) e
 where skyriaus_id = 80
   and rownum < 6
/
```

Naudojame gerą sąjungą SQL/XML, PL/SQL su DBUri (DBUri servletu), norėdami pateikti ataskaitą RSS formatu.

Pirmiausia, sukuriame PL/SQL funkciją, kuri paima skyriaus ID (skyriaus identifikacinis numeris) ir grąžina XMLType (XML tipo) objektą, suformuotą kaip RSS dokumentą, kaip pavaizduota pirmajame pavyzdyje.

Kodo pavyzdys 1: PL/SQL funkcija, grąžinanti XMLType objektą kaip RSS dokumentą

```
create or replace function gauti_skyriu_xml (p_skyriaus_id in number)
     return xmltype
    as
      -- tai bus naudojama XML dokumento sudarymui ir grąžinimui
      l_xml xmltype;
    begin
      --  naudosime  SQL/XML  užklausą  RSS  sugeneravimui  ir  grąžinimui  į
lokalųjį kintamąjį
      select xmlelement( "rss",
               xmlattributes( 'v.2.1' as "version" ),
               xmlelement( "channel",
                 xmlforest( 'Motomanų skyrius' as "title",
                            'http://www.motomanai.lt' as "link",
                            p_skyriaus_id "description",
                            'lt' as "language",
                            'motomanai.lt 2003-2006, MOTOMANAI LT'
                              as "copyright"
                 ),
```

```
                                xmlelement("image",
                                  xmlforest('Motociklininkų sezono atidarymas' as "title",
                                            'http://www.motomanai.lt/atidarymas.jpg'
                                              as "url",
                                            'http://www.motomanai.lt/atidarymas.html'
"link",
                                            '50' as "width",
                                            '50' as "height",
                                            'pavyzdinis RSS dokumentas'
                                            as "description")
                                ),
                                ( select xmlagg(
                                           xmlelement( "item",
                                             xmlforest(x.vardas||' '||
                                               x.pavarde||',      '||x.idarbinimo_data      as
"title",
                                               '/oradb/DRB/DARBUOTOJAI/ROW[DARBUOTOJO_ID='
||
                                                 x.darbuotojo_id || ']' as "link",
                                               x.idarbinimo_data as "description" ) ) )
                                    from ( select e.vardas, e.pavarde, e.idarbinimo_data,
                                                  e.darbuotojo_id, d.skyriaus_pavadinimas
                                             from darbuotojai e, skyriai d
                                            where d.skyriaus_id = p_skyriaus_id
                                              and d.skyriaus_id = e.skyriaus_id
                                            order by e.idarbinimo_data desc) x
                                   where rownum < 6 ) ) )
                      into l_xml
                      from dual;
                    --
                    return l_xml;
                  end gauti_skyriu _xml;
```

Verta paminėti, kad funkcijos, kurias matome pirmajame pavyzdyje, – XMLElement(), XMLAttributes(), XMLForest() bei XMLAgg() – yra SQL/XML dalis, atsirandantis standartas, kuris leidžia SQL užklausoms grąžinti XML dokumentus.

XMLElement() funkcija sukuria XML elementą dokumente. Duoti šiai funkcijai parametrai parodo ar elementas turės atributus, ar lizdinį XML. XMLAttributes() funkcija sukuria vieną ar daugiau atributų apibrėžtų konkrečiam elementui, ir XMLForest() funkcija sukuria vieną ar daugiau pasirinktų elementų. Galiausiai XMLAgg() funkcija sudaro eilutes į XML gijų sankaupą. Šios SQL/XML funkcijos duoda mums visą lankstumą, kurio mums reikia, norint sukurti gilius lizdinius XML dokumentus su neribotu lankstumu.

Kitas žingsnis yra sukurti XMLType vaizdą, kuris grąžintų RSS dokumentą kiekvienam SKYRIAUS_ID iš SKYRIU lentelės. Iškviečiame SKYRIU lentelei SKYRIAUS_ID, kuris perduodamas GAUTI_SKYRIU_XML funkcijai, gaudame RSS dokumento vaizdą kiekvienai eilutei SKYRIU lentelėje:

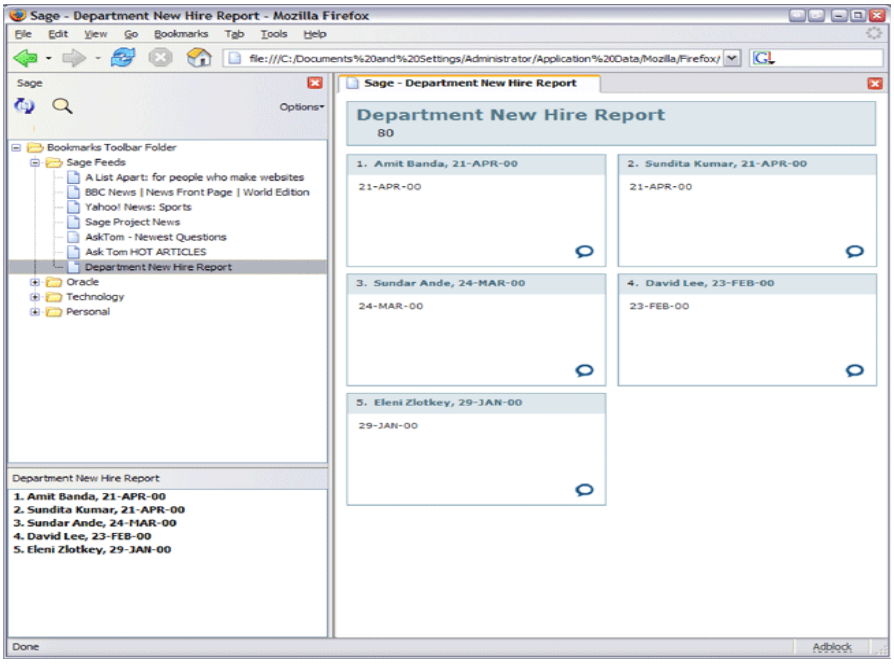
```
create or replace view
idarbinimas_skyriuose
of xmltype with object id (
  extract(object_value,
  '/rss/channel/description/text()')
  .getnumberval()) as
  select gauti_skyriu_xml(
         skyriaus_id)
    from skyriai
/
```

Šiame SQL apibrėžime sukuriame XMLType vaizdą (angl. view) pavadintą IDARBINIMAS_SKYRIUOSE. Kiekvienas XMLType vaizdas privalo turėti unikalų objekto ID, kuris pateikiamas kaip unikalus identifikatorius eilutei ar vaizdui. Sukuriame objekto ID ištraukdami SKYRIAUS_ID, naudojant XPath išraišką ('/rss/channel/description/text()') iš OBJECT_VALUE. OBJECT_VALUE yra tariamas stulpelis, sukurtas vaizdo užklausos metu. Kuomet vaizdas yra sukuriamas, viskas ką mums reikia padaryti tai jį iškviesti naudojant interneto naršyklę.

Norint perduoti šiuos duomenis iš SQL vaizdo į internetą, naudosime duomenų bazės savybę pavadintą DBUri Servlet. Ji leidžia peržiūrėti lenteles ar vaizdus Oracle duomenų bazėje per internetą. DBUri užklausa yra suformuojama taip: interneto naršyklėje DBUri užklausos XML rezultatas atrodo štai taip:

```
- <rss version="0.91">
  - <channel>
      <title> Motomanų skyrius</title>
...
```

Norint pamatyti suformuotą XML rezultatą, reikia naudoti RSS skaitymo priemonę kaip, pvz., „Sage“ – RSS skaitymo priemonė, kuri yra priderinta prie Mozilla Firefox naršyklės. Pridėjus nuorodą į rezultatą „Sage“ pateikia tokį vaizdą, kaip parodyta paveikslėlyje 1.



1 pav. RSS naujienų XML vaizdas Sage RSS skaitymo priemonėje

1.3.2 XML valdymo įrankiai MS *SQL Server*

XML naudojamas duomenų mainams:

- Business to Business (B2B).
- Business to Consumer (B2C).
- Intra-Business

Microsoft SQL Server 2000 XML palaikymo objektai

- Suderinamumas su HTTP
- SQL užklausos SELECT išplėtimas (FOR XML) leidžiantis gauti rezultatus XML formatu.
- XML rėžimai
- XML rodiniai
- XPath užklausos
- OPENXML

1.3.2.1 Suderinamumas su HTTP

SQL sakiniai adresų laukelyje (URL). SQL sakinių ir „stored procedures“ įrašyms ir vykdymas tiesiai iš adresų laukelio naršyklėje, palengvina duomenų pasiekiamumą duomenų bazėje.

Šablonai. Tai yra iš anksto parengti XML dokumentai, kuriuose saugoma informacija apie XML užklausas. Jie gali būti pasiekiami iš adreso lauko, todėl jie labai tinka gauti duomenis nežinant konkrečios DB struktūros. Šablono duomenų išrinkimui gali būti naudojamos šios priemonės:

- SELECT sakiniai
- „Stored procedure“ vykdymas
- Parametrų perdavimas
- XPath užklausos.

HTML POST. Tai yra HTML formų kintamųjų perdavimas apdorojimui. MS SQL Server 2000 aplaiko XSL (*angl. Extensible Stylesheet Language*), t.y. stilių failus, kurie yra panaudojami duomenų išvedimui į ekraną.

1.3.2.2 XML režimai (FOR XML)

SQL sakiniuose naudojami raktažodžiai FOR XML, gražina ne standartinius eilučių rinkinius, o XML formato duomenis. Priklausomai nuo poreikių, galima panaudoti kelis formavimo būdus (rėžimus). RAW, AUTO, EXPLICIT

1.3.2.2.1 RAW

RAW rėžimas kiekvieną gautą rezultatų eilutę transformuoja į XML elementą, o stulpelių reikšmes pateikia kaip atributus, pvz.:

```
Užklausa:
SELECT '<root>';
SELECT TOP 3 DarbID, Vardas, Pavarde FROM Darbs FOR XML

Rezultatas: RAW, XMLDATA
<Schema name="Schema1" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoft-com:datatypes">
<root>
<ElementType>
<AttributeType name="DarbID" dt:type="i4" />
<AttributeType name="Vardas" dt:type="string" />
<AttributeType name="Pavarde" dt:type="string" />
<attribute type="DarbID" />
<attribute type="Vardas" />
<attribute type="Pavarde" />
</ElementType>
</Schema>
<row xmlns="x-schema:#Schema1" DarbID="1" Vardas="Karolis"
Pavarde="Pilypaitis" />
<row xmlns="x-schema:#Schema1" DarbID="2" Vardas="Antanas"
Pavarde="Petrauskas" />
</root>
```

1.3.2.2.2 AUTO

AUTO režimas rezultato eilutes grąžina XML medžio struktūra, kiekvienam stuplepliui išskirdamas po naują XML elementą, pvz.:

```
Užklausa:
SELECT '<root>';
SELECT TOP 3 DarbID, Vardas, Pavarde FROM Darbs as
```

```
DarbDetail FOR XML AUTO, ELEMENTS, XMLDATA
SELECT '</root>';
```

Rezultatas: AUTO, XMLDATA

```
<root>
<Schema name="Schema1" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoftcom:
datatypes">
<ElementType name="DarbDetail" content="eltOnly" model="closed" order="many">
<element type="DarbID" />
<element type="Vardas" />
<element type="Pavarde" />
</ElementType>
<ElementType name="DarbID" content="textOnly" model="closed" dt:type="i4" />
<ElementType name="Vardas" content="textOnly" model="closed" dt:type="string"
/>
<ElementType name="Pavarde" content="textOnly" model="closed"
dt:type="string" />
</Schema>
<DarbDetail xmlns="x-schema:#Schema1">
<DarbID>1</DarbID>
<Vardas>Karolis</Vardas>
<Pavarde>Pilypaitis</Pavarde>
</DarbDetail>
<DarbDetail xmlns="x-schema:#Schema1">
<DarbID>2</DarbID>
<Vardas>Antanas</Vardas>
<Pavarde>Petrauskas</Pavarde>
</DarbDetail>
<DarbDetail xmlns="x-schema:#Schema1">
<DarbID>3</DarbID>
<Vardas>Vardenis</Vardas>
<Pavarde>Pavardenis</Pavarde>
</DarbDetail>
</root>
```

1.3.2.2.3 EXPLICIT

EXPLICIT rėžimas leidžia nurodyti kokios truktūros XML failas reikalingas, pvz.:

Užklausa:

```
SELECT '<root>';
SELECT
1 as Tag,
NULL as PARENT,
E.DarbID as [Darb!1!DarbID],
NULL as [DarbDetail!2!!element],
NULL as [Pravarde!3!!element],
NULL as [Pavarde!4!!element]
FROM (SELECT TOP 3 DarbID,Vardas,Pavarde from Darbs) E
UNION ALL
SELECT
```

```
2, 1,
E.DarbID as [Darb!1!DarbID],
NULL as [DarbDetail!2!!element],
NULL as [Pravarde!3!!element],
NULL as [Pavarde!4!!element]
FROM (SELECT TOP 3 DarbID,Vardas,Pavarde from Darbs) E
UNION ALL
SELECT
3, 2,
E.DarbID as [Darb!1!DarbID],
NULL as [DarbDetail!2!!element],
E.Vardas as [Pravarde!3!!element],
NULL as [Pavarde!4!!element]
FROM (SELECT TOP 3 DarbID,Vardas,Pavarde from Darbs) E
UNION ALL
SELECT
4, 2,
E.DarbID as [Darb!1!DarbID],
NULL as [DarbDetail!2!!element],
NULL as [Pravarde!3!!element],
E.Pavarde as [Pavarde!4!!element]
FROM (SELECT TOP 3 DarbID,Vardas,Pavarde from Darbs) E
ORDER BY [Darb!1!DarbID]
FOR XML EXPLICIT, XMLDATA
SELECT '</root>';
```

Rezultatas: EXPLICIT, XMLDATA

```
<root>
<Schema name="Schema1" xmlns="urn:schemas-microsoft-com:xml-data"
xmlns:dt="urn:schemas-microsoftcom:
datatypes">
<ElementType name="Darb" content="mixed" model="open">
<AttributeType name="DarbID" dt:type="i4" />
<attribute type="DarbID" />
</ElementType>
<ElementType name="DarbDetail" content="mixed" model="open" />
<ElementType name="Pravarde" content="mixed" model="open" />
<ElementType name="Pavarde" content="mixed" model="open" />
</Schema>
<Darb xmlns="x-schema:#Schema1" DarbID="1">
<DarbDetail>
<Pravarde>Karolis</Pravarde>
<Pavarde>Pilypaitis</Pavarde>
</DarbDetail>
</Darb>
<Darb xmlns="x-schema:#Schema1" DarbID="2">
<DarbDetail>
<Pravarde>Antanas</Pravarde>
<Pavarde>Petrauskas</Pavarde>
</DarbDetail>
</Darb>
<Darb xmlns="x-schema:#Schema1" DarbID="3">
```

