KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMCIJOS SISTEMŲ KATEDRA

Aidas Oželis

Elektroninio verslo procesų modeliavimo ir tinklo paslaugų kūrimo metodika

Magistro darbas

Darbo vadovė:
doc. dr. L.Nemuraitė

Kaunas
2004

Turinys

1. Įvadas

Galimybė komunikuoti už įmonės ribų yra būtina sąlyga ateities kompiuterizuotoms verslo sistemoms. Tai apima ne tik komunikavimą su klientais bei vartotojais ir administracija, bet ir su kitomis įmonėmis.

Tradicinė įmonės paslaugų integracija per EDI buvo priklausoma nuo konkrečios programinės įrangos ir kontroliuojama rankiniu būdu. Naujų ryšių tarp partnerių sukūrimas buvo ilgas bei brangus procesas. Kompanijų viduje veikė pažangios komponentinės technologijos, kaip CORBA, DCOM, Java RMI, kurios leido patogiai ir dinamiškai integruoti naujus programinės įrangos komponentus. Šių komponentų paslaugas, publikuojamas vietiniame tinkle, atrasdavo kiti komponentai ir sukurdavo su jais ryšį.

Pritaikymui Internete tokios technologijos reikalauja pernelyg glaustų tarpusavio sąsajų ir kelia saugumo problemų, todėl Internete klasikinės autonominės programos nyksta. E. verslo programos vis labiau virsta į dinamišką Web Servisų rinkinį.

Web Servisai apjungia kompiuterius bei kitus įrenginius per Internetą duomenų apsikeitimui. Web Servisus galima laikyti kaip PĮ komponentus, kurie surenkami Internete naudojant standartinius protokolus ir atlieka tam tikras funkcijas arba vykdo tam tikrą verslo procesą. Web Servisų esmė yra dinamiškas PĮ kūrimas iš laisvai susietų, pakartotinio panaudojimo PĮ komponentų, kas yra labai svarbu tiek technologiniu, tiek verslo požiūriu. Verslo užduotys gali būti sprendžiamos automatiškai bendradarbiaujant sistemoms. Atskirais atvejais Internete teikiami žemesnio lygio servisai tokie kaip: autentikacija, kreditinės kortelės galiojimo nustatymas, tarpininkavimo paslaugos ir t.t. Aukštesnio lygio servisai tokie kaip visas verslo procesas naudoja žemesnio lygio servisus tam tikra seka ar iš karto vienu metu. Verslo paslaugos gali būti visiškai decentralizuotos ir pasiskirstę Internete, bei pasiekiamos įvairiais komunikavimo įrenginiais.

Tačiau vien web servisų nepakanka automatizuotiems verslo procesams vykdyti. Reikalinga infrastruktūra web servisų kompozicijai bei paties proceso apibrėžimo semantika verslo bei jo techninės realizacijos požiūriu. Realiausios rinkos palaikymo susilauksiančios technologijos turėtų remtis jau egzistuojančiais standartais tokiais kaip XML, SOAP, WSDL nereikalaujančiais papildomų technologinių bei žmogiškųjų resursų.

Automatizuotų elektroninio verslo procesų kūrimas reikalauja ne tik išsamios verslo srities analizės bei aiškaus web servisų kūrimo technologijų suvokimo, bet ir efektyvios metodikos leidžiančios taikliai nustatyti verslo sąveikų tipus, dalyvius bei proceso veiklų sekas, efektyviai organizuoti jų vykdymo eiliškumą.

Automatizuoti verslo procesai dar labiau išnaudoja naujausių bei esamų interneto technologijų potencialą bei paverčia internetą globalia bendra verslo platforma, kur komunikuoja žmonės ir organizacijos, atlikdamos įvairias veiklas ir paslaugas. Tai ženkliai sumažina SME (mažų ir vidutinių įmonių) atėjimo į naujas rinkas barjerus. Dinamiškos įmonės ir dinamiškos vertės kūrimo grandinės naudingos ir gal būt būtinos tvirtai konkurencijai užtikrinti.

Šiame darbe analizuojamos web servisų komponavimo į automatizuotus verslo procesus problemos bei jų sprendimo būdai. Tam tikslui apžvelgtos automatizuotų verslo procesų vykdymo kalbos, iš kurių tolimesniam tyrimui pasirinkta naujausia verslo procesų vykdymo kalba BPEL4WS. Išanalizuota BPEL4WS specifikacija bei verslo procesų modeliavimas su BPWS4J moduliu IBM Eclipse platformoje. Norint modeliuoti BPEL4WS aprašomus procesus aukštesniame konceptualiajame lygyje ir standartinėmis programinės įrangos projektavimo priemonėmis, išnagrinėtos galimybės tą atlikti naudojant universalią modeliavimo kalbą UML ir Rational Rose paketą. Sukurta metodika BPEL4WS procesams kurti su UML bei realizuoti. Realizuotas eksperimentinis kelių partnerių elektroninio verslo procesas, kuris vykdomas naudojant BPWS4J varikliuką. Procesui vykdyti reikalingi Web servisai realizuoti Microsoft .NET platformoje.

2. Sąveikaujančių elektroninio verslo procesų įgyvendinimo priemonių bei problemų analizė

2.1. Tyrimo tikslai ir esamos situacijos analizė

Tyrimo sritis – verslo procesų automatizavimas panaudojant web servisų sąveikavimo technologijas.

Tyrimo objektas – tiekimo procesų automatizavimas elektroninio verslo sistemoje

Problema: web servisų kūrimo sritis nauja, trūksta patirties šiai technologijai taikyti. Web servisų sąveikavimo standartai dar tik kuriami, todėl rinkoje beveik nėra nei veikiančių pavyzdžių, nei aiškių metodikų kaip automatizuoti biznio procesus.