```
<DarbDetail>
<Pravarde>Vardenis</Pravarde>
<Pavarde>Pavardenis</Pavarde>
</DarbDetail>
</Darb>
</root>
```

1.3.2.3 XML režimų savybės (options)

Vykdant XML užklausą, galima atlikti tokius pasirinkimus:

- **XMLDATA.** Grąžina XML dokumento schemą.
- **BINARY base64.** Dvejetainiai duomenis užkoduoti base64 formatu
- **ELEMENTS.** AUTO rėžime galiojantis pasirinkimas. Nustatoma ar stulpeliai grazinami kaip XML elemento atributai ar kaip atskiri subelementai.

1.3.2.4 XPath užklausos

SQL Server 2000 yra įdiegtas Xpath palaikymas. Tai yra kalba , kuria yra apibrėžiamos duomenų strultūros ir jų pateikimas XML formatu

1.3.2.5 OPENXML

OPENXML yra Transact-SQL kalbos aprašymai, kurie generuoja reliacinių duomenų transformacijas į XML ir atvirkščiai. Šie duomenis įrašomi į kompiuterio atmintį ir yra naudojami XML duomenims po transformacijos sudertinti su realiaciniais. OPENXML technologija užtikrina XML duomenų įrašymą į duomenų bazės lenteles, sakinių SELECT, INSERT, UPDATE ir DELETE vykdymą XML formato duomenims. Jei naudojamos saugomos procedūros (*angl. stored procedures*), XML dokumentas gali būti naudojamas kaip tekstinis *char, nchar, varchar, nvarchar, text* arba *ntext* tipo parametras.

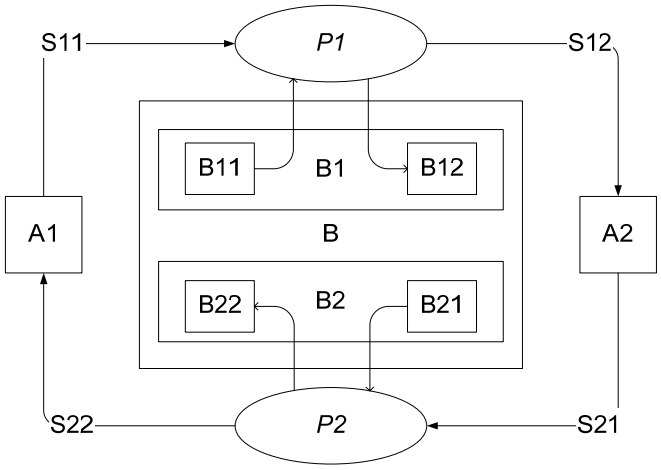
1.3.2.6 OLEDB, ADO ir XML

SQLOLEDB objektas buvo išplėstas, XML ir XPath palaikyti. Buvo integruota nauja **IcommandStream** sąsaja šablonų perdavimui apdoroti į OLE DB. Papildyta OLEDB specifikacija, palaikanti **IStream**, XML rezlutatų gražinimui per XPath užklausas, XML rodinių naudojimui. Šis objektas apibrėžtas ADO 2.5 versijoje ir yra naudojamas visose naujose versijose. Objektas naudojamas šablonų, XPath užklausų siuntimui, o taip pat XML relutatų priėmimui iš realicinių duomen bazių.

2 XML DUOMENŲ SRAUTŲ VALDYMO MODELIS NAUDOJANT KOMUNIKACINES KILPAS

2.1 Verslo transakcijų modeliavimas komunikacinėmis kilpomis

Komunikacinė kilpa [17] – būdas formaliai aprašyti verslo transakcijas tarp dviejų verslo vienetų. Kiekviena komunikacinė kilpa susideda iš dviejų aktorių – užsakovo A1 ir vykdytojo A2, duomenų srautų S11 – S22 ir dviejų procesų – užsakovo veiksmų P1 ir P2. Duomenų srautų dinamikos apibrėžiamos būsenų perėjimais B.



2 pav. Binarinė komunikacinė kilpa

Šis modelis buvo pasiūlytas nagrinėjant naujų funkcinių reikalavimų įvedimą jau sukurtoms ir veikiančioms informacinėms sistemoms.

Toks formalizavimo modelis leidžia užtikrinti kai kuriuos semantinio itegralumo kriterijus:

Darnos (angl. *coherence*) – galinės būsenos suderinamos su aktorių tikslais.

Pilnumo (angl. *completeness*) – nėra probleminių būsenų, kurios rodytų būtinybę tobulinti kompiuterizuotą sistemą.

Įgyvendinamumas (angl. *viability*) – komunikacinė kilpa įgyvendinama, jei

$A1 \xrightarrow[p]{} B11, A2 \xrightarrow[o]{} B22, A2 \xrightarrow[p]{} B21, A1 \xrightarrow[g]{} B12$, tuomet

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

čia simbolis $\longrightarrow$ žymi pragmatines priklausomybes: $\xrightarrow[g]{}$ reiškia tikslą, $\xrightarrow[p]{}$ problemą, $\xrightarrow[o]{}$ – progą, $\xrightarrow[+]{}$ – teigiamos įtakos priklausomybę ir $\xrightarrow[-]{}$ – neigiamos

įtakos priklausomybę. Kitaip tariant, komunikacinė kilpa yra įgyvendinama, jei recipiento problema neigiamai veikia iniciatoriaus problemą ir recipientas gali įvykdyti reakcijos veiksmą, kurio rezultatas teigiamai veikia tiek recipiento, tiek iniciatoriaus tikslus[18].

Įvedant naujus funkcinius reikalavimus, kinta ir duomenų srautai ir jų būsenos. Kai apsiribojame vieninteliu duomenų formatu – XML, komunikacinėmis kilpomis aprašytų transakcijų duomenų srautus patogu saugoti duomenų bazėje. Duomenų aprašams naudojami standartiniai XML dokumentų struktūros aprašai (DTD ar XSD) [21].

Šio darbo pirmoje dalyje paminėta, kad verslo transakcija neturi griežto formalaus apibrėžimo. Kai kuriuose transakcijų modeliavimo metoduose iš viso nepasakyta kokios būtent veiksmų sekos gali būti traktuojamos kaip verslo transakcijos.

Šiame darbe verslo transakcija suprantama kaip nedalomas verslo partnerių veiksmų vienetas. Keisdamiesi įvairaus paobūdžio duomenimis partneriai siekia įgyvendinti savus tikslus. Laikantis tokios koncepcijos, verslo transakcijų vaizdavimas pragmatiškai motyvuotomis komunikacinėmis kilpomis yra pats tinkamiausias.

Daugelyje verslo transakcijų modeliavimo kalbų (ebXML[13], XLANG[15] ir kt.), esminis dėmesys yra skiriamas verslo partnerių veiksmų aprašymui, jų sekai, susietumui. Įžvelgiama analogija su UML kalbos darbų sekų diagramoms. Dažnai toks modeliavimas yra susietas su konkrečia platformine realizacija (komerciniai produktai[10], [11]). Specifikuojant verslo transakcijas išplėstinėmis komunikacinėmis kilpomis yra aiškiai įvardijami aktoriai, nusakomi jų vykdomi veksmai, o taip pat apibrėžiami duomenų srautai ir jų būsenos, bei nustatomos būsenų priklausomybės. Toks požiūris yra nepriklausomas nuo platformos PIM (*angl. platform independent model*).

Verslo transakciją specifikuotą komunikacine kilpa galima sieti su UML kalbos panaudojimo atveju – veiksmų seka, kuri ją inicijavusiems aktoriams duoda kažkokį rezultatą. Kilpoje rezultatas yra pasiektas tikslas, kuris priešingai nei daugelyje kitų modelių yra duomenų būsena. Laikantis tokio principo galima sudaryti keletą kilpų vienam verslo procesui realizuoti. Vaizduojant duomenis reikia statiniais ir dinaminiais ryšiais susieti jų būsenas. Galimos tokios duomenų priklausomybės:

- *sin* – abipusio apibenrdinimo priklausomybės
- *composed_of* – agregavimo priklausomybė
- *exist* – gyvavimo priklausomybė
- $\subseteq$ - poaibio priklausomybė

Projektuojant IS, vienas esminių etapų yra verslo procesų identifikavimas ir formalus jų aprašymas. Verslo transakcijų specifikavimas komunikacinėmis kilpomis leidžia sąlyginai nesunkiai ir su nedideliais kaštais integruoti naujus funkcinius reikalavimus iškeltus IS. Taip pat jas galima panaudoti kuriant naują IS.

2.2 Duomenų valdymas tinklo tarnybinių stočių komponentuose

Plėtodami šio mokslinio tyrimo idėjas stengiamės didžiausią dėmesį kreipti į šių dienų verslo poreikių, keliamus uždavinius bei problemas. Ypatingą dėmesį skiriame savo patirties dalykinių sričių analizei. O tai būtent yra prekyba internetu, pvz. internetinė prekių gyvūnams parduotuvė, bei lizingo bendrovė. Internetinės parduotuvės keliamas tikslas yra didinti pardavimus.

Šiam tikslui pasiekti keliami įvairūs uždaviniai, pradedant kuo tikslesnių prekių katalogo atnaujinimu, patogiai pateikiant informacija ir baigiant visapusišku klientų poreikių tenkinimu. Šių dienų pirkėjas internetu pageidauja netik patogaus pristatymo į namus, tačiau vis dažniau tikisi brangesnius pirkinius įsigyti išsimokėtinai. Pastarojo poreikio patenkinimui elektroninė parduotuvė turi bendradarbiauti su lizingo paslaugas teikiančiomis bendrovėmis. Štai ir iškyla poreikis kompiuterizuoti šių verslo partnerių bendradarbiavimą. Klientas pageidauja iškart, dar besirinkdamas prekes matyti pirkimo išsimokėtinai sąlygas, išsirinkti jam labiausiai priimtiną variantą, galbūt net ir patvirtinti sutartį nuotolinių būdu per internetą. Deja šiuo metu tokios sutarties sudarymui nėra reikiamo teisinio pagrindo, tai pat trūksta elektroninio parašo įgyvendinimo realiame gyvenime. Nepaisant šių trikdžių jau dabar galima viską paruošti kompiuterizuotame formate ir klientui belieka ateiti pas lizingo bendrovės partnerį (juo gali būti internetinės parduotuvės atstovai) ir savo parašu patvirtinti sutartį dėl prekių įsigijimo išsimokėtinai.

Taigi įspraudžiame savo nagrinėjamą uždavinį šiuos rėmus – du verslo partneriai, tarp kurių vyksta duomenų apsikeitimo transakcijos internetinėje erdvėje. Labai patogu rinktis nemokamas, gerai sukurtas ir plačiai naudojamas programavimo aplinkas, pažangias technologijas. Internetinėje erdvėje tokios vienos populiariausių yra PHP (angl. PHP Hypertext Preprocessor) bei JavaServlet technologijos. Kiekviena jų skiriama skirtingų verslo partnerių informacinių sistemų realizacijai.

Atlikdami šį mokslinį taikymo pavyzdį naudojant visiškai nepriklausomas, bei kartu skirtingas informacines sistemas bandome įrodyti, jog mūsų kuriamas modelis yra nepriklausomas nuo realizacijos skirtingose komunikacinės kilpos aktorių sistemose. Svarbu yra susiderinti duomenų apsikeitimo protokolą bei duomenis perduoti bendrai suprantamu formatu – šiuo atveju puikiai tinka XML duomenų standartas, kadangi jis yra labai plačiai naudojamas internete ir yra labai patogus informaciniams mainams.

2.2.1 DAO – PHP objektas komunikacijai su duomenų bazėmis

Tarkime jau turime dvi ar daugiau informacinių sistemų, kurios sugeba bendrauti bendrai apibrėžtu protokolu ir yra paruoštos keistis informacija. Mums tereikia duomenis paruošti duomenų mainams. Jei informacinės sistemos operuoja įvairiais duomenimis, kaip taisyklė jie yra saugomi duomenų saugyklose, neretai duomenų bazių valdymo sistemose. Tad, norint duomenis paruošti perdavimui kitai informaciniai sistemai, reikia juos įvairiais pjūviais bei sąlygomis ištraukti iš duomenų bazių. Būtent šis poreikis mus paskatino sukurti patogų bei lankstų įrankį bendravimui su duomenų baze. Duomenų srautams perduoti naudojame XML standartą. Taigi iškyla klausimas kaip efektyviai išspręsti duomenų išrinkimą, XML dokumentų generavimą, išsiuntimą ir gautų dokumentų analizavimą, bei importavimą į duomenų bazę. Taip pat šioje komunikacinėje kilpoje labai svarbu užtikrinti duomenų patikimumą, bei saugumą, kadangi jie keliaus internetu.

Kadangi vieną iš dviejų mūsų taikymo uždavinyje dalyvaujančių sistemų, kuriame PHP programavimo kalba, nusprendėme šį įrankį sukurti šioje kalboje. Padarę analizę radome panašių produktų, kurie galėtų tenkinti mūsų keliamą tikslą, tačiau jie yra nepakankamai lankstūs ir todėl kyla problemų kuomet norime pritaikyti konkretiems uždavinių sprendimams, bei kas ypatingai svarbu – kyla nemažai problemų kuomet projekto veikimo eigoje keičiasi informacinės sistemos funkciniai reikalavimai ir reikia darbą su duomenimis reikia pritaikyti naujai iškilusiems poreikiams taikyti.

Tad PHP aplinkoje bendravimui su duomenų bazėmis siūlome naudoti mūsų sukurtą programinį objektą, kurį pavadinome DAO (Data Abstraction Object – Duomenų Abstrakcijos Objektas). Jo pirminė bei pagrindinė paskirtis yra jungtis prie duomenų bazių, naudojantis jo metodais generuoti užklausas ir jas vykdyti. Šis objektas užklausas gali sugeneruoti ir įvykdyti nuo paprasčiausių išrinkimo iki sudėtingų įterpimo, šalinimo, ar išrinkimo per kelias duomenų bazės lenteles. Taip pat galima vykdyti užklausas transakcijomis, jas patvirtinti ar atšaukti.

Taigi DAO išpildo ir poreikį išrinkus duomenis perduoti jį į XML dokumentą, kuris būtų siunčiamas kaip užklausimas ar gaunamas kaip atsakymas komunikuojant keliems verslo partneriams komunikacinėse kilpose.

Taikymo pavyzdyje naudojame DAO sukurtą jungimuisi prie Postgresql duomenų bazės. Ši duomenų bazė yra gan aukšto lygio, turi savyje tokius komponentus kaip vaizdai (angl. views), skaitliukai (angl. sequences), transakcijas, išorinius raktus (angl. foreign keys), funkcijas (angl. stored procedures), palaiko transakcijas (angl. transactions) ir kitas.

Bendravimas su duomenų baze vyksta SQL kalbos užklausomis, tad ir DAO gali būti naudojamas visoms duomenų bazėje esančioms galimybėms išnaudoti.

Šio objekto esmė yra tame, jog jis turi metodus jungimuisi prie duomenų bazės, SQL užklausų generavimui ir iš duomenų bazės gautų rezultatų apdorojimui. Metodai žinoma universalūs sąlyginai, tačiau prireikus kažko ypatingo, ko negalima gauti naudojant esamus metodus, visuomet galima pasirašyti vaikinę klasę, kuri paveldėtų DAO.

Kadangi SQL kalbos sintaksė yra labai panaši įvairiose duomenų bazėse, o DAO būtent ir turi priemones SQL užklausų generavimui, tai DAO gali jungtis prie visų duomenų bazių kurioms PHP turi palaikymo bibliotekas, be to galima Windows operacinės sistemos aplinkoje jungtis į ODBC (Open DataBase Connectivity – jungtis prie atvirų duomenų bazių) – taigi galima aptarnauti nuo paprasčiausių duomenų bazių kaip MySQL, Access iki aukštesnio lygio – PostgreSQL, Oracle, MSSQL ir k.t. Netgi jei duomenų bazės vienodai išpildo sugeneruotas SQL užklausas, tai visiškai pakanka aprašyti naujus jungimosi ir kreipimosi į duomenų bazę metodus.

Vykdant užklausas į duomenų bazę, juo labiau kai servisai yra pasiekiami internetu, labai svarbu užtikrinti saugumą, kad nebūtų galima įsibrauti per užklausas (angl. SQL injection). DAO savyje turi priemones, padedančias sutvarkyti arba nufiltruoti pavojingus užklausos fragmentus, kurie galėtų sukelti problemų. Generuojant užklausą iš užklausai paruoštų duomenų masyvų yra išvengiamos (angl. escape string) duomenyse esančios kabutės, kabliataškiai bei kiti specialūs SQL simboliai. Taip pat tikrinama lentelės laukų sintaksė, ORDER BY, GROUP BY sąlygos.

Kuomet visas bendravimas su duomenų baze vyksta per vieną jungtį – DAO, nekyla jokių problemų įjungiant užklausų analizės mechanizmus

Užklausos generuojamos metodams perduodant tokius parametrus: su kokia lentele/lentelėmis bus atliekami veiksmai, ką reikės išrinkti arba ką reikės įterpti, apribojančios sąlygos (angl. where) masyvo struktūra. Pastarąjį masyvą reikia kurti kiekvieno užklausos kvietimosi metu, tačiau tai yra tikrai nesudėtinga – o svarbiausia tai yra lankstu ir universalu, paprasta pritaikyti naujai iškilusių funkcinių reikalavimų išpildymui.

Iš duomenų bazės gautų duomenų masyvų galima generuoti XML. Tačiau jei norima duomenis griežčiau kontroliuoti, pravartu XML dokumentus generuoti naudojant kokią nors šablonų sistemą. Tai tikrai yra patogu ir vėlgi – paprasta pritaikyti naujiems sistemai keliamiems reikalavimams. Be abejo siunčiant XML reikia sugeneruoti iš XML struktūrą apibrėžiančias schemas, pvz. DTD arba XSD.

Trumpai apžvelgsime pagrindinius DAO klasės metodus. Šie metodai yra pritaikyti Postgresql duomenų bazei, tačiau tinka iškart arba juos pritaikius ir kitoms duomenų bazėms (metodų algoritmai yra pateikti priede Nr. 3):

- Sisteminiai jungimosi klasės metodai:
    - setLink(&$link) – parametras $link yra prisijungimo į duomenų bazę resursas. Perdavus jį šiam metodui jis yra priskiriamas krasės vidiniam kintamajam.
    - getLink() – šis metodas skirtas gauti klasei priskirtą prisijungimo į duoemnų bazę resursui. Jei jo nėra yra grąžinama tuščia NULL reikšmė.
    - lastError() – metodas, gražinantis paskutinią esamos sesijos metu įvykusią klaidą. Klaidos gali būti įvairios, pvz. kreipimasis į neegzistuojančią lentelę, integralumo pažeidimas įterpiant naujus įrašus ar trinant esančius, kokiam nors veiksmui atlikti teisių trūkumas ir panašiai.
    - lst($activate=true) – metodas padedantis programuotojui realizuoti bei ieškoti klaidų. Šis metodas tiesiog aktyvuoja požymį, jog prieš vykdant užklausą į duomenų bazę būtų išvedamas pranešimas, kurio turinys būtų pačios užklausos SQL kodas. Tai labai padeda ieškant klaidų.
- Skaitliukų (angl. sequence) metodai:
    - getSequenceVal($seq) – šis metodas grąžina esakančią skaitliuko (angl. sequence) reikšmę.
    - getNextSequenceVal($seq) – metodas grąžina sekančią skaitliuko (angl. sequence) reikšmę. Skaitliuko sekanti reikšmė nebūtinai yra vienetu didesnė už esamą. Žingsnis, mažiausia bei didžiausia reikšmės, taip pat pradinė reikšmė ir ciklišKumas yra nurodomi skaitliukokūrimo metu.
- Pagalbiniai metodai užklausų generavimui:
    - escapeInput(&$string, $striptags = true) – metodas paruošia tekstinį (angl. string) kintamąjį užklausos formavimui. Yra labai svarbu saugiai paruošti pavojingus simbolius kaip kabutės, kabliataškiai, specialūs simboliai, nes dėl jų gali tapti neteisinga pati užklausa, arba tuo gali pasinaudoti internetiniai programišiai (angl. hacker) – taip vadinamos SQL atakos (angl. SQL injection).
    - escapeSqlStr($string) – metodas labia panašus į escapeInput, tačiau Ėtring kintamąjį dar ir pridėdamas kabutes priekyje ir gale – tiap paruošdamas tektinę reikšmę.

- escapeBlob($string) – kadangi duomenų bazėje galima saugoti labia įvairių tipų kintamuosius, tad galima saugoti ir binarinę informaciją, pvz. paveiksliukus ir kitokius failus. Tam jog sėkmingai įterptume reikia failų turinį apdoroti, tam ir skirta ši funkcija.
- text_input($text, $do_what=array()) – vienas iš metodų naudojamas saugumui SQL užklausoje užtikrinti. Jis tiesiog patikrina tekstinio lauko reikšmę.
- extract_array_field($arr_field, $fill_keys=true) – metodas skirtas masyvo tipo kintamųjų iš duomenų bazės pavertimui PHP masyvu, kad programuotojai galėtų juos naudoti programavimo aplinkoje.

- Transakcijas valdantys metodai:
  - begin() – metodas skelbiantis transakcijos pradžią.
  - commit() – metodas patvirtinantis sėkmingą transakcijos pabaigą ir taip skelbiantis jog visi transakcijoje įvykdyti veiksmai turi būti patvirtinti. Jei transakcijoje įvyko klaida – jei vienas veiksmas nebus patvirtintas.
  - rollback() – metodas grąžinantis transakcijoje vykdytus veiksmus.
- Išrinkimo, įterpimo, šalinimo, atnaujinimo veiksmus vykdantys metodai:
  - query($q, $print = ") – tai metodas, kuris įvykdo sugeneruotą užklausą į duomenų bazę ir grąžina resursą į rezultatą. Resursas yra rodyklė į duomenų bazės atsakymą.
  - fetch_assoc($r) – $r parametras turi būti rodyklė į duomenų bazės užklausos atsakymą. Jei tai buvo atsakymas į išrinkimo funkciją tai šis metodas grąžina asociatyvų išrinktų laukų masyvą. Jei buvo išrinkta daug eilučių, ši metodą reikia kartoti tol kol atsakymo resursas nėra tuščias.
  - select($table, $what='*', $where='', $run=true) – metodas suformuojantis išrinkimo užklausą ir įvykdantis ją, jei yra $run parametro reikšmė yra teisinga (angl. true). Svarbu yra akcentuoti, jog šie argumentai užtikrina lankstų, patogų ir greitą programuotojo programavimą. $table arguentas – tai lentelė, vaizdas (angl. view) arba jie atskirti kableliais, iš kurių bus išrenkama infrmacija. $what – argumentas saugantis išrinkimo reikšmę, pvz. ‘*‘ jei norima išrinkti visą informaciją, arba tik konkretūs laukai jei nereikia išrinkti visų laukų. $where – apribojanti sąlyga, gali būti tekstinė reikšmė arba reikšmių masyvas. Jei tekstinė reikšmė – tai tiesiog ji panaudojama formuojant užklausą, jei masyvas – pirmiau kreipiamasi į metodą prepare_where, kuris iš masyvo sugeneruoja tekstinią

apribojimo sąlygą. Priklausomai nuo $run argumento reikšmės užklausa gali būti įvykdoma arba gražinta sugeneruota.

- get_first($table, $what='*', $where='') – šis metodas labai panašus į select, tačiau apriboja išrinktų rezultatų skaičių iki 1. Taip pat grąžina rezultatą jau kaip asociatyvų masyvą.
- get_all($table, $what='*', $where='') – šis metodas įvykdo išrinkimo užklausą kaip ir select, tačiau grąžina ne resursą, o visų rezultatų asociatyvų masyvą.
- insert($table, $values_array, $do_what=array('escape','striptags'), $run=true) – metodas vykdantis įterpimo užklausą. $table argimentas yra lentelės pavadinimas į kurį vykdomas įterpimas, $what – asociatyvus masyvas laukų kurie bus įterpiami ir jų reikšmės, $do_what yra asociatyvus masyvas reikšmių pagal kurias nustatoma, kokias paruošimo funkcijas reikai atlikti su kiekviena įterpiamų laukų reikšme prieš formuojant užklausas. Tai daroma saugumo sumetimais.
- update($table, $values_array, $where='', $do_what=array('escape','striptags'), $run=true) – update metodas vykdo atnaujinimo užklausą, ši užklausa skiriasi nuo insert užklausos tik tuo, jog turi $where sąlygos atributą, kuris nusako kurias eilutes lentelėje reikės pakeisti pagal Ėvalues_array neasociatyvių duomenų masyvo reikšmes.
- delete($table, $where='', $run=true) – šis metodas skirtas trynimo užklausos formavimui ir įvykdymui. argumentai nurodo tentelę, iš kurios bus trinama, bei where salygą, kurią atitinkantys įrašai ir bus pašalinti.
- prepare_where($where) – tia yra labia svarbus metodas. Deja dėl to jog tia yra konfidenciali informacija negalime pateikti šio metodo algoritmo, tačiau principas yra paprastas: šis metodas iš asociatyvaus laukų ir reikšmių masyvo sugeneruoja where sąlyga, kuri vėliau yra naudojama formuojant užklausos SQL. Labai svarbus šio metodo saugumas, todėl kiekvienas laukas ir lauko raktas yra patikrinami ir praleidžiami tik saugūs simboliai.

Iš duomenų bazės gautų duomenų masyvų galima generuoti XML. Tačiau jei norima duomenis griežčiau kontroliuoti, pravartu XML dokumentus generuoti naudojant kokią nors šablonų sistemą, pvz. „Smarty“. Tai tikrai yra patogu ir vėlgi – paprasta pritaikyti naujiems sistemai keliamiems reikalavimams. Be abejo siunčiant XML reikia sugeneruoti iš XML struktūrą apibrėžiančias schemas, pvz. DTD arba XSD.

Norėdami įrodyti mūsų sukurto objekto unikalumą ieškojome alternatyvių sprendimo variantų. Pavyko rasti panašius objektus, skirtus PHP aplinkai – tai PEAR klasių bibliotekoje esanti DB_DataObject.

PEAR – PHP praplėtimas ir Aplikacijos Saugykla (angl. PHP Extension and Application Repository).

- Struktūrizuota atvirojo kodo biblioteka.
- Sistema kodo pasidalinimui ir paketo atnaujinimui.
- PHP rašomo kodo standartas.
- PECL – PHP praplėtimų bendruomenės biblioteka (angl. PHP Extension Community Library).
- Internetinė svetainė, pašto konferencija parsisiuntimo serverių veidrodžiai PHP/PEAR PEAR bendruomenei.

PEAR yra bendruomenės vystomas projektas. Pradėtas kurkti Sting S.Bakken 1999 susilaukė nemažo populiarumo ir žmonių palaikymo.

PEAR kodo biblioteka yra suskaidyta į paketus. Kiekvienas paketas yra atskiras projektas, kurį vysto atskira kūrėjų komanda, kuris turi versijos numerį, dokumentaciją, apibrėžtas sąsajas su kitais paketais

[19] Tarp visų PEAR klasių yra ir panaši į DAO – DB_DataObject abstrakcijos sluoksnis. DB_DataObject atlieka dvi funkcijas:

- Generuoja SQL užklausas naudojant aprašytais kūrimo ir remiantis iš anksto apibrėžtais vidiniais objekto kintamaisiais.
- Veikia lyg duomenų saugykla duomenų lentelės eilutei, t.y. saugo paskutinės užklausos rezultatus kaip klasės kintamųjų masyvą.

Pagrindinė klasė yra sukurta paveldėjimui kiekvienai duomenų lentelei, kad galima būtų sudėlioti duomenų logiką (angl. data logic) duomenų klasės viduje.

Šių PHP objektų palyginimas

2 lentelė. Programavimo kokybės palyginimas nenaudojant objektų, naudojant DAO ir PEAR DB_DataObject

	Nenaudojant DB aptarnavimo objektų	**PEAR DB_DataObject**	**DAO**
Programinis priemonių atnaujinimas	nereikalingas	geras	vidutinis
Saugumo užtikrinimas	prastas	geras	geras
Programinio kodo apimtis	didelė	vidutinė	maža
Kintančių funkcinių reikalavimų realizacija	labai prastas	vidutinis	geras

Lentelėje Nr. 2 matome pateiktą programavimo kokybės palyginimą nenaudojant objektų, naudojant mūsų sukurtą ir siūlomą DAO bei PEAR bibliotekoje esantį DB_DataObject. Programavimo kokybę vertiname pagal šiuos pasirinktus kriterijus:

Programinis priemonių atnaujinimas. Jei nenaudojami kokie nors objektai bendravimui su duomenų bazėmis tai natūralu, jog nereikia jų ir atnaujinti. DB_DataObject programinis atnaujinimas yra labai geras, kadangi šį objektą kuria entuziastų komanda bei šis produktas yra atvirojo kodo (angl. free source). DAO programinį palaikymą vertiname gan kritiškai, tačiau ir ateityje jis bus vystomas bei tobulinamas, kadangi jau dabar DAO yra naudojamas bent dešimtyje internetinių projektų, jų funkciniai reikalavimai nuolat keičiasi bei atsiranda naujų.

Saugumo užtikrinimas. Nenaudojant tokių objektų tenka SQL užklausas rašyti ten kur tik prireikia atlikti kreipinius į duomenų bazę – užtikrinti saugumą taip yra tikrai sudėtinga, kadangi kiekvienas SQL užklausos fragmentas kiekvienoje situacijoje turi būti tikrinamas iš naujo. Visiškai kita situacija naudojant užklausas į duomenų bazę vykdančius objektus. Jie savo viduje generuodami SQL kiekvieną jiems perduotą ar priskirtą kintamąjį bei jo reikšmę gali patikrinti, pavojingus simbolius atmesti ar teksto fragmentus nukenksminti (angl. escape string). Tai yra labai patogus būdas bei apsisaugojimo metodas nuo įsibrovimo į duomenų bazes (angl. SQL injection).

Programinio kodo apimtis. Programuojant reikia siekti netik algoritmo teisingumo, išbaigtumo ir patikimumo, tačiau kartu ir aiškumo. Tai labai svarbu programuojant vidutinio dydžio ir didelius projektus, kurie ra ne vienadieniai, o kuriami ilgalaikiam veikimui. Visuomet projekto veikimo metu iškyla nauju funkcinių reikalavimų ar tenka senuosius tiesiog suderinti su pasikeitusiais poreikiais. Programinio kodo aiškumas, paprastumas, dokumentavimas yra labai svarbūs požymiai, kurie padeda programuotojui greičiau išsiaiškinti problemos esmę, greičiau atlikti norimus pakeitimus. Kodo apimtis yra vienas iš kriterijų kad pakeitimai būtų atliekami sparčiai. Nenaudojant objektų tenka atlikti labai daug bereikalingų veiksmų, kurie labai stipriai išplečia programinio kodo apimtis ir labai apsunkina naujų reikalavimų realizavimą, taip pat padidina klaidų tikimybę. Remiantis kelerių metų programavimo patirtimi galime teigti, jog kodą be tokių objektų perrašant ir naudojant DAO – kodo apimtis sumažėja net nuo 2 iki 3 kartų. DB_DataObject klasė taip pat įneša daug aiškumo, tačiau dideliu programinio kodo taupumu nepasižymi, kadangi kiekvieną duomenų lentelės lauką reikia aprašyti kaip klasės vidinį kintamąjį, iškyla keblumų kuomet norime atlikti veiksmus su keliomis lentelėmis vienu metu – reikia papildomai programuoti norint atlikti sudėtingesnes užklausas į duomenų bazę.

Kintančių funkcinių reikalavimų realizacija. Nenaudojant objektų tiems patiems algoritmams realizuoti yra prirašoma žymiai didesni kiekiai programinio kodo. Akivaizdu, jog didžiuliame kodo kiekyje norit atlikti kad ir menkiausius pakeitimus reikia daug daugiau laiko sugaišti aiškinimuisi, analizei, naujo funkcinio reikalavimo realizavimui nei, kad naudojant objektus. DB_DataObject objektas nėra pats geriausias pasirinkimas jei reikia nuolat tobulinti informacinę sistemą, kadangi jo nors ir aiškus, tačiau gan griežtas metodų aprašymas sukelia nemažai keblumų kuomet reikia keisti duomenų srautus. Tačiau DAO būtent neapibrėžia konkrečių duomenų srautų sudėties, duomenų tipų. Jei pasikeičia duomenų srautas bei duomenų bazės lentelių struktūra – DAO užklausų generavimo variklis tai vykdo kaip eilinį užklausos generavimą iš šių, kad ir naujai suformuotų duomenų masyvų. Neretai projektuose pakanka pakeisti duomenų bazės schemos elementus bei kliento dalį sudarančius šablonus (internetiniuose projektuose tai yra HTML dokumentų ar jų fragmentų šablonai), tiksliau duomenų saugojimo bei įvedimo laukus (angl. input), pasirinkimo sąrašus (angl. select), teksto įvedimo laukus (angl. textarea), kad tokie primityvūs nauji funkciniai reikalavimai būtų išpildomi netgi neatliekant jokių pakeitimų PHP algoritmo failuose.

Apibendrinant palyginamąjį tyrimą galime daryti išvadą, jog DAO turi tvirtų pranašumų prieš PEAR bibliotekose esantį DB_DataObject objektą, bei DOA naudojimą yra daug pranašesnis prieš tiesiog jungimosi į duomenų bazes programavimą kiekvienoje vietoje prireikus atlikti veiksmus duomenų bazėje. DAO ypač tinka realizuojant nuolat kintančius funkcinius reikalavimus – padeda taupyti sąnaudas bei laiką.

2.2.2 XML dokumentų parengimas panaudojant Java Servlet priemones

Kaip alternatyvą PHP siūlome ir Java Servlet technologiją – tai Sun Microsystems sukurta technologija dinaminių puslapių generavimui. Java Servlet programuojamas Java programavimo kalba. Panašiai kaip ir PHP tai yra serveryje esantis komponentas, susietas su užklausos adresu, generuojantis atsakymus. Naudodami trijų lygių architektūrą (angl. *three tier architecture*) (duomenų bazės lygis – serverio apdorojančio užklausas lygis – atvaizdavimo lygis), galime užtikrinti apsikeitimą duomenimis bei įvairias operacijas su jais. Viduriniame lygyje apdorodami duomenis, galime juos paruošti saugojimui į duomenų bazę ar siuntimui.

XML duomenų saugojimui duomenų bazėje galime naudoti jau anksčiau aprašytus DAO principus (realizuotus Java kalba), o jų apdorojimui Java įrankius darbui su XML, pvz. Sun Microsystems sukurtą klasių rinkinį JAXP (angl. Java API for XML Processing)[5]. Šis ir panašūs įrankiai skirti XML dokumentų apdorojimui ir kūrimui, t.y. duomenų srautų apdorojimui. Naudodami juos kartu su DAO, turime įrankį, kuris užtikrina duomenų integralumą ir pilnumą keičiantis funkciniams reikalavimams [21].

3 IŠPERKAMOSIOS NUOMOS SUTARTIES SUDARYMO E-PARDUOTUVĖJE TAIKYMO PAVYZDYS

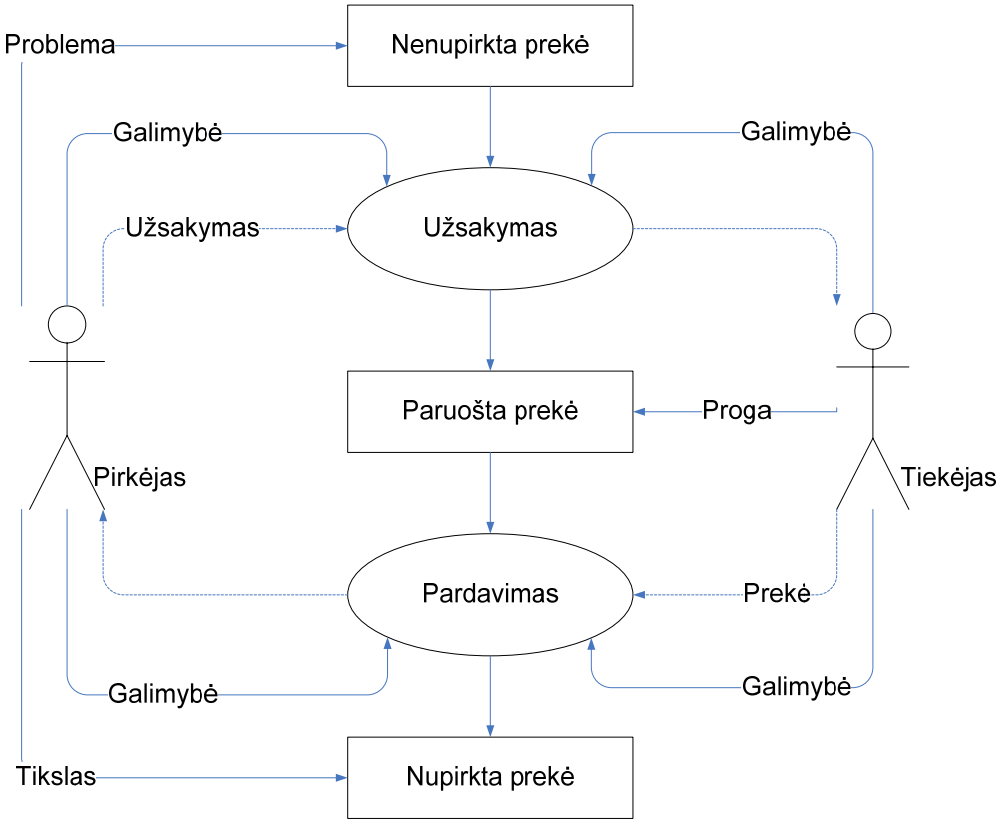
3.1 Dalykines srities apibrėžimas

Sparčiai besuvystant internetinėms technologijoms, didėja elektroninių paslaugų spektras. Darbe nagrinėjamam modeliui realizuoti, buvo atliktas taikymo pavyzdys. Pasirinkta dalykinė sritis – e-parduotuvė ir joje esančių prekių pirkimas išsimokėtinai sudarant sutartį su lizingo kompanija. E-parduotuvė – tai paprastos prekių parduotuvės analogas internetinėje erdvėje. Galima pasirinkti norimas prekes, jų kiekius ir pan.. Už prekes yra atsiskaitoma įvairiais būdais .Standartiniu tokioms parduotuvėms būdu – kreditinėmis kortelėmis, banko pavedimu ir pan. Šiame taikymo pavyzdyje plėtojamas būdas atsiskaityti per lizingo kompaniją, sudarant su ja išperkamosios nuomos sutartį. Tai kol kas analogų neturintis atsiskaitymo būdas Lietuvoje. Taikymo pavyzdyje didžiausias dėmesys kreipiamas į išplėstinėmis komunikacinėmis kilpomis specifikuotų verslo transakcijų realizacija trijų lygių architektūroje, duomenims apibrėžti naudojant XML standartą.

Kaip jau minėta, tai yra precedentinis taikymo pavyzdys Lietuvoje, tačiau kol kas jis yra plėtojamas tik techniniame lygyje, nes trūksta juridinių pagrindų tokiam taikymo pavyzdžiui realiai gyvuoti. Kadangi lizingo kompanija, kaip kreditorius turi patikrinti kliento mokumą, ji gali kreiptis į atitinkamas institucijas (socialinio draudimo duomenų bazę, skolininkų administravimo informacinę sistemą ir pan.). Bet tam reikalingas vienareikšmiškas kliento sutikimas dėl šių duomenų tvarkymo. Kol kas nėra realiai veikiančių elektroninio parašo nuostatų, todėl tokio sutikimo dokumentas elektroninėje erdvėje kol kas yra neįmanomas.

3.2 Kilpų sudarymas

3.2.1 Informacijos apie produkciją atnaujinimas e-parduotuvėje

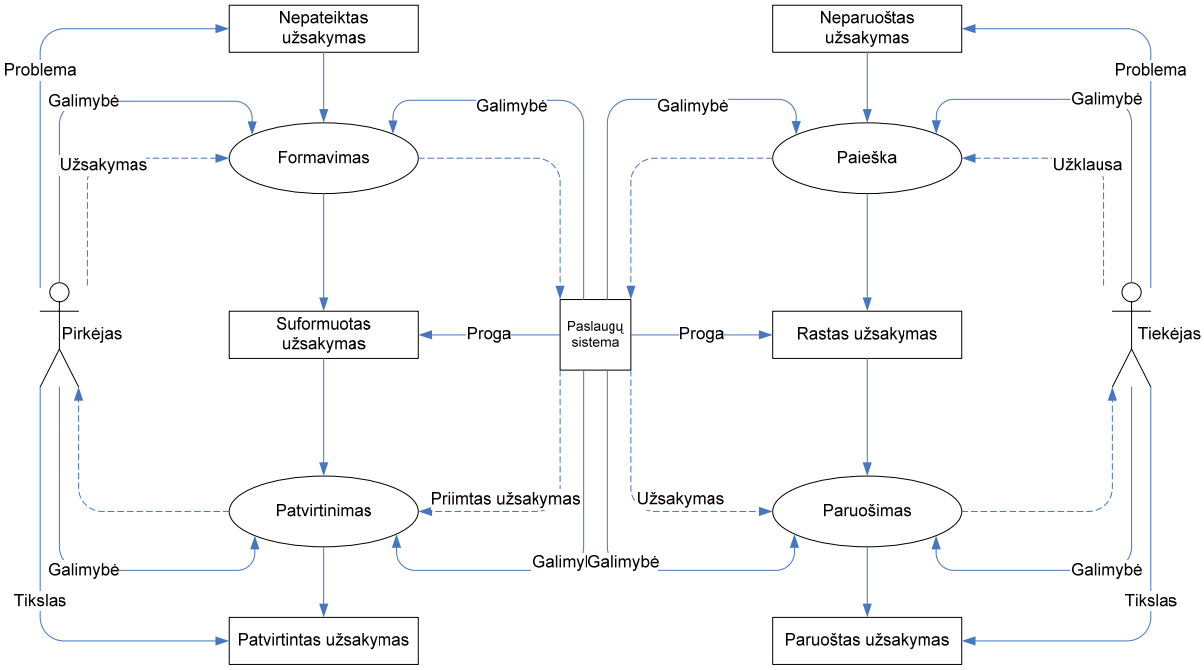


3 pav. Pirkėjo-tiekėjo komunikacinė kilpa

3 paveiksle pateikiama tipinė elementari komunikacinė kilpa, kurioje aktorius-iniciatorius *Pirkėjas* veiksmais (inicijuotu procesu) *Užsakymas* siunčia srautą *Užsakymas* kitam aktoriui *Tiekėjas*. *Tiekėjas* reakcijos veiksmais (reakcijos procesu) *Pardavimas* grąžina srautą *Prekė*. Inicijuotas procesas *Užsakymas* užsakymo objektą iš būsenos *Nenupirkta prekė* perveda į būseną *Paruošta prekė*, o reakcijos procesas *Pardavimas* užsakymo objektą iš būsenos *Paruošta prekė* perveda į būseną *nupirkta prekė*. Dviejų aktorių - *Pirkėjas* ir *tiekėjas* - komunikacinė kilpa yra pragmatiškai motyvuota, jei ji tenkina procesų gyvybingumo kriterijų. Pagal pragmatinių priklausomybių apibrėžimą šis kriterijus gali būti formuluojamas taip: elementari komunikacinė kilpa **"**iniciatorius: *Pirkėjas* → inicijuotas procesas: *Užsakymas* → recipientas**:** *tiekėjas* → reakcijos procesas: *Pardavimas* → iniciatorius**:**

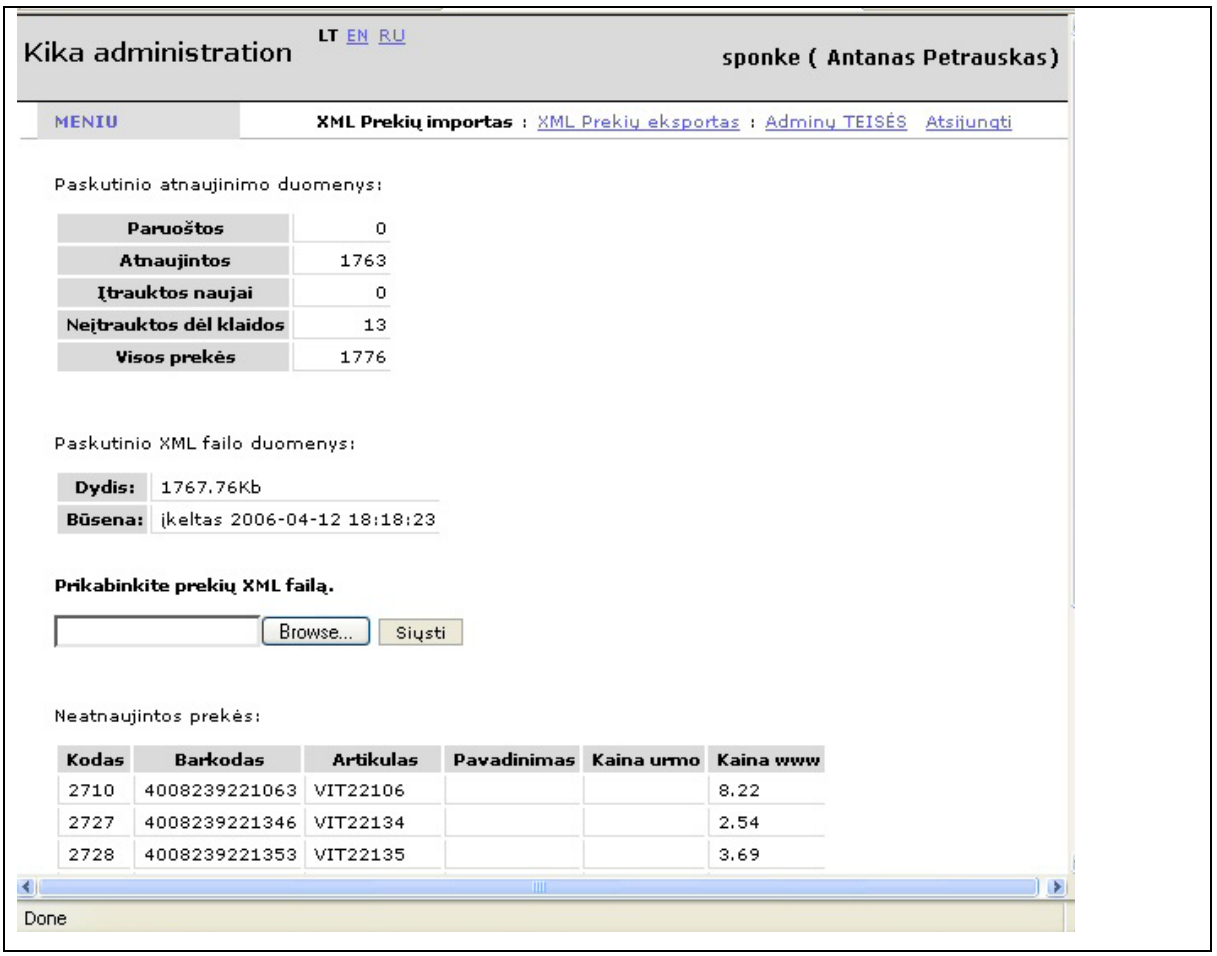
Pirkėjas" bus gyvybinga, jei gavėjas turi progą reakcijos procesu veiklos objekto tarpinę būseną *Paruošta prekė* pervesti į galinę būseną *Parduota prekė* ir tokiu būdu savo atsakomaisiais veiksmais teigiamai veikia iniciatoriaus problemos *Nenupirkta prekė* išpildymą ir iniciatoriaus tikslo *Parduota prekė* pasiekimą. Abu šie aktoriai - iniciatorius *Pirkėjas* ir gavėjas *Tiekėjas*, vienoje komunikacinėje kilpoje yra susieti bendrais procesais *Užsakymas* ir *Pardavimas*, turi galimybes šiuos procesus įvykdyti

.



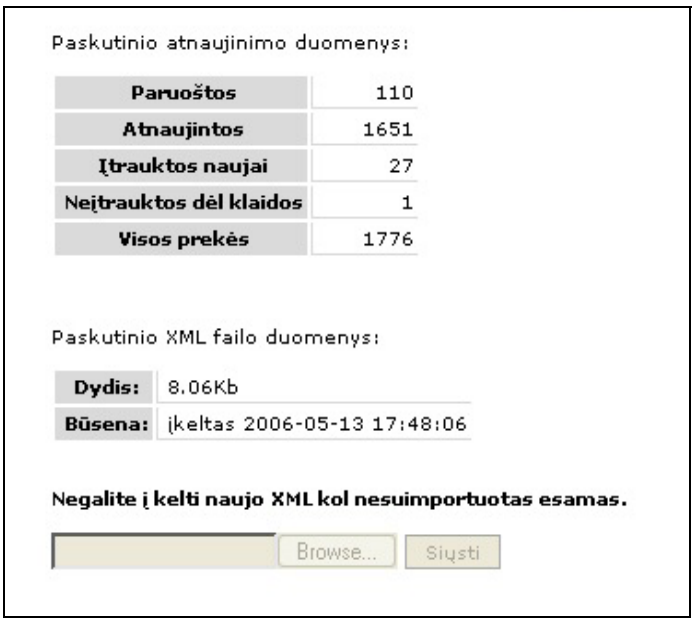
4 pav. Pirkėjo-tiekėjo išplėstinė komunikacinė kilpa

Paveiksle Nr. 4 yra pateikta išplėstinė komunikacinė kilpa, kurioje tarp aktorių Pirkėjas ir Tiekėjas atsiranda informacinė sistema Paslaugų sistema. Ši sistema turi progą įtakoti būsenas Suformuotas užsakymas bei Rastas užsakymas. Ši sistema atlieka tarpininko vaidmenį vykdant visos komunikacinės kilpos procesus. Informacinė sistema savyje išpildo Užsakymo formavimo, užsakymo paieškos, užsakymo paruošimo bei priimto užsakymo patvirtinimo procesus. Matome, jog ši komunikacinė kilpa yra pragmatiškai motyvuota, kadangi aktorius Pirkėjas turi tikslą pateiktą užsakymą suformuoti bei tikėtis jog jis bus įvykdytas. Aktorius Tiekėjas taip pat turi tikslą Pažymėti informacinėje sistemoje jau supakuotus pardavimui užsakymus pažymėti kaip paruoštus užsakymus. Pagal šį požymį vėliau gali būti informuojami klientai apie jau paruoštus atsiėmimui užsakymus.



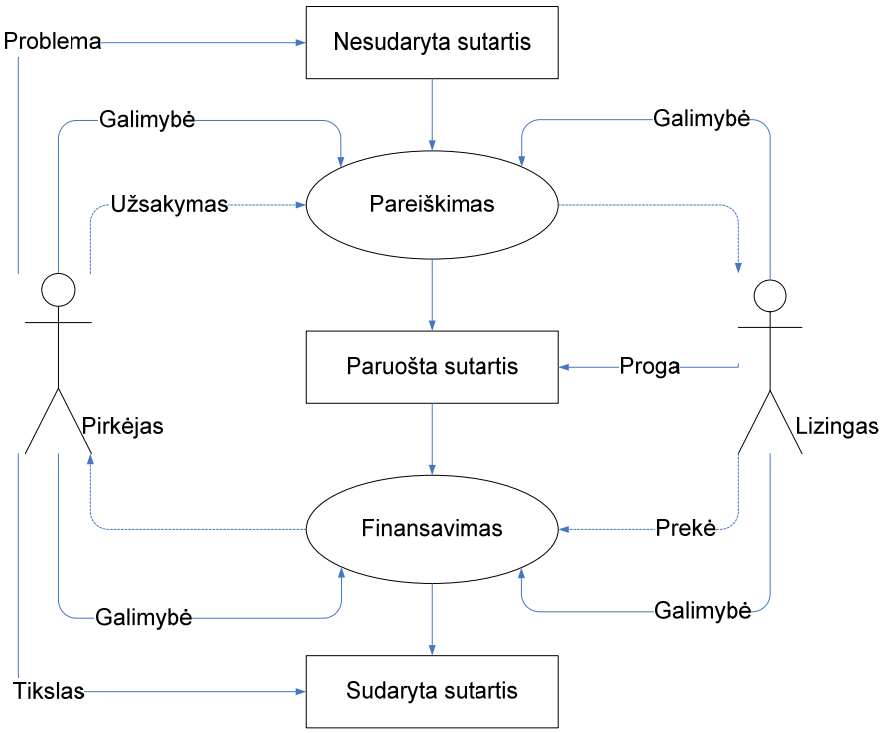
5 pav. Prekių duomenų atnaujinimo vartotojo sąsajos vaizdas

Paveiksle Nr.5 pateiktas administratoriaus sąsajos vaizdas, kuriame yra galimybė prisekti paruoštą prekių atnaujinimui duomenų XML failą. Pirmiausiai duomenys iš XML yra sukeliami į laikinųjų prekių lentelę, šiuo būdu nustatant kurias būtent prekes reikės atnaujinti, kurios prekės yra importuojamos pirmą kartą, tad reikės sukelti visus duomenis (netik kainą, ir likučius sandeliuose). Nusiuntęs šį failą administratorius turi galimybę stebėti importavimo eigą bei rezultatus: kiek prekių dar liko paruoštų įkėlimui, keik prekių jau atnaujinta, kiek prekių įtraukta naujai, kiek prekių neįtrauktos dėl duomenų trūkumo ar kitokių klaidų, kiek duomenų bazėje yra iš viso skirtingų prekių. Tai matome paveiksle Nr.6.

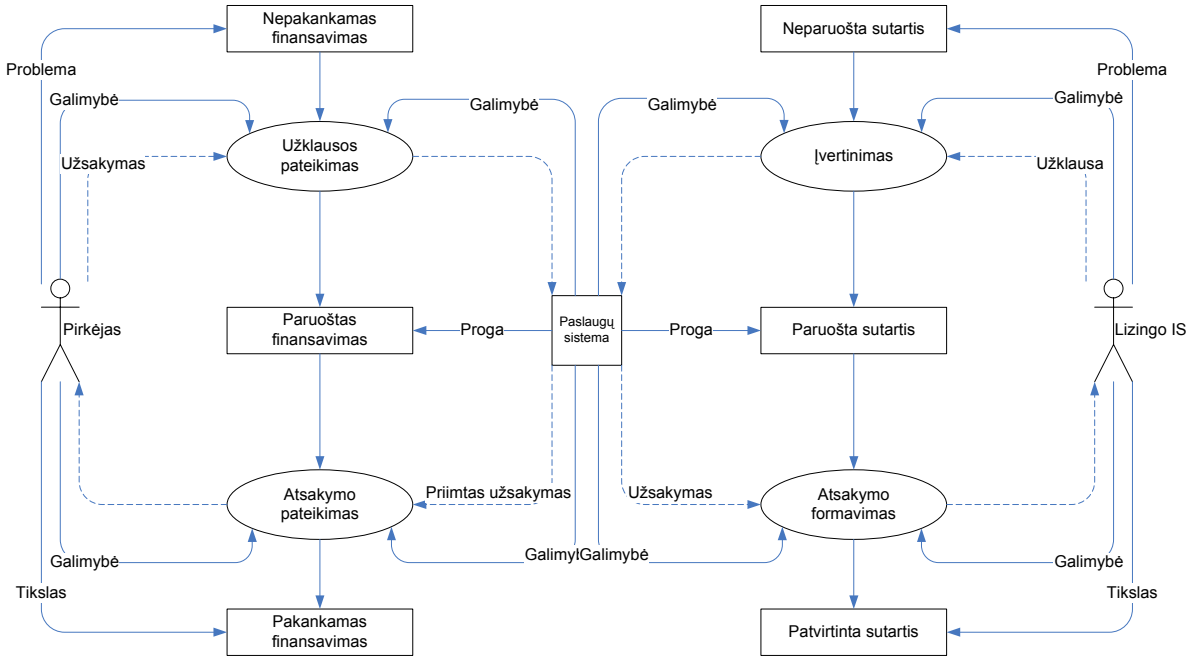


6 pav. Prekių duomenų atnaujinimo vaizdas XML importavimo metu

3.2.2 Išperkamosios nuomos sutarties užsakymas



7 pav. Pirkėjo-lizingo komunikacinė kilpa



8 pav. Pirkėjo-lizingo išplėstinė komunikacinė kilpa

9 pav. Pavyzdinio taikymo bandomoji sąsaja

Paveiksle Nr.9 pavaizduota pavyzdinė anketa, kurią užpildęs pirkėjas gali sužinoti savo galimybes gauti paskolą iš lizingo bendrovės nurodytai sumai, bei pasirinktam išsimokėjimo terminui. Šiuo atveju paspaudus „Tvirtinti“ mygtuką duomenys yra siunčiami į serverį, kuriame yra suformuojamas ir pasirašomas privačiu SSL raktu XML dokumentas, kuris yra siunčiamas Lizingo bendrovės informacinei sistemai. Pastaroji viešuoju sutiktina ar XML atitinka parašą, jei taip tuomet kreipiasi į SoDrą išrinkdama kliento mokumo informaciją. Jei viskas tvarkoje klientui siunčiamas teigiamas atsakymas, kitu atveju neigiamas. Paveiksle Nr. 10 vaizduojamas sugeneruotas bandomųjų XML duomenų paketas, parašas, užtikrinantis duomenų patikimumą, bei gautas sėkmingas XML atsakymas iš lizingo įmonės IS.

10 pav. Pavyzdinis XML duomenų failas, SSL parašas, gautas rezultatas iš nutolusios IS

Paruoštas XML:

```
<?xml version="1.0" encoding="utf-8"?>
<doc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.fotofabrikas.lt/kika/admin/data/schemas.xsd/xml_outgoing.xsd">
<uzklausa id="0">
```

```
<vardas>Vardenis</vardas>
<pavarde>Pavardenis</pavarde>
<asmens_kodas>38206040000</asmens_kodas>
<paso_numeris>LR00000</paso_numeris>
<galiojimo_data>2015-01-01</galiojimo_data>
<gatve>Studentų</gatve>
<namas>03</namas>
<butas>124</butas>
<miestas>Kaunas</miestas>
<pasto_kodas>LT-40004</pasto_kodas>
<trukme>18</trukme>
<suma>3000</suma>
<pradinis_inasas>1200</pradinis_inasas>
</uzklausa>
</doc>
```

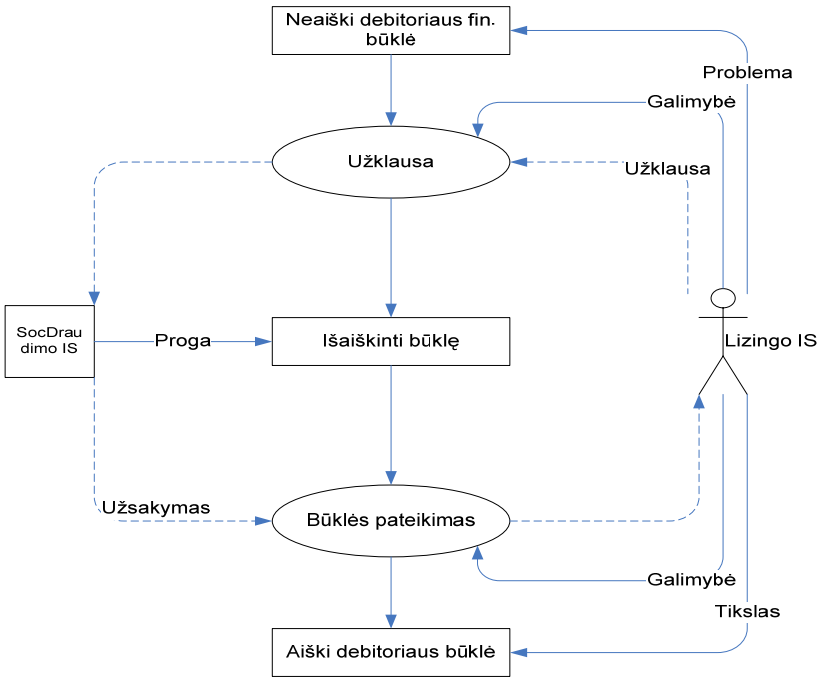
XML identifikavimo parašas:

```
ibgtXZ3HkLfUPC56Lmj4m9f/rpWyvQ5GAU/E3Lwsu4iTpTuC9/Kw5M7bdxcD9fbYqzYaqg2ZW
0Jrp5Z4qe/bFzl7pEFT5x1NLrLsHHEOgTQHoR7AzfyHYuLLcvzJscPptrbhW01duCFnlXCbsrguRDUrJ0lW329oUSkZu
v0xmi0=
```

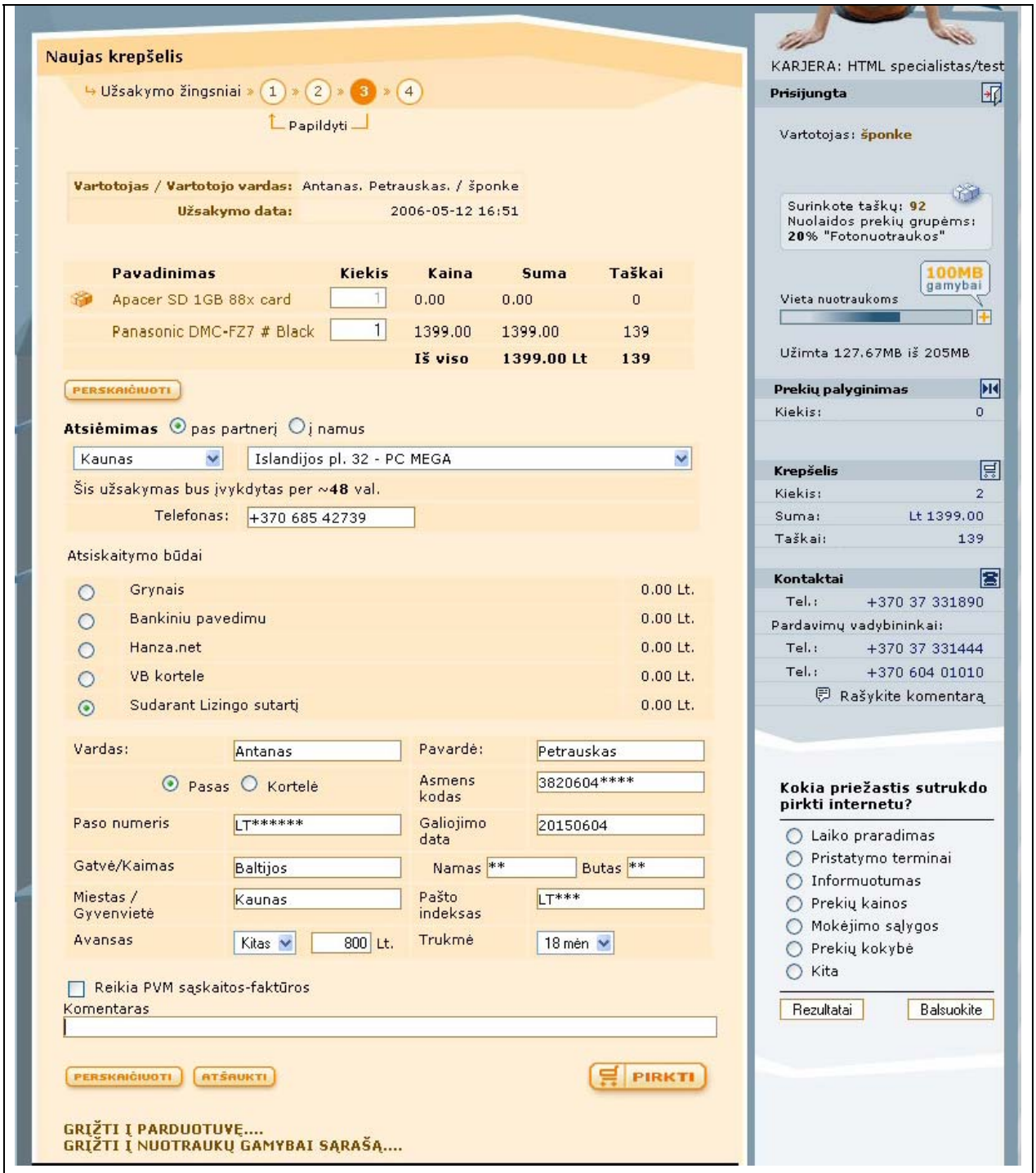
Gautas atsakymas:

```
<?xml version="1.0" encoding="utf-8"?>
<doc xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.fotofabrikas.lt/kika/admin/data/schemas.xsd/xml_in
comming.xsd">
<uzklausa id="0">
<atsakymas>1</atsakymas>
</uzklausa>
</doc>
```

3.2.3 Debitorių finansinės būklės nustatymas



11 pav. Sodros-lizingo IS komunikacinė kilpa



12 pav. Užsakymo formavimo vaizdas iš realiai veikiančios e-parduotuvės

Paveiksle Nr.12 yra pateiktas vartotojo sąsajos vaizdas, matomas realiai veikiančioje www.fotofabrikas.lt elektroninėje parduotuvėje. Klientai suformavę savo pirkimų krepšelį išsirenka atsiėmimo vietą, apmokėjimo būdus. Vienas iš jų yra apmokėjimas sudarant išperkamosios nuomos sutartį. Verta paminėti, jog šiuo metu nėra įmanomas vartotojo mokumo patikrinimas lizingo kompanijų informacinėse sistemose dėl galiojančios teisinės bazės. Tačiau kadangi vis stiprėja toks poreikis, tai galima tikėtis, jog ilgai netrukus turėsim netik teorinę, tačiau ir praktinę galimybę sudaryti sutartį internetu ir patvirtinti ją užsakovo elektroniniu parašu.

3.3 Saugumo užtikrinimas OpenSSL priemonėmis

3.3.1 SSL raktai

Kuomet apsikeitimai duomenimis vyksta nesaugioje internetinėje erdvėje, iškyla didžiulis pavojus, jog atsiras piktavalių, kurie mėgins patikrinti informacines sistemas turėdami negerų kėslų. O galimybių tam tikrai yra. Be abejo yra sprendimas keliems verslo partneriams susijungti į bendra tinklą, kuris būtų nepasiekiamas iš viešos erdvės. Tačiau neretai tai reikalauja ypatingai didelių investicijų. Tenka ieškoti mažiau lėšų reikalaujančios išeities.

OpenSSL ssl bibliotekoje įgyvendinami SSL v2/v3 (angl. Secure Sockets Layer) ir TLS v1 (angl. Transport Layer Security) protokolai. OpenSSL komandinės eilutės įrankis naudojamas iškviesti OpenSSL bibliotekos metodus. Jie gali būti naudojami [20]:

- RSA, DH ir DSA raktų parametramų kūrimui
- X.509 sertifikatų, CSR ir CRLs kūrimui
- Užkodavimui bei atkodavimui su kodavimo algoritmais
- SSL/TLS kliento bei serverio komunikavimo patikrinimams
- S/MIME pasirašytų ar užkoduotų elektroninių laiškų apdorojimui

3.3.2 Raktų generavimas

Bendru atveju, raktų sukūrimo procedūra atrodo taip:

1. Klientas sukuria savo viešojo ir slaptojo rakto porą, be to suformuojama sertifikato užklausa (certificate request), į kurios sudėtį įeina viešasis raktas (Komerciniai web serveriai dažnai turi priemones raktams generuoti).
2. Klientas nusiunčia sertifikato užklausą mums.
3. Mes pasirašome jų užklausą savo raktu ir suformuojame sertifikatą

Viena pusė siųsdama pranešimą kitai, pasirašo jį su slaptuoju raktu, o kita pusė, turėdama atitinkamą viešąjį raktą (pusę poros), gali patikrinti, ar pranešimas nėra pakeistas. Autorizacijos centrai nenaudojami, mes patys esame sertifikavimo organizacija (certificate authority).

Sertifikatui gauti reikalinga kliento sertifikato užklausa, su šiais duomenim:

C=LT (šalis)
L=<Vietove> (pvz. Kaunas)
O=<Organizacija> (Organizacija)
OU=BankLink
CN=<serverio vardas> (pvz. banklink-server.organizacija.lt>
E=<kontaktinis e-mail>

Raktų generavimui mes siūlome naudoti OpenSSL(http://www.openssl.org), kuris yra pakankamai dažnai atnaujinamas nemokamas kriptografijos įrankių rinkinys.

Sukompiliuota OpenSSL versiją galima gauti iš http://www.modssl.org/contrib/, tačiau naudoti ją siūlome tik tada, jei jūs pasitikite minėtuoju interneto puslapiu.

Rakto ir sertifikato užklausos generavimas (OpenSSL):

openssl req -new -config <Kelias iki openssl.cnf> -out cert_request.pem -keyout private_key.pem

- Jūsų paklaus slaptažodžio slaptajam raktui.
- Jūsų paklaus duomenų, sertifikato užklausai.

Peržiūrėti sertifikato užklausą galite:

openssl req -in cert_request.pem -text -noout

Išsaugoti privatų raktą nešifruota forma galite:

openssl rsa -in private_key.pem -out new_private_key.pem

Pakeisti privataus rakto šifravimo slaptažodi galite:

openssl rsa -in private_key.pem -3des -out new_private_key.pem

Peržiūrėti SSL sertifikatą galite:

openssl x509 -in cert.pem -text -noout

Pavyzdys, kaip sugeneruoti sertifikato užklausą, parodytas http://www.modssl.org/docs/2.8/ssl_faq.html#ToC28

3.3.3 Raktų ir “Certificate Request” pavyzdžiai

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQCvqlkOMrYxGw85j7rkF+KNChajMJL+o6pwGK2bt1IRP/VQxWyB
4AqrljzcUDn1wWTmEzISEJqxF9t/WN8jWcdFCO1Zws5inYt0+WZqkZW+MXVCf+8g
sn8Tbdt7C69++ZH6MUqjJj+Q5EjauchXDmPWPg2v07HBZBIvAhTFDVE1ZwIDAQAB
AoGAMj3OarkmUrUijZwGH+aU7THNd68U26+Dt7vXK4oq9rQMPaW5ewvRAXJexcRM
T3WYnhUvZOtK0wu3w5xvdXTgAOz+Y7ZIIziQCut0+Uy4QERLnKGXZ4830DDh+M7m
oXeF3e+Eur4vx6Whg38jbZs+WyaRJqB2DgNnHhtakyZb/IECQQDkr2KKE6GSJT8h
FXih5Ux2boiN85ehZmN9Maprxviz3V90leyINL6dXapmBDBXClGV2QCDDOMqMEcw
nfqI31HJAkEAxKXBLrvveJ3G5YnOL0IhlGdpBURJtr7h3Pd6E9exdLJ0NBesaour
oj7eJ0+5CSHtCz9ST7S7WgtnguVGaERlrwJBANe8ZRehgXEf1jhdyxnf+YxYplZU
ER3gO8cljYJziLgWBTE1JpkwOQq4DEKFfRz/zQuyVftLsxpfxSkmP+3Tz5ECQQCs
f75rkX5qrvS3i9/rQraUKPZOIW4MOXuFyy0yVMYc2SHoRFABko23oDBeCagGI9V9
RUmeE6s5PMHLZ/YQJtePAkEAn/0DCozbYbz+xlGwhg7kHvH0LqCB6sITxfzMaFQb
T8EGmloj4wIkgvEImgBTyJ3xIZulcDwFpT9qApIKmcCm3Q==
-----END RSA PRIVATE KEY-----
```

Tas pat slaptasis RSA raktas, užšifruotas slaptažodžiu "kala" (be kabučių):

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,E6C541BC0138209A

UM49jAy/j7b1JTcuaWJzseIQBk+P7ZOWI6Wp9myGiDk+0+AVj4klNaNOWiH1L3kT
Fma3rOuKGF80ntoPNWcjEcLgK4Zjte026XXcr2w/yM1E8yVlmjVcrLAjR1l9bfe1
rKigDJVEsyUjGDI61rDcfEAbCSO/E5nl4P0M139PeV4paoG03VnvALNQcmI30gHW
BhDK9T2kpwG2rz1ATTf30e7GzpMWvpd7ONMV+39flHAMg7IgRy3GBbERjrwsalxr
dCMdhYyGvrnrBDQDmKuV95jZJu8siTeFDKlFINRS7DsLmhYXCIj50wOUizPLKvxV
dFvAZLiEVXWJ8UoZMn1+QYPNrEty1ox/0rFVM4o1rxK6zSyDoMAoNH6ynoWlLGDx
+nokFgobIWXvKZ2zHhm4+TUbF1cavOXX45yZ837/7udgDnQQ4P02vqpgI0w3oVGG
+5S4gU4ZCjUaax3L3UDAgYRagPY9kJGYcSiJ3399j6gUgcin7ebFAAhfKsi6ioRQ
GadiqhlACMKs30H2oxXJ+4rjNIP6zKlHvkYb3HFrpKOGdArN9KEX7qrOkWDpQaLS
pNlTjR1rKbzsLoqk9cbnRINf+8/Y0BHtnmhISmR0AM4uKYiAfcoCtwwfXMWG5oFT
OMF9oVCGLXFlNpPA60PpLvfYEwZuSEuLr2kJAMOfkSeJyDbInbztwKERRaB2LikC
LxGvpBBphCf48fR8EIQSN9VH7fSLLd3DohkLi5XupUZxxp58TG2WyqBz0isQPfp0
wGvz5ATJ5xtZQ6thrIZlayKV40zUiDSXu9N6ugRG/YJLCncaw8xrWg==
-----END RSA PRIVATE KEY-----
```

Sertifikato užklausa, PEM formate:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIB+TCCAWICAQAwgY8xCzAJBgNVBAYTAkxUMRAwDgYDVQQHEwdWaWxuaXVzMRww
GgYDVQQKExNBQiBiYW5rYXMgSGFuc2EtTFRCMREwDwYDVQQLEwhCYW5rTGluazEa
MBgGA1UEAxMRYmFua2xpbmsuaGFuc2EubHQxITAfBgkqhkiG9w0BCQEWEndlYm1h
c3RlckBoYW5zYS5sdDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAr6pZDjK2
MRsPOY+65BfijQoWozCS/qOqcBitm7dSET/1UMVsgeAKq5Y83FA59cFk5hMyEhCa
sRfbf1jfI1nHRQjtWcLOYp2LdPlmapGVvjF1Qn/vILJ/E23bewuvfvmR+jFKoyY/
kORI2rnIVw5j1j4Nr9OxwWQSLwIUxQ1RNWcCAwEAAaApMCcGCSqGSIb3DQEJDjEa
MBgwCQYDVR0TBAIwADALBgNVHQ8EBAMCBPAwDQYJKoZIhvcNAQEEBQADgYEALdf7
7J/GLfGZE27vXRLJrqAvjZeeV3/QlJMQTjhFihVb8N9h6Z+s7e+VbgEpKq5feVot
gbN8oPPbSYC0TVPOdRpKIi7+IzfVWWFgSZxUGKqdkypaGcRM4qa+wgPFsUi6syhZ
c5gJDvcjp+lEUvOett2/KT7KLFiQs9o/glY8PEI=
-----END CERTIFICATE REQUEST-----
```

Ta pati sertifikato užklausa, tekstiniam formate:

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=LT, L=Vilnius, O=AB bankas Hansa-LTB, OU=BankLink,
CN=banklink.hansa.lt/Email=webmaster@hansa.lt
        Subject Public Key Info:
```