Praktinė problema, kurią siekiama išspręsti šiame darbe – sukurti web servisus tiekėjų valdymui elektroninio verslo sistemoje, kadangi praktiškai daugelyje egzistuojančių e.komercijos sistemų tiekėjų valdymo klausimai yra neišspręsti

Tema aktuali tiek teoriniu, tiek praktiniu požiūriu**.** Teoriniai Web servisų modeliavimo, projektavimo klausimai nėra galutinai susiformavę. Pavyzdžiui, tik neseniai pasirodė verslo dokumentų formavimo standarto gairės. Tikslinga kuo greičiau įsisavinti ir taikyti praktikoje web servisų technologiją, kadangi tai efektyvus kūrimo ir daugkartinio panaudojimo būdas. Šiame darbe tikimasi išspręsti tiekėjų valdymo problemą būtent web servisų technologijos priemonėmis.

Kokybiškam Web servisui sukurti reikalingas tinkamas verslo proceso informacinio turinio apibrėžimas, kurio tinkamam realizavimui reikalingi:

- Dokumentų tipai, apibūdinantys verslo dokumentų (pirkimo užsakymų, važtaraščių) turinį.
- Semantika – dokumentų tipų elementų reikšmės turi būti semantiškai korektiškos ir tinkamai interpretuojamos tiek paslaugos tiekėjo, tiek gavėjo. Šiam tikslui reikalingas žodynas. Be to paties dokumento tikslas turi aiškiai semantiškai nurodyti kaip interpretuoti patį dokumentą.

- Transportavimas. Serviso užklausėjas kaip ir serviso tiekėjas turi susitarti kokį transportavimo mechanizmą naudos.
- Pranešimų sekos apibrėžimas reikalingas užtikrinti, jog pranešimai siunčiami po vieną kartą, o pakartotinas pranešimo gavimas yra kitas pranešimas.
- Proceso veiksmų sekos apibrėžimas – verslo pranešimų sekos logika.
- Duomenų apsauga – autorizuotas atitinkamų Web Servisų naudojimas.
- Specifinis verslo partnerio konfigūravimas – dėl skirtingos serviso tiekėjo ir gavėjo verslo logikos gali būti reikalingas specifinis serviso konfigūravimas

Šiuolaikinės e. verslo sistemos turi įgalinti bet ką bendradarbiauti su bet kuo, tai veda prie automatizuoto įvairių duomenų tipų bei verslo logikos palaikymo. Tam reikalinga sukurti išplečiamas tarpininkavimo paslaugas XML pagrindu (XML dokumentu apsikeitimas Web Servisu pagalba).

Verslo procesų automatizavimas reikalauja kurti procesų pagrindu veikiančias programas (process-based applications). Tai tokios programos, kurių struktūra dalijama į du atskirus sluoksnius: viršutinis sluoksnis – biznio proceso sluoksnis XML kalba aprašyta programos vykdymo logika (šiame darbe aprašoma BPEL), o apatiniame sluoksnyje esantys Web servisai atitinka programos funkcinę logiką ir yra vykdomi pagal viršutinio sluoksniu scenarijų.

Tokia architektūra turi keletą ženklių privalumų: tiek biznio procesą tiek pačius web servisus galima keisti ir modifikuoti tiesiogiai neįtakojant vienas kito. Tai suteikia sistemai laibai didelį lankstumą, kas svarbu sąveikaujant su kitomis sistemomis bei įmonėmis.

Darbe bus taikomi objektinės analizės, komponentinio kūrimo ir web servisų kūrimo metodai.

Web servisų kūrimas bus paremtas standartais:

- UDDI – The Universal Description, Discovery, and Integration – klientams teikia Web Servisų suradimo ir registravimo paslaugas.

Naudojant UDDI interfeisą galima dinamiškai susirasti reikalingus ar partnerio siūlomus Web Servisus.

- WSDL – Web Services Description Language – apibūdina Web Serviso prieigą ir protokolus.
- SOAP – Simple Object Access Protocol – pranešimų schema apibūdinanti vienodą XML aprašytų duomenų siuntimą, apibūdina transportavimo formatą.

Web servisų projektavimui bus taikoma UML ir Rational XDE paketas.

Kadangi verslo proceso informacinis turinys e. verslo sistemai apibrėžiamas XML, tai gali būti naudojamas paprastas tekstinis redaktorius. Web Servisų funkcionalumui realizuoti bus naudojamas MS Visual Studio .NET.

2.1.1 Vartotojų bei jų poreikių analizė

Kadangi kuriama sistema iš esmės bus bendradarbiaujančių Web Servisų sistema, šios sistemos vartotojai skirstomi į du tipus: netiesioginiai vartotojai bus Web Servisų paslaugų tiekėjai, o tiesioginiai vartotojai – Web Servisų vartotojai, besinaudojantys Web Servisų paslaugomis. Web Servisai šiuo atveju yra pagrindinis partnerių bendradarbiavimo mechanizmas vykdant atitinkamus verslo procesus, todėl pagrindiniai vartotojų poreikiai yra, kad Web Servisai kiek įmanoma geriau realizuotų jų verslo procesą ir tuo pačiu atitiktų bendrąją proceso sąsają, kad Web Servisą kaip verslo proceso etapą galėtų naudoti kiti verslo partneriai vykdantis atitinkamus verslo uždavinius.

2.1.2 Informacijos sistemai keliami nefunkciniai reikalavimai ir apribojimai

Pagrindinis Web Servisų nefunkcinis reikalavimas yra tai, kad jie turi būti prieinami kiekvieną dieną bei ištisą parą. Turi būti nesunkiai išplečiami. Jų sąsaja turi

išlikti pastovi, nors realizacija smarkiai pakistų, todėl labai svarbu iš anksto tinkamai išanalizuoti verslo procesą.

2.2 Verslo procesų modeliavimo kalbų, atliekančių tinklo paslaugų kompoziciją, apžvalga

Kadangi Web Servisų kūrimo technologijos yra šviežias reiškinys, lietuviškos literatūros šia tema kol kas nėra, be to, dauguma informacijos gauta iš interneto. Dauguma literatūros šaltinių apžvelgia BizTalk bei ebXML technologijas.

Web Servisų įtraukimas į verslo proceso vykdymą galimas keliais automatizavimo laipsniais:

- "Rankinis" Web Servisų iškvietimas naudojant naršyklę
- Programinis iškvietimas – kuomet konkretų Web Servisą naudoja programa konkrečiai užduočiai atlikti.
- Iškvietimas susikurtų verslo procesų kontekste – BizTalk Orchestration
- (pusiau) automatinis Web servisų suradimas, (pusiau) automatinis verslo sąlygų aptarimas – ebXML

Pirmi du metodai vaizduoja žmogaus valdomus procesus, o likę du yra automatizuoti workflow tipo procesai, kur žmogus tik pateikia tam tikrus sprendimus, bet proceso vykdymu nesirūpina.

ebXML ne tik apibrėžia globalų visuotiną verslo dokumentų apsikeitimo standartą, bet ir teikia infrastruktūrą verslo bendradarbiavimui, kuri leidžia automatizuoti verslo partnerių suradimo mechanizmą. Infrastruktūra sukurta standartinių pranešimų siuntimo protokolų pagrindu yra saugi ir patikima. Semantinis ebXML metamodelis apibrėžia verslo procesus ir informacijos modelius, pakartotinio panaudojimo esminius komponentus (core components) bei pranešimų struktūros apibrėžimo mechanizmą. Automatizuotas verslo partnerių suradimo, derybų bei bendradarbiavimo mechanizmas įgyvendinamas bendrų saugyklų principu, kur įmonės registruojasi ir nurodo savo paslaugas bei verslo procesų logiką bei semantiką.

Microsoft BizTalk Server pranešimų sistema remiasi papildoma SOAP pranešimų sistemos logika. BizTalk framework'as nereikalauja tikslių individualių dokumentų turinio ar struktūros apibrėžimų, nes juos nusako iš anksto apibrėžtos verslo esybės. Verslo procesų apibrėžimas ir vykdymas remiasi XLANG –Microsoft sukurta XML kalba. Tačiau BizTalk vartotojas, jos nenaudoja, o verslo procesus aprašo struktūrinėmis schemomis (su Visio ar BizTalk'e integruota procesų kūrimo sistema) iš kurių generuojamas XLANG. Automatizuoto procesų kūrimo nėra, nes BizTalk orientuotas į mažas įmones, kur procesų kūrimą gali kontroliuoti žmogus. Identifikavus abstrakčių procesų veiksmų eigą, kiekvienam veiksmui priskiriama jo realizacija (dažniausiai COM+ verslo objektų komponentai). Šis priskyrimas dinamiškas ir gali kisti laike, priklausomai nuo proceso eigos. Šie komponentai jungiasi su duomenų komponentais bei Web Servisais.

BizTalk palaiko tradicines (ACID), sudėtines bei, kas svarbiausia e. versle – ilgalaikes transakcijas.

WSFL – Web Services Flow Language –XML kalba aprašanti Web Servisų komponavimo struktūrą. WSFL nagrinėja du Web Servisų komponavimo tipus: pirmas kuomet specifikuojamas Web Servisų komponavimas kokio nors proceso realizavimui, antra kuomet specifikuojama Web Servisų sąveika apibūdinanti verslo partnerių bendradarbiavimą.

WSMF – Web Service Modeling Framework – Web Servisų modeliavimo rekomendacija siekianti įgalinti pilnai išplečiamą ir lankstų e. verslo kūrimo modelį Web Servisų pagrindu. Šis tikslas pasiekiamas dviejų viena kitą papildančių architektūrų principu: maksimalus e.verslo sistemą realizuojančių komponentų nepriklausomumas vienas nuo kito bei stiprių tarpininkavimo paslaugų kūrimas leidžiant įvairialypes sąveikas. WSMF išplečia WSFL atskirdama privačius procesus nuo viešų bei panaudodama XML ontologijas leidžiančias pakartotiną procesų apibūdinimo panaudojimą.

BPML – Business Process Modeling Language – XML verslo procesų modeliavimo kalba specifikuojanti abstraktų verslo procesų bei jų objektų modelį, kuris

aprašo įvairius įmonės verslo procesus, apimant įvairaus sudėtingumo veiksmus bei transakcijas, duomenų valdymą, lygiagretų vykdymą ir kitą veikimo semantiką. Taip pat pateikiama XML Schema leidžianti įvairalypėms sitemoms suprasti šį modelį.

2.3 Vykdomosios elektroninio verslo kalbos BPEL4WS analizė

2.3.1 Tinklo paslaugų kompozicija į procesus BPEL4WS kalboje

Web servisų esmė yra universalus sąveikavimas tarp programų naudojant egzistuojančius interneto standartus. Technologiniu požiūriu Web Servisai sudaryti iš trijų lygmenų, kurių kiekvienas atsakingas už atskiras Web Serviso funkcionavimo sritis:

- UDDI – The Universal Description, Discovery, and Integration – klientams teikia Web Servisų suradimo ir registravimo paslaugas. Naudojant UDDI interfeisą galima dinamiškai susirasti reikalingus ar partnerio siūlomus Web Servisus.
- WSDL – Web Services Description Language – apibūdina Web Serviso prieigą ir protokolus.
- SOAP – Simple Object Access Protocol – pranešimų schema apibūdinanti vienodą XML aprašytų duomenų siuntimą, apibūdina transportavimo formatą.