```
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:af:aa:59:0e:32:b6:31:1b:0f:39:8f:ba:e4:17:
                    e2:8d:0a:16:a3:30:92:fe:a3:aa:70:18:ad:9b:b7:
                    52:11:3f:f5:50:c5:6c:81:e0:0a:ab:96:3c:dc:50:
                    39:f5:c1:64:e6:13:32:12:10:9a:b1:17:db:7f:58:
                    df:23:59:c7:45:08:ed:59:c2:ce:62:9d:8b:74:f9:
                    66:6a:91:95:be:31:75:42:7f:ef:20:b2:7f:13:6d:
                    db:7b:0b:af:7e:f9:91:fa:31:4a:a3:26:3f:90:e4:
                    48:da:b9:c8:57:0e:63:d6:3e:0d:af:d3:b1:c1:64:
                    12:2f:02:14:c5:0d:51:35:67
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment, Data
Encipherment
    Signature Algorithm: md5WithRSAEncryption
        2d:d7:fb:ec:9f:c6:2d:f1:99:13:6e:ef:5d:12:c9:ae:a0:2f:
        8d:97:9e:57:7f:d0:94:93:10:4e:38:45:8a:15:5b:f0:df:61:
        e9:9f:ac:ed:ef:95:6e:01:29:2a:ae:5f:79:5a:2d:81:b3:7c:
        a0:f3:db:49:80:b4:4d:53:ce:75:1a:4a:22:2e:fe:23:37:d5:
        59:61:60:49:9c:54:18:aa:9d:93:2a:5a:19:c4:4c:e2:a6:be:
        c2:03:c5:b1:48:ba:b3:28:59:73:98:09:0e:f7:23:a7:e9:44:
        52:f3:9e:b6:dd:bf:29:3e:ca:2c:58:90:b3:da:3f:82:56:3c:
        3c:42
```

IŠVADOS

Norint garantuoti patikimą verslo informacinių sistemų funkcionavimą, jas projektuojant būtina formaliai aprašyti verslo transakcijas. Tokiam aprašymui siūlomi nemažai metodų, komercinių produktų, standartų ir pan., kurie buvo apžvelgti. Šiame darbe buvo gilinamasi į išplėstines komunikacines kilpas verslo transakcijoms modeliuoti. Naudojant šį metodą verslo transakcija suprantama kaip nedalomas veiklos vienetas, vaizduojamas pragmatiškai motyvuota komunikacine kilpa.

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.

Išplėstinių komunikacinių kilpų modelis verslo transakcijoms specifikuoti yra nepriklausomas nuo konkrečios platforminės IS realizacijos. Tai yra kanoninis vienetas, kuris yra labai aiškiai suprantamas ir atitinka pačią IS projektavimo koncepciją – formalų procesų aprašymą. Šiame darbe pateiktas siūlymas pritaikyti šį modelį konkrečiai realizacijai. Komunikacinėmis kilpomis aprašytus aktoriaus procesus realizuoti trijų lygių architektūroje. Toks platforminis sprendimas ypatingai tinka elektroniniam verslas – verslui (B2B) procesams realizuoti.

Kad užtikrinti patikimą duomenų apsikeitimą tarp dviejų ar kelių verslo partnerių elektroninėje erdvėje, būtinas konkretus ir aiškus duomenų apibrėžimas. Kadangi IS projektavimui siūlomas trijų lygių platforminis sprendimas, kurių konkrečių realizacijų yra nemažai, reikia parinkti tinkamą duomenų formatą. Tam labai tinka šiuo metu dažnai naudojamas XML standartas. XML duomenų valdymas (gavimas, perdavimas, transformavimas, kūrimas ir pan.) trijų lygių architektūroje gali būti realizuotas duomenų bazės lygyje arba tarnybinės stoties komponentų lygyje. Duomenų bazės lygyje XML srautų valdymas kol kas nėra labai išplėtotas. Tik didžiosios DBVS(Oracle, MS SQL Server) siūlo įrankių XML srautams apdoroti. Kadangi tai yra sąlyginai brangūs komerciniai produktai, XML apdorojimas dažnai vykdomas tarnybinės stoties komponentuose.

Darbe buvo nagrinėjamos dvi tarnybinių stočių komponentų realizacijos – PHP ir JavaServlet.

Sukūrėme duomenų abstrakcijos objektą, kurį pavadinome DAO. Šis objektas yra realizuotas PHP programavimo kalboje, tačiau mes pateikiame jį kaip objekto modelį. Šis objektas yra skirtas jungtis į įvairias duomenų bazes. Jos funkcijos yra generuoti SQL

užklausas patogiu ir priimtinu programuotojams metodu, atlikti šias užklausas į duomenų bazę, gautą atsakymą paversti PHP kalboje priimtinais duomenų masyvais. Programuotojai, gaudami atsakymus asociatyviais masyvais gali nesunkiai ir tvarkingai atlikti tolimesnius dalykinės srities reikalavimus atitinkančią realizaciją. Labai svarbu paminėti, jog DAO užklausų generavimas užtikrina ne tik naudojimo patogumą, tačiau ir saugumą. Tai labai svarbu jei projektai yra prieinami bet kokiam internautui, tarp kurių gali pasitaikyti ir programišių (angl. hacker). Taip labai apribojamos galimybės įvykdyti SQL ataką ir įsiskverbti į informacinę sistemą (angl. SQL injection).

JavaServlet technologijos atveju, XML apdorojimui buvo naudotos tik standartinės priemonės ir bibliotekos. Padarius lyginamąją analizę su DAO objektu galima teigti, kad būtų tikslinga sukurti panašų objektą Java kalbos pagrindu. Taip yra todėl, kad dažnai standartiniuose XML valdymo, jungimosi prie DBVS objektuose yra nemažai perteklinių metodų, sudėtingos jų realizacijos.

Šiame darbe nagrinėjamam modeliui realizuoti buvo atliktas taikymo pavyzdžio tyrimas. Pavyzdini taikymo dalykinė sritis: e-parduotuvė ir pirkimas joje išsimokėtinai sudarant sutartį su lizingo kompanija. Pavyzdžio taikymo metu nustačius verslo procesus, buvo sudarytos juos aprašančios išplėstinės komunikacinės kilpos, kurių procesai buvo realizuoti trijų lygių architektūra. Duomenų apibrėžimui panaudotas XML formatas. Praktiškai patikrintas sukurto DAO objekto veikimas, įsitikinta XML formato universalumu, bei trijų lygių architektūros naudingumu realizuojant B2B procesus.

LITERATŪROS SĄRAŠAS

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

[2] Moss E. Nested Transactions. MIT Press, 1985 pp. 189.

[3] Weikum G., Schek H. Concepts and Applications of Multi-level Transactions and Open Nested Transactions. Morgan Kaufmann, 1992. pp. 146-183.

[4] Grefen P., Vonk J., Apers P. Global transaction support for workflow management systems: from formal specification to practical implementation. The VLDB Journal N 10, 2001, pp. 316-333.

[5] Nemuraitė L., Paradauskas B. Duomenų bazės ir semantiniai modeliai. Monografija. Kaunas: Technologija (2002) pp. 227-243.

[6] Klein J. Transaction Internet Protocol Version 3.0. Proceedings Of The Forty-First Internet Engineering TaskForce. 1998. Los Angeles, USA

[7] Alonso G., Kamath M., Agrawal D., Abbadi A.El, Gunthor R., Mohan C. Failure handling in large scale workflow management systems. Technical Report RJ9913, IBM Almaden Research Center, 1994.

[8] Alonso G, Agrawal D, Abbadi A. El, Kamath M, Gunthor R., Mohan C. Advanced transaction model in workflow context. Proceedings of the 12th International Conerence. on Data Engineering, New Orleans, USA 1996

[9] Krishnamoorty V., Shan M.C. Virtual Transaction Model for Workflow Applications. Procs of SAC'00. Como, Italy. 2000.

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

[11] HP, HP Web Service Transaction 1.0 Tech Preview. 2002 HP. Prieiga per internetą: www.hpmiddleware.com/HPISAPI.dll/hpmiddleware/products/webservices_transactions/default.jsp. Žiūrėta 2006-01-09

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

[13] EbXML.org, 'ebXML – Enabling a global electronic market', Organization for the Advancement of Structured Information Standards (OASIS), 2004. www.ebxml.org (žiūrėta 2006-03-15)

[14] Hakvoort R. EbXML and its impact on conventional Business Information Systems, 1st Twente Student Conference on IT, Enschede 14 June 2004,Faculty of EEMCS – University of Twente, Enschede, The Netherlands

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

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

[17] Nemuraitė L., Paradauskas B., Salelionis L. Extended Communicative Action Loop for Integration of New Functional Requirements // Information Technology and Control. Kaunas: Technologija, 2002, no. 2(23), p. 18–26.

[18] Ričardas Kanaitis, Bronius Paradauskas. Naujų funkcinių reikalavimų įvedimas informacinėse sistemose, Informacinės technologijos: X tarpuniversitetinė magistrantų ir doktorantų konferencija, Kaunas, 2005, pp.155-158

[19] PEAR bibliotekos DB_DataObject PHP objekto dokumentacija. Prieiga per internetą: http://pear.php.net/package/DB_DataObject. Žiūrėta 2006-03-10

[20] OpenSSL informacijos saugykla. Prieiga per internetą: http://www.openssl.org/docs/. Žiūrėta 2006-05-01

[21] Antanas Petrauskas, Karolis Pilypaitis. XML srautų valdymas komunikacinėse kilpose. Straipsnis 11-osios taruniversitetinės doktorantų ir magistrantų konferencijos Informacinės Technologijos 2006 pranešimų laidinyje, II dalis. Kaunas, 2006, p 83-85.

TERMINŲ ŽODYNAS

XML dokumentas – XML programavimo kalba aprašytas dokumentas.
XML schema – XML kalba parašytas dokumentas, skirtas XML dokumentui apibrėžti.
Duomenų bazė – kartu saugomų ir susijusių duomenų visuma, kuriai būdinga integruotumas, neperteklišKumas, nepriklausomumas.
Duomenų bazių valdymo sistema – programų rinkinys, atliekantis duomenų apdorojimo operacijas.

SUTRUMPINIMŲ SĄRAŠAS

XML – išplėstinė žymių kalba

HTML – hiperteksto žymių kalba

DB – duomenų bazė

DBVS - duomenų bazių valdymo sistema

SQL – struktūrizuota užklausų kalba

IS – informacinė sistema

DTD – dokumento tipų aprašymas

PĮ – programinė įranga

DAO – mūsų sukurtas duomenų abstrakcijos modelis

PRIEDAI

3.4 Mokslinis straipsnis, pristatytas konferencijoje Informacinės Technologijos 2006

XML SRAUTŲ VALDYMAS KOMUNIKACINĖSE KILPOSE

Antanas Petrauskas, Karolis Pilypaitis

Kauno technologijos universitetas, Informacijos sistemų katedra

Verslas tobulėja ir keičiasi bemaž kiekvieną dieną. Verslo informacinės sistemos privalo greitai ir lanksčiai prisitaikyti prie šių kintančių pokyčių. Interneto ir e-komercijos veržimasis dar labiau sustiprino šią tendenciją. Kompanijos vis dažniau bendrauja tarpusavyje per internetą. Projektuojant ir atnaujinant informacines sistemas svarbu tinkamai valdyti duomenis ir jų srautus, užtikrinant jų pilnumą ir integralumą. Šiame darbe apžvelgiami XML duomenų valdymo metodai verslo tranzakcijose, pasiūlomas įrankis užtikrinantis saugų ir teisingą duomenų apsikeitimą

1. Elektroninių ryšių valdymas ebXML

EbXML (electronic business eXtensible Markup Language – elektroninio verslo išplėstinio žymėjimo kalba) yra kylantis standartas verslo ryšių valdymo srityje.

EbXML yra gan nauja technologija. EbXML projektas 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 jis buvo pristatytas 1999m.

2004m. kovo mėnesį International Standartization Organization (ISO) patvirtino keturių ebXML OASIS standartų rinkinį. ISO/TS 15000 techninės specifikacijas sudaro keturios dalys, kurių kiekviena atitinka vieną ebXML‘o modulio standartą:

• ISO 15000-1: ebXML partnerio profilio susitarimo specifikacija (ebCPP - Collaborative Partner Profile Agreement Specification)

• ISO 15000-2: ebXML pranešimų apdorojimo specifikacija (ebMS - Messaging Service Specification)

• ISO 15000-3: ebXML registrų informacijos modelis (ebRIM - Registry Information Model)

• ISO 15000-4: ebXML registrų servisų specifikacija (ebRS - Registry Services Specification)

Svarbu suprasti kad ebXML nėra viena technologija, tai greičiau specifikacijų rinkinys skirtas kurti e-verslo struktūrą.

Projekto tikslas buvo suformuluotas ebXML projekto kūrimo pradžioje: „Suteikti atvirą XML grįstą infrastruktūrą, kuri leistų plačiai naudoti elektroninio verslo informaciją veikiančiu, saugiu bei pastoviu būdu visoms šalims“ [2]. Mažos ir vidutinio dydžio įmonės tai pat gali naudoti elektroninio B2B (Business to Business) transakcijas per ebXML.

ebXML komponentai:

- ebXML partnerio profilio susitarimo specifikacija (ebCPP)

  Šios specifikacijos tikslas yra užtikrinti veikimą tarp dviejų informacinių sistemų, net jei jos yra skirtingų kūrėjų.

- ebXML pranešimų apdorojimo specifikacija (ebMS)

  Ši specifikacija apibrėžia pranešimų formatą ir antraštinį dokumentą naudojamus ebXML pranešimų persiuntimui per tokius protokolus kaip SMTP ar HTTP. Taip pat apibrėžia programinės įrangos veiksmus, kuri siunčia ir priima ebXML pranešimus. Ši programinė įranga gali būti realizuota kaip savarankiškas pranešimų valdiklis arba gali būti e-verslo funkcionalumo integracijos produktas arba programinis servisas.

- ebXML registrų servisų specifikacija (ebRS)

  ebXML registrai suteikia stabilų verslo subjekto įvedamų duomenų saugojimą. Tokia informacija yra naudojama palengvinti ebXML grįstam B2B bendradarbiavimui bei transakcijoms. Įvestas turinys gali būti XML schema ir dokumentai, procesų aprašai, ebXML branduolio komponentai, konteksto aprašymai, UML modeliai, informacija apie šalis ir programinės įrangos komponentus.. Ši paviešinta informacija yra saugoma kaip objektai saugykloje ir juos valdo ebXML registrų servisai. Štai tame yra esmė.

- ebXML registrų informacijos modelis (ebRIM)

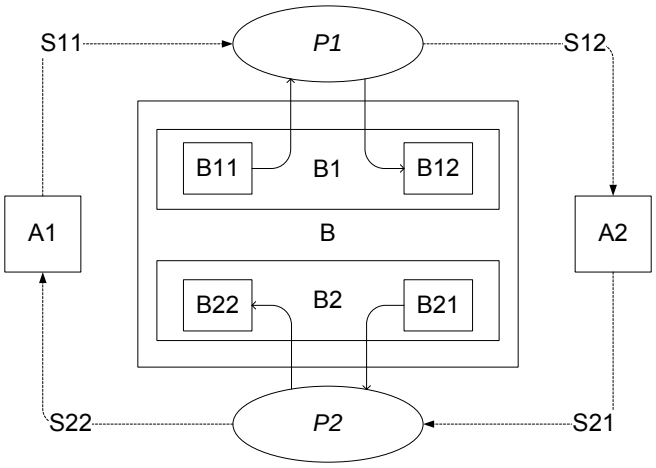
Registro informacijos modelis yra aukšto lygio ebXML registrų schema. Ji apibrėžia registre saugomų objektų tipus ir saugomų objektų organizavimą registre.

- Verslo procesų specifikavimo schema (ebBPSS)

EbBPSS tikslas yra apjungti e-verslo procesų modeliavimą ir e-verslo programinių komponentų specifikavimą. Tiksliau tarant: ebBPSS yra specifikavimo rinkinys, būtinas apibrėžiant dviejų verslo partnerių bendradarbiavimą. Negana to, ji nustato partnerių realaus laiko sistemų parametrus, tam kad e-verslo programiniai komponentai galėtų bendradarbiauti [1].

2.Verslo tranzakcijų specifikavimas komunikacinėse kilpose

Komunikacinė kilpa – būdas formaliai aprašyti verslo tranzakcijas tarp dviejų verslo vienetų. Kiekviena komunikacinė kilpa susideda iš dviejų aktorių – užsakovo A1 ir vykdytojo A2, duomenų srautų S11 – S22 ir dviejų procesų – užsakovo veiksmų P1 ir P2. Duomenų srautų dinamikos apibrėžiamos būsenų perėjimais B.



1 pav. Išplėstoji komunikacinė kilpa

Šis modelis buvo pasiūlytas nagrinėjant naujų funkcinių reikalavimų įvedimą jau sukurtoms ir veikiančioms informacinėms sistemoms.

Toks formalizavimo modelis leidžia užtikrinti kai kuriuos semantinio itegralumo kriterijus:

- Darnos (angl. *coherence*) – galinės būsenos suderinamos su aktorių tisklais.
- Pilnumo (angl. *completeness*) – nėra probleminių būsenų, kurios rodytų būtinybę tobulinti kompiuterizuotą sistemą.
- Įgyvendinamumas (angl. *viability*) – komunikacinė kilpa įgyvendinama, jei

$$A1 \xrightarrow[p]{} B11, A2 \xrightarrow[o]{} B22, A2 \xrightarrow[p]{} B21, A1 \xrightarrow[g]{} B12, \text{ tuomet}$$

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

čia simbolis $\longrightarrow$ žymi pragmatines priklausomybes: $\xrightarrow[g]{}$ reiškia tikslą, $\xrightarrow[p]{}$ problemą, $\xrightarrow[o]{}$ – progą, $\xrightarrow[+]{}$ – teigiamos įtakos priklausomybę ir $\xrightarrow[-]{}$ – neigiamos įtakos priklausomybę. Kitaip tariant, komunikacinė kilpa yra įgivendinama, jei reicipiento problema neigiamai veikia iniciatoriaus problemą ir recipientas gali įvykdyti reakcijos veiksmą, kurio rezultatas teigiamai veikia tiek recipiento, tiek iniciatoriaus tikslus[4].

Įvedant naujus funkcinius reikalavimus, kinta ir duomenų srautai ir jų būsenos. Kai apsiribojame vieninteliu duomenų formatu – XML, komunikacinėmis kilpomis aprašytų tranzakcijų duomenų srautus patogu saugoti duomenų bazėje. Duomenų aprašams naudojami standartiniai XML dokumentų struktūros aprašai (DTD ar XSD).

3.XML valdymas web serverių komponentuose

Keičiantis funkciniams reikalavimams komunikacinėse kilpose keičiasi ir duomenų srautai. Tai nebūtinai yra tokio pat formato egzempliorių kiekių pokytis, o duomenų padaugėja ar sumažėja, kinta duomenų struktūra. Norint kuo greičiau įdiegti pokyčius ir neskiriant daug resursų sistemų perdarymui ar patobulinimui reikia turėti lanksčią sistemą, kurią reikalui esant galima būtų nesunkiai priderinti pasikeitusių funkcinių reikalavimų išpildymui.

Šiuo metu viena populiariausių programavimo kalbų skirtų internetinėms technologijoms yra PHP. Ji populiari ne vien dėl savo paprastumo, tačiau ir dėl plačių galimybių, Norėdami įrodyti, jog tai nėra žemo lygio programavimo kalba ir jog ja galima labai lanksčiai įgyvendinti verslo subjekto poreikius pasirinkome būtent PHP.

Tarkime turime vieną informacinę sistemą, kurią sudaro duomenų bazė, ją aptarnaujanti aplikacija. Ir yra poreikis duomenis siųsi ir priimti iš kitos nutolusios informacinės sistemos. Duomenų srautams perduoti naudojame XML standartą. Taigi iškyla klausimas kaip efektyviai išspręsti duomenų išrinkimą, XML dokumentų generavimą, išsiuntimą ir gautų dokumentų analizavimą, bei importavimą į duomenų bazę. Taip pat šioje komunikacinėje kilpoje labai svarbu užtikrinti duomenų patikimumą, bei saugumą, kadangi jie keliaus internetu.

Šiai užduočiai įgyvendinti sukūrėme PHP objektą, kurį pavadinome DAO (Data Abstraction Object – Duomenų Abstrakcijos Objektas). Jo paskirtis yra jungtis prie bet kokios duomenų bazės, atlikti užduotas užklausas bei išrinkimus, gautus duomenis sustandartizuoti pvz. į XML dokumentus, jei yra poreikis – pasirašyti SSL raktais. Ir lygiai taip pat gavus XML dokumentus, juos patikrinti, bei suimportuoti į duomenų bazę. Šio objekto esmė yra tame, jog jis turi metodus jungimuisi prie duomenų bazės, SQL užklausų generavimui, iš duomenų bazės gautų rezultatų apdorojimui, XML dokumentų generavimui iš PHP duomenų masyvų. Metodų universalūs žinoma sąlyginai, tačiau prireikus kažko ypatingo ko negali gauti naudojant esamus metodus, visuomet galima pasirašyti vaikinę klasę, kuri paveldėtų DAO. Užklausos generuojamos metodams perduodant tokius parametrus: su kokia lentele/lentelėmis bus atliekami veiksmai, ką reikės išrinkti arba ką reikės įterpti, apribojančios sąlygos (where) masyvo struktūra. Pastarąjį masyvą reikia kurti kiekvieno užklausos kvietimosi metu, tačiau tai yra tikrai nesudėtinga – o svarbiausia tai yra lankstu ir universalu, paprasta pritaikyti naujai iškilusių funkcinių reikalavimų išpildymui. DAO gali jungtis prie visų duomenų bazių kurioms PHP turi palaikymo bibliotekas, be to galima Windows operacinės sistemos aplinkoje galima jungtis į ODBC – taigi galima aptarnauti nuo paprasčiausių duomenų bazių kaip MySQL, Access iki aukštesnio lygio – PostgreSQL, Oracle, MSSQL ir k.t. Netgi jei duomenų bazės vienodai išpildo sugeneruotas SQL užklausas, tai pilnai pakanka aprašyti jungimosi ir kreipimosi į duomenų bazę metodus.

Iš duomenų bazės gautų duomenų masyvų galima generuoti XML. Tačiau jei norima duomenis griežčiau kontroliuoti pravartu XML dokumentus generuoti naudojant kokią nors šablonų sistemą. Tai tikrai yra patogu ir vėlgi – paprasta pritaikyti naujiems sistemai keliamiems reikalavimams. Be abejo siunčiant XML reikia sugeneruoti iš XML struktūrą apibrėžiančias schemas, pvz. DTD arba XSD.

Kaip alternatyvą PHP siūlome ir Java Servlet technologiją – tai Sun Microsystems sukurta technologija dinaminių puslapių generavimui. Java Servlet programuojamas Java programavimo kalba. Panašiai kaip ir PHP tai yra serveryje esantis komponentas, susietas su užklausos adresu, generuojantis atsakymus. Naudodami trijų lygių architektūrą (angl. *three tier architecture*) (duomenų bazės lygis – serverio apdorojančio užklausas lygis – atvaidzavimo lygis), galime užtikrinti apsikeitimą duomenimis bei įvairias operacijas su jais. Viduriniame lygyje apdorodami duomenis, galime juos paruošti saugojimui į duomenų bazę ar siuntimui.

XML duomenų saugojimui duomenų bazėje galime naudoti jau anksčiau aprašytus DAO principus (realizuotus Java kalba), o jų apdorojimui Java įrankius darbui su XML, pvz. Sun Microsystems sukurtą klasių rinkinį JAXP (angl. Java API for XML Processing)[5]. Šis ir panašūs įrankiai skirti XML dokumentų apdorojimui ir kūrimui, t.y. duomenų srautų apdorojimui. Naudodami juos kartu su DAO, turime įrankį, kuris užtikrina duomenų integralumą ir pilnumą keičiantis funkciniams reikalavimams.

4.Išvados

Norint garantuoti patikima verslo informacinių sistemų funkcionavimą, būtina formaliai aprašyti verslo tranzakcijas. Projektuojant verslo valdymo sistemas, verslo tranzakcijoms specifikuoti tikslinga naudoti komunikacines kilpas. Duomenų apsikeitimui naudojant XML, duomenų srautus galima patogiai valdyti serverio lygyje panaudojant DAO (Data Abstraciotn Model), kurtą PHP pagrindu. Naudojant šią priemonę yra užtikrinamas greitas ir patogus naujų funkcinių reikalavimų įvedimas ar kitimas duomenų srautų atžvilgiu. Alternatyva šioms priemonėms yra Java Servlet technologijos, kurios leidžia naudotis Java kalba sukurtais įrankiais XML srautams apdoroti.

Literatūra

[1] **Ron Hakvoort,** ebXML and its impact on conventional Business Information Systems, *1st Twente Student Conference on IT, Enschede 14 June 2004,* Faculty of EEMCS – University of Twente, Enschede, The Netherlands

[2] ebXML.org, ‘ebXML – Enabling a global electronic market’, Organization for the Advancement of Structured Information Standards (OASIS), 2004. www.ebxml.org (žiūrėta 2006-03-15)

[3] **Broy M.,** Functional Specification of Time Sensitive Communicating Systems, 2002, VOL 88, SPRINGER VERLAG, ISSN 0258-1248

[4] **Ričardas Kanaitis, Bronius Paradauskas.** Naujų funkcinių reikalavimų įvedimas informacinėse sistemose, *Informacinės technologijos: X tarpuniversitetinė magistrantų ir doktorantų konferencija,* Kaunas, 2005, pp.155-158

[5] **Elliotte Rusty Harold.** Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX. Pearson Education, 2002

[6] **Nemuraitė L, Paradauskas B., Salelionis L.**, Extended Communicative Action Loop for Integration of New Functional Requirements, *Information technology and control.* Kaunas: Tecnologija, 2002, pp. 18-26.

XML Data Control In Communicative Action Loop

In changing world of e-commerce and internet based communication, business transactions and data flows play a key role in successful business management. It is vital to formalize business transactions. Communicative action loop was proposed for modeling these transactions. XML is often used for data exchange. In this paper we propose a multipurpose tool – DAO (Data Abstraction Object), based on PHP to control, manage and save XML document data. A PHP alternative way – Java Servlet technology is also discussed for using Java tools for XML processing.

3.5 Įmonės rekomendacija, kurioje įvertinamas mūsų sukurtas metodas.

Atsiliepimas

2004 metų rudenį Antanas Petrauskas pradėjo tobulinti UAB Fotofabrikas vykdomų projektų informacinių sistemų kūrimo modelį.

Nuolatinis naujų sprendimų ieškojimas bei patirtis leido Antanui sukurti duomenų abstrakcijos objektą DAO. Šis objektas užtikrina netik patikimą ir saugų jungimąsi į duomenų bazes, tačiau kartu padeda patogiai realizuoti šiandien programuotojams iškeltus uždavinius. Šio objekto naudojimas leidžia taupyti programinės įrangos kūrimo kaštus, sutrumpina projektų vykdymo laiką. Taip pat esame labai patenkinti XML duomenų srautų valdymo metodais vykdant informacinius mainus su verslo partneriais. Pasiūlytas būdas ir priemonės XML dokumentų paruošimui, perdavimui ir gavimui mums yra labai priimtinos ir tenkina visus įmonės poreikius.

Esame patenkinti esamais rezultatais bei laukiame naujų įmonei reikalingų inovacinių sprendimų.

2006 05 15 UAB Fotofabrikas direktorius Laimis Inokaitis

3.6 DAO objekto programinis kodas (dao.class.php)