Sistemų sąveikai reikia daugiau nei tik galimybės paprastai sąveikauti naudojant standartinius protokolus. Web servisų technologija kaip įkomponavimo platforma įgyja pilną potencialą tik tuomet kai programos bei biznio procesai gali integruoti sudėtingas sąveikas naudojant standartizuotą procesų apjungimo modelį. WSDL teikiamas sąveikos modelis iš esmės yra būsenų neišlaikančių (stateless) sinchroninių ar nesusijusių asinchroninių sąveikų modelis. Biznio procesų sąveikos modeliai reikalauja sinchroninių bei asinchroninių pranešimų apsikeitimo išlaikant būseną ilgai trunkančiose sąveikose apimančiose du ar kelis partnerius. Tokių sąveikų apibrėžimui reikalingas formalus pranešimų apsikeitimo protokolo apibrėžimas. Tokių

biznio protokolų apibrėžimas apima tikslias visiems proceso dalyviams matomų pranešimų apsikeitimų specifikacijas neatskleidžiant jų vidinio realizavimo.

Business Process Execution Language For Web Services – BPEL4WS (sutrumpintai BPEL) suteikia galimybę specifikuoti biznio procesus ir tai kaip jie siejasi su Web servisais. BPEL yra apjungtos IBM Web Services Flow Language – WSFL bei Microsoft XLANG specifikacijos į bendrą standartą, kurio tikslas yra tapti visuotinu web servisų komponavimo standartu.

Vidinio (privataus) procesų realizavimo atskyrimas nuo viešo proceso elgsenos aprašo turi dvi priežastis. Pirmoji yra akivaizdi, jog nėra norima verslo partneriams atskleisti vidinių sprendimų priėmimo bei duomenų tvarkymo aspektų. Kita priežastis yra ta, jog atskirta privačių procesų realizacija gali laisvai kisti neįtakodama paties verslo protokolo.

BPEL4WS naudoja pranešimų savybių notaciją identifikuoti su pačiu protokolu susijusią informaciją. Savybės gali būti traktuojamos kaip „skaidrūs“ duomenys kurie siejasi su viešaisiais protokolo aspektais, priešingai nei „nematomi“ duomenys naudojami vidinėse/privačiose funkcijose. Skaidrūs duomenys veikia viešą biznio protokolą tiesiogiai, o nematomi duomenys pirma apdorojami „back-end“ sistemose ir biznio protokolą įtakoja neapibrėžtai, kadangi sprendimų priėmimo būdas yra neaiškus. Iš esmės protokolo vykdymą įtakojantys duomenys turi būti skaidrūs, t.y. traktuojami kaip savybės. Pvz.: užsakymo protokole pardavėjas turi servisą kuris priima užsakymą ir siunčia sutikimą ar nesutikimą vykdyti užsakymą, kuris remiasi įvairiais kriterijais (prekių buvimas sandėlyje, pasitikėjimas pirkėju). Žinoma, sprendimo priėmimo procesas yra nematomas, tačiau sprendimo faktas turi būti atvaizduotas kaip elgsenos alternatyvos viešame biznio protokole. Kitaip tariant protokolas turi turėti pasirinkimo veiksmą pardavėjo serviso elgsenoje, tačiau kas pasirenkama yra neapibrėžta. Tokį neapibrėžtumą galima modeliuoti leidžiant priskirti nematomus duomenis pranešimo savybei, dažniausiai iš išvardintų galimų variantų aibės. Savybė tuomet gali būti panaudota apibrėžiant sąlyginį elgesį apimantį elgsenos alternatyvas neatskleidžiant paties sprendimo priėmimo proceso. BPEL4WS leidžia naudojant neapibrėžtus duomenis atpažinti viešą elgseną ir paslėpti privačius aspektus.

Pagrindinės BPEL4WS sąvokos gali pritaikomos dviem būdais. BPEL4WS procesas gali apibrėžti biznio protokolo rolę, naudojant abstraktaus proceso notaciją. Pvz.: tiekimo grandinės protokole pirkėjas ir pardavėjas yra dvi atskiros rolės kiekviena

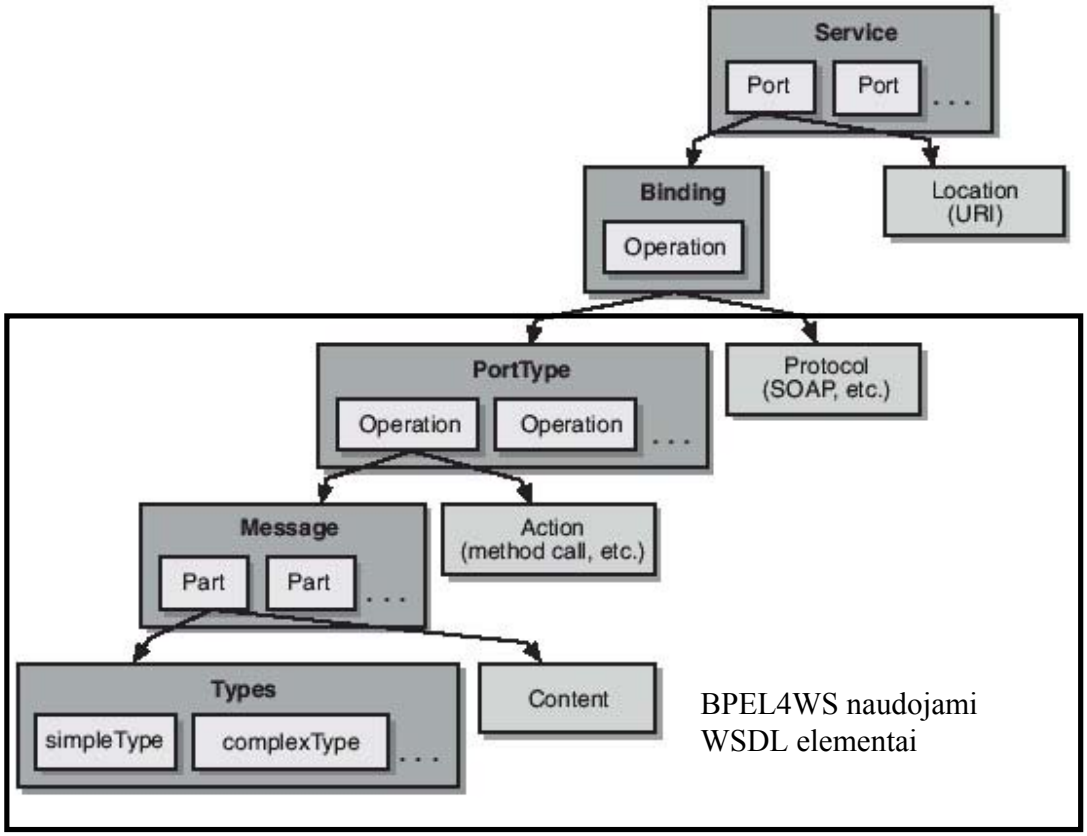
su savais abstrakčiais procesais. Ryšys tarp šių rolių modeliuojamas kaip serviso sąsaja (service link). Abstraktūs procesai apdoroja tik su protokolu susijusius duomenis. BPEL4WS teikiamas su protokolu susijusių duomenų identifikavimo būdas yra pranešimų savybės (message properties). Abstraktūs procesai naudoja neapibrėžtus duomenis tam, kad paslėpti privačius elgsenos aspektus.

BPEL4WS taip pat leidžia apibrėžti vykdomą biznio procesą (executable business process). Proceso logika bei būsena nustato web servisų vykdymo seką. Taip apibrėžti procesai nuosekliai vykdomi nepriklausomai kokioje platformoje, programavimo kalba realizuoja patį procesą skirtinguose sąveikaujančiuose partnerių web servisuose. BPL4WS apibrėžia modelį ir sintaksę apibrėžti biznio procesų elgseną paremtą sąveika tarp partnerių ir paties proceso. Sąveika su kiekvienu partneriu vyksta per web servisų interfeisus, o ryšio tarp partnerių struktūra apibrėžiama serviso sąsajomis (service link). BPL4WS taip pat apibrėžia kaip suderinti sudėtines sąveikas taip pat logiką bei būsenas būtinas tokiam suderinimui. Taip pat yra nustatytas mechanizmas apibrėžti išimtines situacijas bei klaidų apdorojimą, ir kas svarbiausia, yra nustatytas mechanizmas apibrėžti kaip kompensuoti įvykdytus veiksmus, kurių vykdymo metu buvo klaidos ar išimtinės situacijos.

BPL4WS sluoksnis yra aukščiausiai tarp keleto kitų XML specifikacijų: WSDL, XML Schema, bei XPath. WSDL pranešimų ir XML Schema tipų apibrėžimai sudaro duomenų modelį, kurį naudoja BPEL4WS procesai. XPath naudojamas duomenų manipuliavimui. Visi išoriniai resursai bei partneriai yra vaizduojami kaip WSDL servisai.

2.3.2 BPEL4WS struktūra

BPL4WS susideda iš dviejų pagrindinių dalių: WSDL pranešimų ir XML Schema tipų apibrėžimų bei paties proceso apibrėžimo. BPL4WS naudoja tik dalį WSDL specifikacijos duomenims apibrėžti: kokie pranešimai bus siunčiami, kokios operacijos bus vykdomos ir kokiems portų tipams priklauso tos operacijos. Tokiu būdu BPL4WS procesas tampa Web servisu. 1 pav. pavaizduota BPEL4WS naudojama WSDL struktūros dalis



1 pav. BPEL4WS naudojami WSDL struktūros elementai.

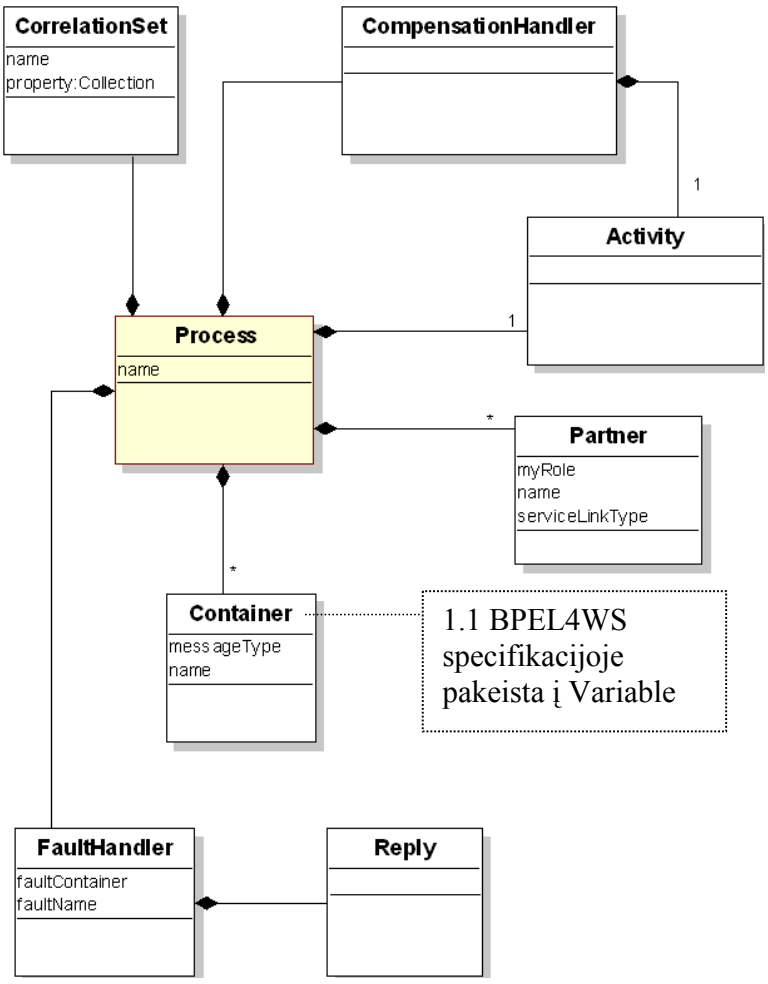
BPL4WS WSDL apibrėžimas yra išplečiamas serviso ryšiu (service link). Serviso ryšio tipas charakterizuoja ryšį tarp dviejų servisų apibrėžiant roles, kurios dalyvauja servisų sąveikoje bei nurodo tų rolių portų tipus. Užsakymų priėmimui naudojamas serviso ryšys:

```
<slnk:serviceLinkType name="PirkimoUzsakymoProcesas:Saskaita">
     <slnk:role name="SaskaitosServisas">
         <slnk:portType name="Uzsakymas:KainosSkaiciavimas"/>
     </slnk:role>
     <slnk:role name="SaskaitosUzklausimas">
         <slnk:portType name="Uzsakymas:SaskaitosAtsakymas"/>
     </slnk:role>
 </slnk:serviceLinkType>
```