```
<?php

	/**
	*	$id$, $login$, $date$
	*	Class fo DB qction automating (like Data Abstraction Object)
	*/

	class DAO {

		/**
		*	Validation-compatible part
		*	@author Ramūnas Pikčius
		*	2004 09 09
		*/


		var $link = null;				// Database connection resource

		function DAO(){

		}


		/**
		*	Sets DB connection link identifier
		*	@param $link db link resource
		*/
		function setLink(&$link) {

			$this->link =& $link;

		}


		/**
		*	Get db link
		*/
		function getLink() {

			return $this->link;

		}


		/**
		*	Get last db error message
		*	@return error message, string
		*/
		function lastError() {

			return pg_errormessage();

		}


		/**
		*	Returns current value of sequence
		*	@param string $seq pg sequence name, string
		*	@return current sequence value, integer
		*/
		function getSequenceVal($seq) {

			$query = "SELECT currval('$seq') AS curr;";
			$arr = $this->firstRow($query);
			return (!empty($arr['curr'])) ? $arr['curr'] : false;

		}


		/**
		*	Return next value of given sequence
		*	@param string $seq
		*	@return int sequence value, or false on failure
		*/
		function getNextSequenceVal($seq) {

			$query = "SELECT nextval('$seq') AS next;";
			$s = $this->firstRow($query);
			return (!empty($s['next'])) ? $s['next'] : false;
		}



		/**
		*	Escape string for SQL query
		*/
```

```
function escapeSqlStr($string) {

        return pg_escape_string(trim($string));

}


/**
*       Escape string for inserting like Binary Large OBject
*/
function escapeBlob($string) {

        return pg_escape_bytea($string);

}


/**
*       Prepares user input for database - removes tages, escapes spec. chars
*       @param $string user input
*       @return prepared string
*/
function escapeInput(&$string, $striptags = true) {

        $estr = $this->escapeSqlStr($string);
        $estr = ($striptags) ? strip_tags($estr) : $estr;
        return cleanStr($estr);           // remove quotes and < symbol

}

/**
*       Start transaction
*/
function begin() {

        $this->query("BEGIN;");

}

/**
*       End transaction (success)
*/
function commit() {

        $this->query("COMMIT;");

}

/**
*       End transaction (failure)
*/
function rollback() {

        $this->query("ROLLBACK;");

}

/**
*       DBWrapper class (validation incompatible)
*       @author Antanas Petrauskas
*       2004 09 09
*
*/


var $lst, $last_notice, $last_error;

/**
*       uzklausu outputinimo aktyvavimas
*/
function lst( $activate=true ){

        $this->lst = $activate;

}


// request_logger_id

var $request_logger_id = null;
var $request_logger_count = 0;
var $request_logger_max_count = 5;
var $request_logger_min_time = 1.0;

var $total_queries_time = 0;
var $total_queries_count = 0;
var $total_fetch_time = 0;
var $total_fetch_count = 0;
/**
*       uzklausos vykdymas
*/
```

```
		function query( $q, $print = '' ){

			//$this->lst();
			/*if (!isset($this->i)) $this->i = 1;
			$this->i += 1;
			lst($this->i);*/



			if ( $print || $this->lst){
				lst( $q, true );
			}

			$t1 = getMicrotime();

			/* add @ here whene releasing */
			$out = @pg_query($this->link, $q);

			$tt = getMicrotime() - $t1;

			//$this->total_queries_time += $tt;
			//$this->total_queries_count += 1;

			$_SESSION['dao']['total_queries_time'] += $tt;
			$_SESSION['dao']['total_queries_count'] += 1;


				/**
				*	DAO error logger
				*/
				$post = array();
				if (pg_last_error($this->link)){
					$post['query'] = $q;
					$post['last_error'] = pg_last_error($this->link);
				}elseif (pg_last_notice($this->link)){
					$post['query'] = $q;
					$post['last_notice'] = pg_last_notice($this->link);
				}elseif($tt > $this->request_logger_min_time){
					$post['query'] = $q;
					$post['last_notice'] = 'NOTICE: query time is too long';
					$post['query_time'] = $tt;
					$no_exit = true;
				}

				if (!empty($post)){

					$this->request_logger_count += 1;

					if ($this->request_logger_count <= $this-
>request_logger_max_count){
						if ($this->request_logger_id){
							$post['request_logger_id'] = $this-
>request_logger_id;
						}

						if (!isset($no_exit)){
							lst($post, 'red');
						}else{
							lst($post, 'blue');
						}

						## YES/NO INSERT
						//$this->insert(TBL_DAO_LOGGER, $post);

						if (GLOBAL_DEBUG_LEVEL && !isset($no_exit)){
							//exit();
						}
					}
				}

			if ($out) {
				return $out;
			}
			return false;


		}


		/**
		*	uzklausos rezultato skaitymas
		*/

		function fetch_assoc( $r ){

			$t1 = getMicrotime();

			$out = @pg_fetch_assoc( $r );

			$tt = getMicrotime() - $t1;
```

```
		//$this->total_fetch_time += $tt;
		//$this->total_fetch_count += 1;
		$_SESSION['dao']['total_fetch_time'] += $tt;
		$_SESSION['dao']['total_fetch_count'] += 1;

		return $out;
	}


	/**
	*	vykdo select'a
	*/
	function select($table, $what='*', $where='', $run=true){

		if (is_array($where)){
			$where = $this->prepare_where($where);
		}

		$q = "SELECT $what FROM $table $where;";
		if ($run){
			return $r = $this->query( $q );
		}else{
			return $q;
		}
	}


	/**
	*	vykdo select'a ir grazina pirma
	*/
	function get_first($table, $what='*', $where=''){

		if (is_array($where)){
			$where = $this->prepare_where($where);
		}

		$q = "SELECT $what FROM $table $where LIMIT 1;";
		return $this->fetch_assoc( $this->query( $q ) );
	}


	/**
	*	vykdo select'a ir grazina viska
	*/
	function get_all($table, $what='*', $where=''){

		if (is_array($where)){
			$where = $this->prepare_where($where);
		}

		$q = "SELECT $what FROM $table $where;";
		$r = $this->query( $q );
		$rz = array();
		while ($row = $this->fetch_assoc( $r )){
			$rz[] = $row;
		}
		return $rz;
	}


	/**
	*	vykdo insert'a
	*/
	function insert($table, $values_array, $do_what=array('escape','striptags'), $run=true){

		$q_1 = "INSERT INTO $table ( ";
		$q_2 = "( ";

		foreach ($values_array as $key => $value){
			$q_1 .= "\"".$this->text_input($key, $do_what)."\",";
			if ($value != '' || $value === 0){
				$q_2 .= "'".$this->text_input($value, $do_what)."',";
			}else{
				$q_2 .= "NULL,";
			}
		}
		$q_1 = substr($q_1, 0, strlen($q_1)-1).")";
		$q_2 = substr($q_2, 0, strlen($q_2)-1).")";

		$q = $q_1." VALUES ".$q_2.";";

		if ($run){
			return $r = $this->query( $q );
		}else{
			return $q;
		}
	}


	/**
	*	vykdo update'a
```

```
		*/
		function update($table, $values_array, $where='', $do_what=array('escape','striptags'),
$run=true){

			if ($where != ''){
				if (is_array($where)){
					$where = $this->prepare_where($where);
				}

				$q_1 = "UPDATE $table SET ";
				foreach ($values_array as $key => $value){

					$q_1 .= "\"".$this->text_input($key, $do_what)."\"=";

					if ($value != '' || $value === 0){
						$q_1 .= "'".$this->text_input($value, $do_what)."',";
					}else{
						$q_1 .= "NULL,";
					}

				}
				$q_1 = substr($q_1, 0, strlen($q_1)-1);

				$q = "$q_1 $where ;";

				if ($run){
					return $r = $this->query( $q );
				}else{
					return $q;
				}
			}
		}


		/**
		*	vykdo delete'a
		*/
		function delete($table, $where='', $run=true){

			if ($where != ''){
				if (is_array($where)){
					$where = $this->prepare_where($where);
				}

				$q = "DELETE FROM $table $where;";
				if ($run){
					return $r = $this->query( $q );
				}else{
					return $q;
				}
			}
		}


		/**
		*	Paruosia Where_ su Order_By, Limit_ ir Group_By
		*/
		function prepare_where($where){

			$do_what = array('escape');
			$do_what_key = array('dbkeys_security', 'escape');

			$where_ = " WHERE true ";
			$order_ = '';
			$offset_= '';
			$limit_ = '';
			$group_ = '';
			if (is_array($where)){
				foreach ($where as $key => $val){
					switch ($key){
						case "_any_":

							if (is_array($val)){
								foreach ($val as $val_){
									$where_ .= " AND " . $this-
>text_input( $val_ );
								}
							}else{
								$where_ .= " AND " . $this-
>text_input( $val );
							}
						break;
						case "_ilike_":
							if (is_array($val)){
								$where_ .= "AND ( false ";
								foreach ($val as $key_ => $val_){
									$where_ .= " OR ( true ";
									if (is_array($val_)){
										foreach ($val_ as
$val__){
```

```
                                                                                                $where_ .= "
AND \"".$this->text_input( $key_, $do_what_key )."\" ILIKE '%". $this->text_input( $val__, $do_what )
."%'";
                                                                                        }
                                                                                }else{
                                                                                        $where_ .= " AND
\"".$this->text_input( $key_, $do_what_key )."\" ILIKE '%". $this->text_input( $val_, $do_what ) ."%'";
                                                                                }
                                                                                $where_ .= " )";
                                                                        }
                                                                        $where_ .= " ) ";
                                                                }
                                                        break;
                                                        case "_compare_":
                                                                if (is_array($val)){
                                                                        foreach ($val as $key_sign =>
$val_sign){
                                                                        if (is_array($val_sign)){
                                                                                foreach ($val_sign as $key_ =>
$val_){
                                                                                        $where_ .= " AND
\"".$this->text_input( $key_, $do_what_key )."\" $key_sign '" . $this->text_input( $val_, $do_what ) . "'";
                                                                                }
                                                                        }
                                                                }
                                                        break;
                                                        case "_null_":
                                                                $where_ .= " AND ". $this->text_input( $val,
$do_what_key ) ." IS NULL ";
                                                        break;
                                                        case "_not_null_":
                                                                $where_ .= " AND ". $this->text_input( $val,
$do_what_key ) ." IS NOT NULL ";
                                                        break;
                                                        case "_order_":
                                                                $order_ = " ORDER BY " . $this->text_input(
$val, $do_what_key );
                                                        break;
                                                        case '_offset_':
                                                                $offset_ = " OFFSET " . $this->text_input(
$val, $do_what );
                                                        break;
                                                        case '_limit_':
                                                                $limit_ = " LIMIT " . $this->text_input( $val,
$do_what );
                                                        break;
                                                        case "_group_":
                                                                $group_ = " GROUP BY " . $this->text_input(
$val, $do_what_key );
                                                        break;
                                                        default:
                                                                // konf. algoritmas
                        }

                        /**
                        *       Paruosia text'a irasymui i DB pagal text_input funkcija
                        */
                        function text_input($text, $do_what=array() ){

                                return text_input($text, $do_what);

                        }

                        /**
                        *       DB int array to PHP array
                        */
                        function extract_array_field( $arr_field, $fill_keys=true ){