Nurodžius vieną rolę serviso ryšio apibrėžime, reiškia, kad servisas gali turėti ryšį su bet kokiu kitu servisu. Taigi XML Schema tipai, WSDL pranešimai (messages), operacijos, portų tipai bei servisų ryšiai apibrėžia proceso duomenis ir užrašomi į *.wsdl failą.

2.3.3 BPEL4WS proceso sudėtis

Surašius visus proceso reikalavimus į *.wsdl failą toliau kuriamas pats procesas. Proceso apibrėžimas rašomas <process> elemente. Po to dažniausiai seka WSDL bei XML Schemų importavimas. BPEL4WS proceso komponentai pavaizduoti 2 pav.



2 pav. BPEL4WS proceso komponentai.

Sekančiu žingsniu apibrėžiami proceso dalyviai. Kiekvienas partneris charakterizuojamas nurodant serviso ryšio tipą. Svarbu atkreipti dėmesį į myRole/partnerRole atributą. Šis atributas nurodo kaip sąveikauja partneris su procesu per nurodytą serviso ryšio tipą. myRole nurodo kokią serviso ryšio rolę atliks procesas, o partnerRole – nurodo atitinkamo partnerio rolę. Be to, BPEL4WS partnerių kiekis nebūtinai turi atitikti realų partnerių skaičių, jis gali būti didesnis, kuomet partneris atlieka keletą funkcijų, jis skaidomas į keletą partnerių. Užsakymo vykdymo BPEL4WS proceso apraše partneriai pavaizduoti 3 pav.

```
  <partners>
    <partner name="Klientas"
xmlns:ns1="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns1:Uzsakymas" myRole="UzsakymoServisas"/>
    <partner name="SaskaitosSiuntimas"
xmlns:ns2="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns2:saskaita" myRole="SaskaitosUzklausimas"
partnerRole="SaskaitosServisas"/>
    <partner name="pristatymoSiuntimas"
xmlns:ns3="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns3:Pristatymas" myRole="pristatymoUzklausimas"
partnerRole="Pristatymas"/>
    <partner name="grafikoSiuntimas"
xmlns:ns4="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns4:Grafikas" partnerRole="Grafikas"/>
  </partners>
<partners>
```

3 pav. proceso partnerių BPEL aprašas

Apibrėžus partnerius toliau galima aprašyti proceso veiklas (activities):

```
<receive>
<reply>
<invoke>
<assign>
<throw>
<terminate>
<wait>
```

<empty>

<sequence>

<switch>

<while>

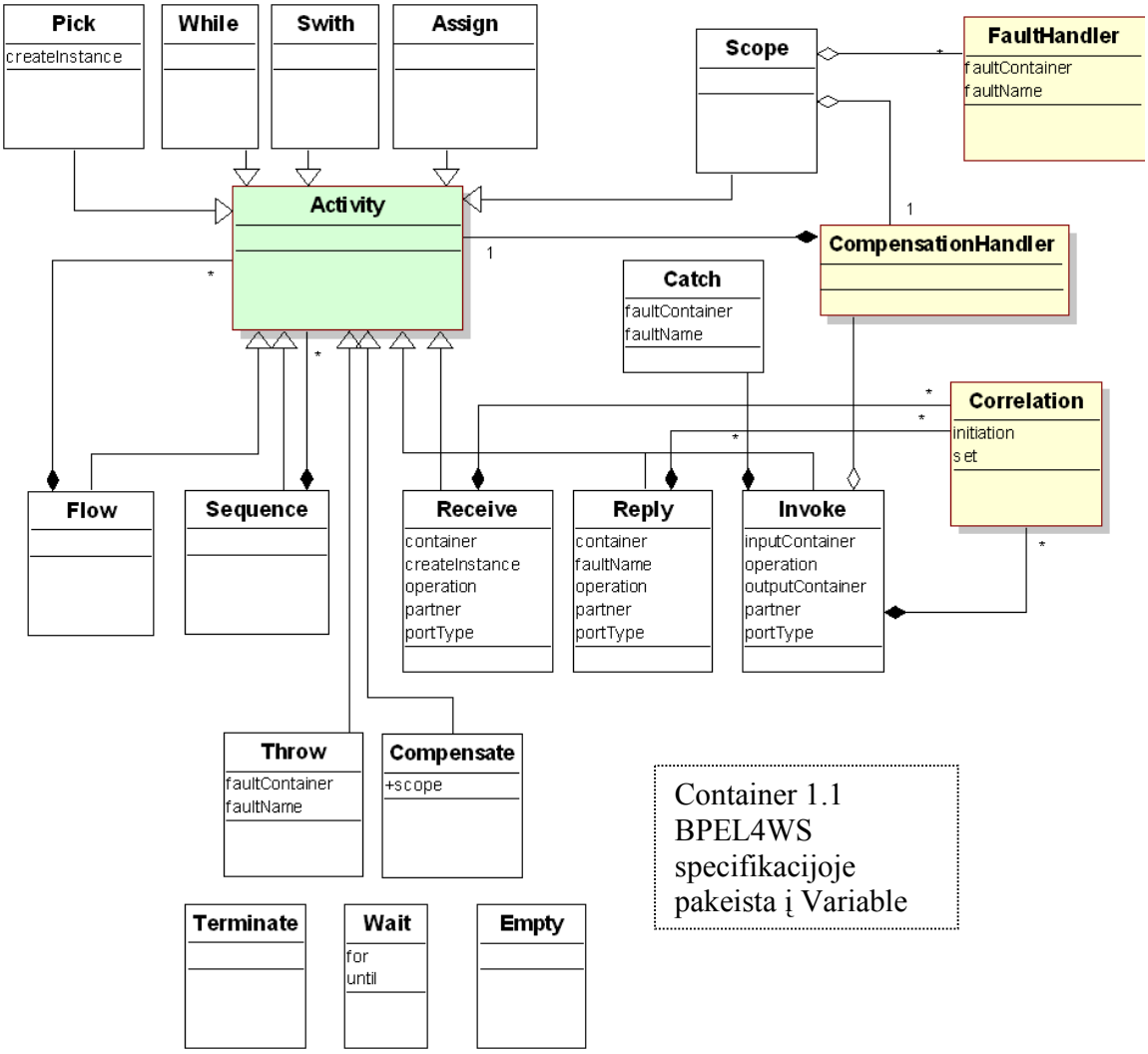
<pick>

<flow>

<scope>

<compensate>

4 pav. pavaizduoti veiklų tipai bei struktūra.



4 pav. BPEL4WS veiklų tipai ir struktūra.

Procesas gali turėti tik vieną veiklą, o veiklos gali susidėti iš kitų veiklų.

„sequence" veikla reiškia, kad šiame elemente išvardintos veiklos bus vykdomos nuosekliai. Procesas dažniausiai turi „sequence" veiklą ir joje išvardinamos kitos veiklos.

Sąveikai su kitais web servisais ir BPEL4WS procesu naudojamos trys veiklos: „receive", „invoke" ir „reply". Jos yra pagrindinės BPEL4WS veiklos. Kiekvienai veiklai nurodžius porto tipą, operacija bei partnerį apibrėžiamas konkretus kreipimasis į web servisą. „invoke" veikla naudojama kuomet procesas kreipiasi į kitus web servisus ir papildomai reikia nurodyti įėjimo bei išėjimo kintamuosius (variable, 1,0 versijoje container) kurie būtų įėjimo-išėjimo parametrai kviečiamai operacijai. Kreipimasis gali būti sinchroninis ar asinchroninis.

Biznio procesas savo paslaugas teikia per „receive" ir „reply " veiklas. „receive" atitinka įėjimo parametrus tam tikrai WSDL operacijai. Jei šis kvietimas reikalauja atsakymo, būtina sukurti „reply" veiklą, kuri atitinka proceso išėjimo parametrus.

Duomenų manipuliacijos vyksta „assign" veikloje, kurioje galima kopijuoti duomenis iš vieno kintamojo į kitą, taip pat įterpti ar konstruoti duomenis naudojant XPath išraiškas. Kintamojo savybes galima gauti parašius šią išraišką:

```
bpws:getVariableProperty ('variableName', 'propertyName')
```

Šių bazinių veiklų vykdymo eigą kontroliuoja struktūrinės veiklos, kurios apibūdina proceso struktūrą komponuojant bazines veiklas į struktūras atitinkančias valdymo šablonus, duomenų srautus, klaidų gaudymą, pranešimų koordinavimą tarp protokolo procesų.

„sequence" veikloje esančios veiklos vykdomos nuosekliai tokia tvarka kokia jos yra surašytos.

„switch" veikla atitinka tradicinių programavimo kalbų „switch" išraišką, kuomet vykdymui parenkama viena veikla iš veiklų sąrašo atitinkanti <case> elemente nurodytą XPath išraiškos sąlygą.

„while" veikla atlieka veiklas tol kol įvykdoma XPath nurodyta sąlyga.

„pick“ veikla sudaryta iš įvykių apdorojimo elementų (event handlers) kurių kiekvienas turi po vieną veiklą. Įvykus kuriam nors įvykiui bus vykdoma atitinkama veikla.

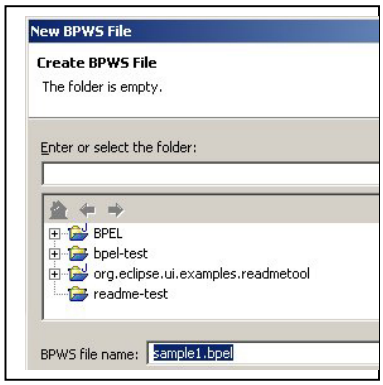
„scope“ veikla grupuoja joje esančias veiklas į tam tikrus loginius vienetus, kurie atitinkamai reaguoja į klaidas.