                                $rez = $arr_field;
                                if ($arr_field){
                                        $arr_field_ = substr($arr_field, 1, -1);
                                        if ($arr_field_){
                                                $rez = array();
                                                $arr = explode(',', $arr_field_);
                                                foreach ($arr as $val){
                                                        if ($fill_keys){
                                                                $rez[$val] = $val;
                                                        }else{
                                                                $rez[] = $val;
                                                        }
                                                }
                                        }
                                }
                                return $rez;
                        }


        }

?>
```

3.7 XML duomenų srautų valdymo verslo komunikacinėse kilpose modelio perspektyvumo tyrimas SWOT metodu

XML duomenų srautų valdymo verslo komunikacinėse kilpose modelio perspektyvumo tyrimas SWOT metodu

Darbo tikslas

Pagrįsti XML duomenų srautų verslo komunikacinėse kilpose modelio sudarymo perspektyvumą naudojant SWOT analizę.

SWOT analizė – kokybinis etapas

Naudojant programą SWOT Expert buvo sudarytas toks sistemą įtakojančių stiprybių, silpnybių, galimybių ir grėsmių sąrašas:

Stiprybės

- Auksta projektuotoju kvalifikacija
- Platus realizacijos technologiju spektras
- Galima naudoti vien nemokama programine iranga
- Patogus metodas verslo transakciju specifikavimui bei realizavimui

Silpnybės

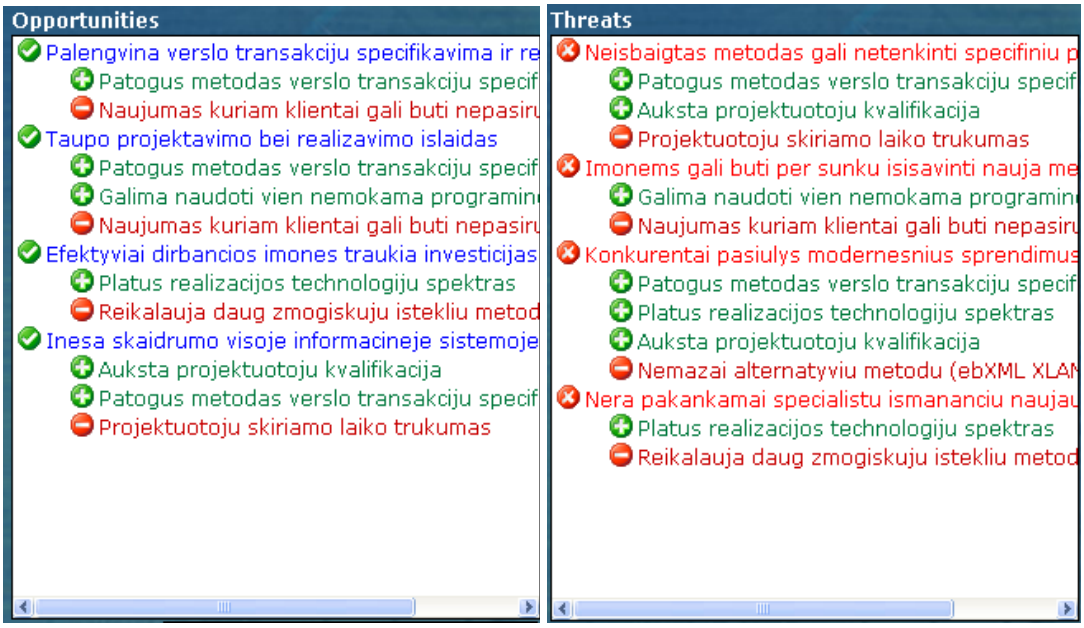
- Nemazai alternatyviu metodu (ebXML XLANG ir kt.)
- Reikalauja daug zmogiskuju istekliu metodo populiarinimui
- Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike
- Projektuotoju skiriamo laiko trukumas

Galimybės

- Palengvina verslo transakciju specifikavima ir realizavima
- Taupo projektavimo bei realizavimo islaidas
- Efektyviai dirbancios imones traukia investicijas is uzsienio
- Inesa skaidrumo visoje informacineje sistemoje

Grėsmės

- Neisbaigtas metodas gali netenkinti specifiniu poreikiu
- Imonems gali buti per sunku isisavinti nauja metoda
- Konkurentai pasiulys modernesnius sprendimus
- Nera pakankamai specialistu ismananciu naujausias technologijas



1 pav. Galimybes ir grėsmes veikiantys faktoriai

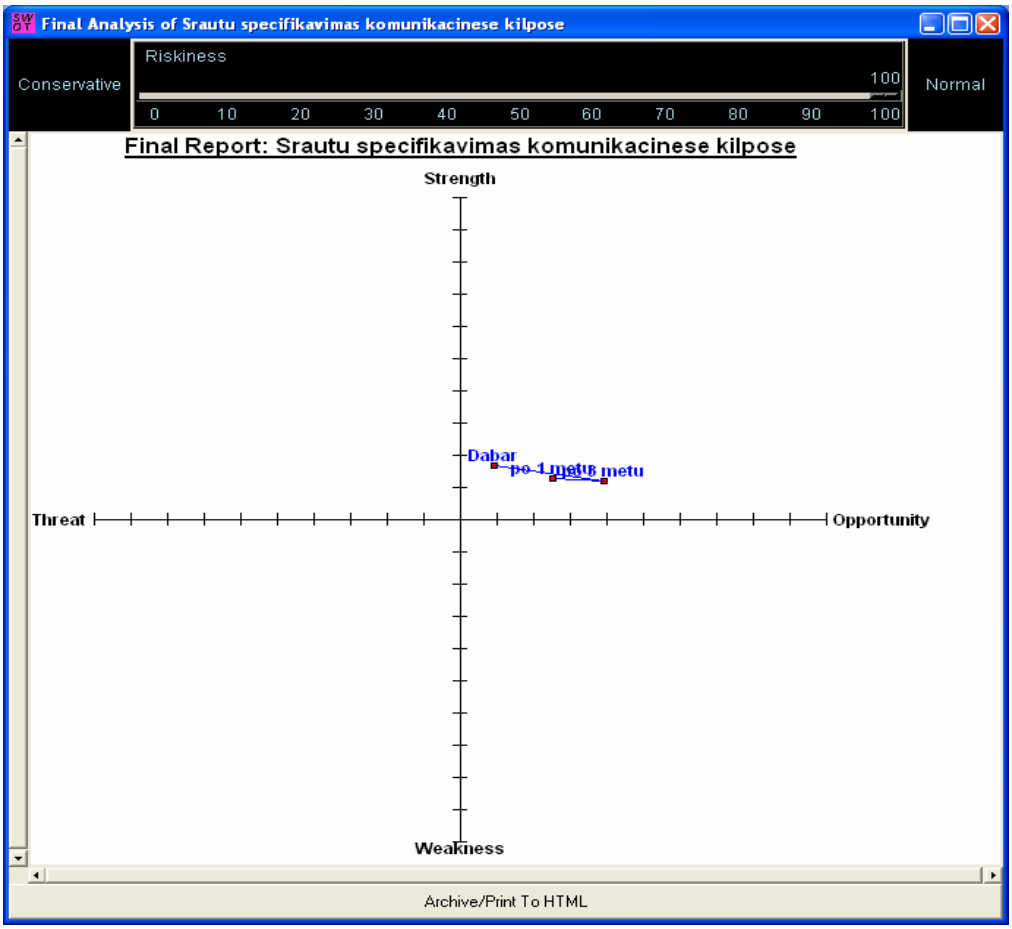
SWOT analizė – kiekybinis etapas

Naudojant programą WinSWOT buvo nustatytos šių stiprybių, silpnybių, galimybių ir grėsmių tikėtinumas ir įtakos stiprumas įvairiais laikotarpiais.

Nustatant kiekybinius rodiklius buvo bandoma lyginti atskirus faktorius pagal įtaką (naudojantis ir kokybiniame etape rastais ryšiais tarp stiprybių bei silpnybių ir galimybių bei grėsmių) ir pagal pasireiškimo tikėtinumą.

Buvo išskirti trys laikotarpiai: dabartis, laikas, praėjus metams, ir laikas, praėjus trejiems metams.

SWOT analizės išvados



2 pav. WinSWOT programa sugeneruota schema

Atliktas tyrimas parodė (2 pav.), kad dabar yra palankus metas vystyti XML srautų valdymo komunikacinėse kilpose metodą. Ateityje, dėl didelės grėsmių įtakos šio metodo aktualumas mažės, tačiau nepaisant to bus pakankamai didelis šio metodo naudojimo ir plėtimo poreikis. Aktualumo mažėjimas susijęs su sparčia alternatyvių metodų plėtra.

Norint sumažinti grėsmių įtaką reiktų daugiau laiko skirti metodo išbaigimui, atsižvelgiant į specifinius verslo klientų poreikius, o taip pat metodo populiarinimui.

SWOT Expert programa gauta ataskaita

SWOT Report

Srautu specifikavimas komunikacinese kilpose

Riskiness value: 100/100

(Low riskiness values indicate conservative assessment, where impact is weighted more than likelihood; high riskiness values indicate unlikely events are weighted less, regardless of their potential impact.)

Issues

- **STRENGTHS**
  - ***Dabar***
    - Issue
      - Patogus metodas verslo transakciju specifikavimui bei realizavimui
      - Patogus metodas verslo transakciju specifikavimui bei realizavimui
      - Impact=100 :: Likelihood=90 :: Overall=90
    - Issue
      - Galima naudoti vien nemokama programine iranga
      - Galima naudoti vien nemokama programine iranga
      - Impact=80 :: Likelihood=100 :: Overall=80
    - Issue
      - Platus realizacijos technologiju spektras
      - Platus realizacijos technologiju spektras
      - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
      - Auksta projektuotoju kvalifikacija
      - Auksta projektuotoju kvalifikacija
      - Impact=100 :: Likelihood=81 :: Overall=81
  - ***po 3 metu***
    - Issue
      - Patogus metodas verslo transakciju specifikavimui bei realizavimui(3)
      - Patogus metodas verslo transakciju specifikavimui bei realizavimui
      - Impact=70 :: Likelihood=60 :: Overall=42
    - Issue
      - Galima naudoti vien nemokama programine iranga(3)
      - Galima naudoti vien nemokama programine iranga
      - Impact=50 :: Likelihood=100 :: Overall=50
    - Issue
      - Platus realizacijos technologiju spektras(3)
      - Platus realizacijos technologiju spektras
      - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
      - Auksta projektuotoju kvalifikacija(3)
      - Auksta projektuotoju kvalifikacija
      - Impact=100 :: Likelihood=60 :: Overall=60
  - ***po 1 metu***
    - Issue
      - Patogus metodas verslo transakciju specifikavimui bei realizavimui(2)
      - Patogus metodas verslo transakciju specifikavimui bei realizavimui
      - Impact=80 :: Likelihood=70 :: Overall=56
    - Issue
      - Galima naudoti vien nemokama programine iranga(2)
      - Galima naudoti vien nemokama programine iranga
      - Impact=70 :: Likelihood=100 :: Overall=70
    - Issue
      - Platus realizacijos technologiju spektras(2)
      - Platus realizacijos technologiju spektras
      - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
      - Auksta projektuotoju kvalifikacija(2)
      - Auksta projektuotoju kvalifikacija
      - Impact=100 :: Likelihood=80 :: Overall=80
- **WEAKNESSES**
  - ***Dabar***
    - Issue
      - Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike
      - Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike
      - Impact=71 :: Likelihood=90 :: Overall=63
    - Issue
      - Reikalauja daug zmogiskuju istekliu metodo populiarinimui
      - Reikalauja daug zmogiskuju istekliu metodo populiarinimui

        - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
        - Projektuotoju skiriamo laiko trukumas
        - Projektuotoju skiriamo laiko trukumas
        - Impact=59 :: Likelihood=39 :: Overall=23
    - Issue
        - Nemazai alternatyviu metodu (ebXML XLANG ir kt.)
        - Nemazai alternatyviu metodu (ebXML XLANG ir kt.)
        - Impact=49 :: Likelihood=50 :: Overall=24
- ***po 3 metu***
    - Issue
        - Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike(3)
        - Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike
        - Impact=60 :: Likelihood=30 :: Overall=18
    - Issue
        - Reikalauja daug zmogiskuju istekliu metodo populiarinimui(3)
        - Reikalauja daug zmogiskuju istekliu metodo populiarinimui
        - Impact=44 :: Likelihood=50 :: Overall=22
    - Issue
        - Projektuotoju skiriamo laiko trukumas(3)
        - Projektuotoju skiriamo laiko trukumas
        - Impact=70 :: Likelihood=69 :: Overall=48
    - Issue
        - Nemazai alternatyviu metodu (3)
        - Nemazai alternatyviu metodu (ebXML XLANG ir kt.)
        - Impact=78 :: Likelihood=81 :: Overall=63
- ***po 1 metu***
    - Issue
        - Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike(2)
        - Naujumas kuriam klientai gali buti nepasiruose ar kritiskai neusiteike
        - Impact=91 :: Likelihood=70 :: Overall=63
    - Issue
        - Reikalauja daug zmogiskuju istekliu metodo populiarinimui(2)
        - Reikalauja daug zmogiskuju istekliu metodo populiarinimui
        - Impact=77 :: Likelihood=79 :: Overall=60
    - Issue
        - Projektuotoju skiriamo laiko trukumas(2)
        - Projektuotoju skiriamo laiko trukumas
        - Impact=62 :: Likelihood=60 :: Overall=37
    - Issue
        - Nemazai alternatyviu metodu (2)
        - Nemazai alternatyviu metodu (ebXML XLANG ir kt.)
        - Impact=62 :: Likelihood=60 :: Overall=37

- **OPPORTUNITIES**
    - ***Dabar***
        - Issue
            - Palengvina verslo transakciju specifikavima ir realizavima
            - Palengvina verslo transakciju specifikavima ir realizavima
            - Impact=100 :: Likelihood=80 :: Overall=80
        - Issue
            - Taupo projektavimo bei realizavimo islaidas
            - Taupo projektavimo bei realizavimo islaidas
            - Impact=90 :: Likelihood=60 :: Overall=54
        - Issue
            - Efektyviai dirbancios imones traukia investicijas is uzsienio
            - Efektyviai dirbancios imones traukia investicijas is uzsienio
            - Impact=90 :: Likelihood=50 :: Overall=45
        - Issue
            - Inesa skaidrumo visoje informacineje sistemoje
            - Inesa skaidrumo visoje informacineje sistemoje
            - Impact=95 :: Likelihood=70 :: Overall=66
    - ***po 3 metu***
        - Issue

        - Palengvina verslo transakciju specifikavima ir realizavima (3)
        - Palengvina verslo transakciju specifikavima ir realizavima
        - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
        - Taupo projektavimo bei realizavimo islaidas(2)
        - Taupo projektavimo bei realizavimo islaidas
        - Impact=95 :: Likelihood=85 :: Overall=80
    - Issue
        - Efektyviai dirbancios imones traukia investicijas is uzsienio (3)
        - Efektyviai dirbancios imones traukia investicijas is uzsienio
        - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
        - Inesa skaidrumo visoje informacineje sistemoje (3)
        - Inesa skaidrumo visoje informacineje sistemoje
        - Impact=100 :: Likelihood=100 :: Overall=100
- ***po 1 metu***
    - Issue
        - Palengvina verslo transakciju specifikavima ir realizavima (2)
        - Palengvina verslo transakciju specifikavima ir realizavima
        - Impact=100 :: Likelihood=91 :: Overall=91
    - Issue
        - Taupo projektavimo bei realizavimo islaidas(3)
        - Taupo projektavimo bei realizavimo islaidas
        - Impact=100 :: Likelihood=100 :: Overall=100
    - Issue
        - Efektyviai dirbancios imones traukia investicijas is uzsienio (2)
        - Efektyviai dirbancios imones traukia investicijas is uzsienio
        - Impact=90 :: Likelihood=100 :: Overall=90
    - Issue
        - Inesa skaidrumo visoje informacineje sistemoje (2)
        - Inesa skaidrumo visoje informacineje sistemoje
        - Impact=100 :: Likelihood=85 :: Overall=85

- **THREATS**
    - ***Dabar***
        - Issue
            - Neisbaigtas metodas gali netenkinti specifiniu poreikiu
            - Neisbaigtas metodas gali netenkinti specifiniu poreikiu
            - Impact=70 :: Likelihood=90 :: Overall=63
        - Issue
            - Imonems gali buti per sunku isisavinti nauja metoda
            - Imonems gali buti per sunku isisavinti nauja metoda
            - Impact=70 :: Likelihood=80 :: Overall=56
        - Issue
            - Konkurentai pasiulys modernesnius sprendimus
            - Konkurentai pasiulys modernesnius sprendimus
            - Impact=60 :: Likelihood=40 :: Overall=24
        - Issue
            - Nera pakankamai specialistu ismananciu naujausias technologijas
            - Nera pakankamai specialistu ismananciu naujausias technologijas
            - Impact=40 :: Likelihood=70 :: Overall=28
    - ***po 3 metu***
        - Issue
            - Imonems gali buti per sunku isisavinti nauja metoda (3)
            - Imonems gali buti per sunku isisavinti nauja metoda
            - Impact=70 :: Likelihood=40 :: Overall=28
        - Issue
            - Konkurentai pasiulys modernesnius sprendimus (3)
            - Konkurentai pasiulys modernesnius sprendimus
            - Impact=70 :: Likelihood=100 :: Overall=70
        - Issue
            - Nera pakankamai specialistu ismananciu naujausias technologijas (3)
            - Nera pakankamai specialistu ismananciu naujausias technologijas
            - Impact=40 :: Likelihood=20 :: Overall=8

- ***po 1 metu***
  - Issue
    - Neisbaigtas metodas gali netenkinti specifiniu poreikiu (2)
    - Neisbaigtas metodas gali netenkinti specifiniu poreikiu
    - Impact=70 :: Likelihood=60 :: Overall=42
  - Issue
    - Imonems gali buti per sunku isisavinti nauja metoda (2)
    - Imonems gali buti per sunku isisavinti nauja metoda
    - Impact=69 :: Likelihood=80 :: Overall=55
  - Issue
    - Konkurentai pasiulys modernesnius sprendimus (2)
    - Konkurentai pasiulys modernesnius sprendimus
    - Impact=60 :: Likelihood=80 :: Overall=48
  - Issue
    - Nera pakankamai specialistu ismananciu naujausias technologijas (2)
    - Nera pakankamai specialistu ismananciu naujausias technologijas
    - Impact=40 :: Likelihood=40 :: Overall=16