„flow“ tai viena iš sudėtingesnių veiklų naudojama lygiagrečiam veiklų vykdymui kontroliuoti. Kiekvienai veiklai sukuriami ryšiai su kitomis veiklomis, ant kurių galima užrašyti vykdymo sąlygas XPath išraiška (guard condition).

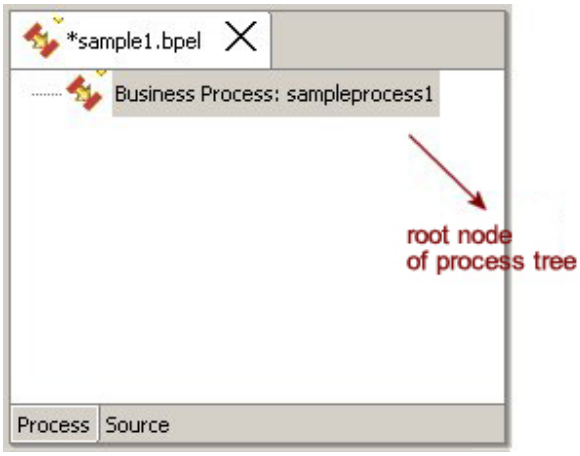
2.3.4 BPEL4WS produktai rinkoje

Sistemos funkcinė analizė atliekama Rational Rose 2000 Enterprise Edition paketu, o procesai modeliuojami su IBM Eclipse BPWS4J įskiepiu (plugin). Šis įrankis leidžia modeliuoti BPEL4WS procesus ir yra nemokamas. Jį galima parsisiųsti internetu: http://www.alphaworks.ibm.com/tech/bpws4j

Norint sukurti BPEL modeli pirmiausia sukuriamas projektas:

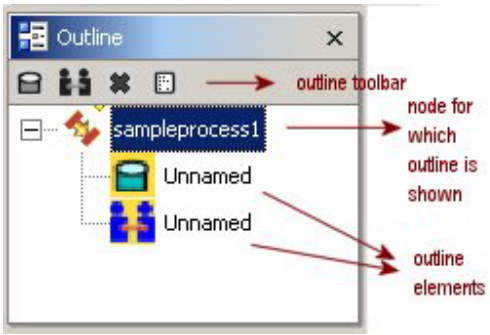


BPEL procesas modeliuojamas proceso vaizdo lange iš meniu pasirenkant veiklas, taip formuojamas proceso veiklų medis:

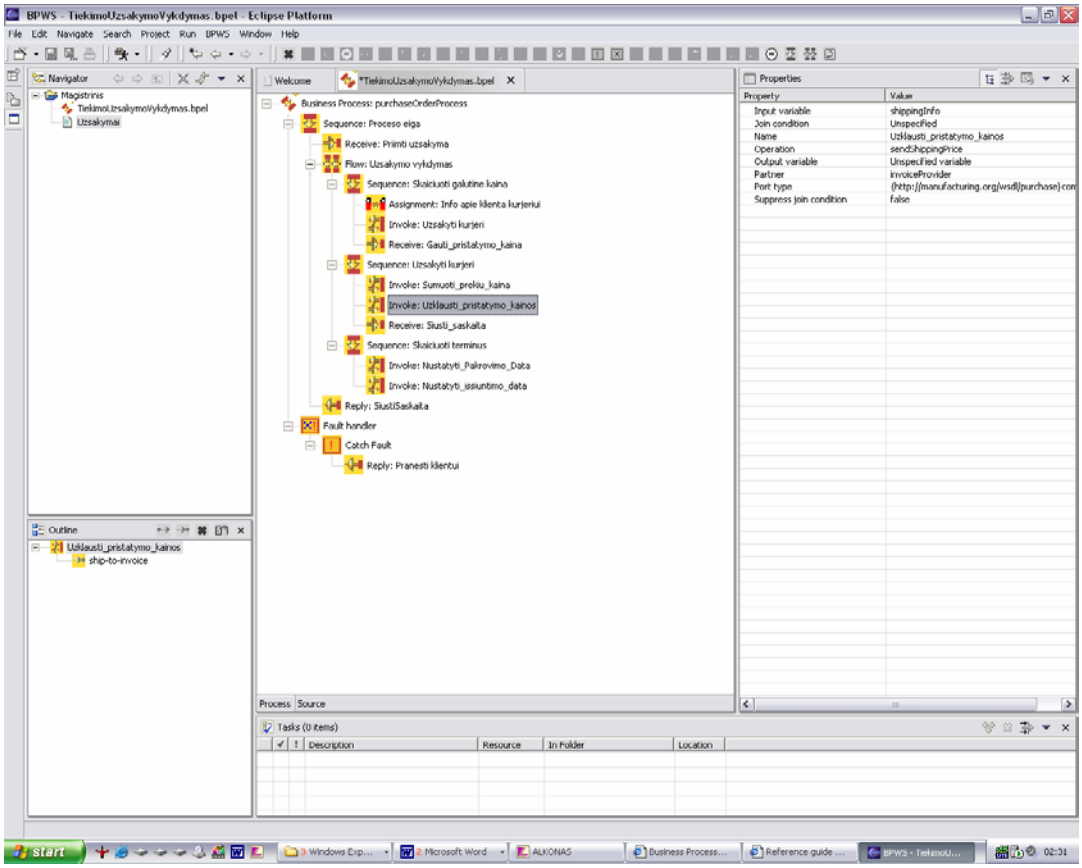


Modeliuojant BPEL procesą automatiškai generuojamas XML kodas, kurį galima pamatyti kodo lange:

„Outline“ lange matoma pasirinktos proceso dalies struktūra: veiklos dalys, proceso kintamieji, rolės ir t.t.



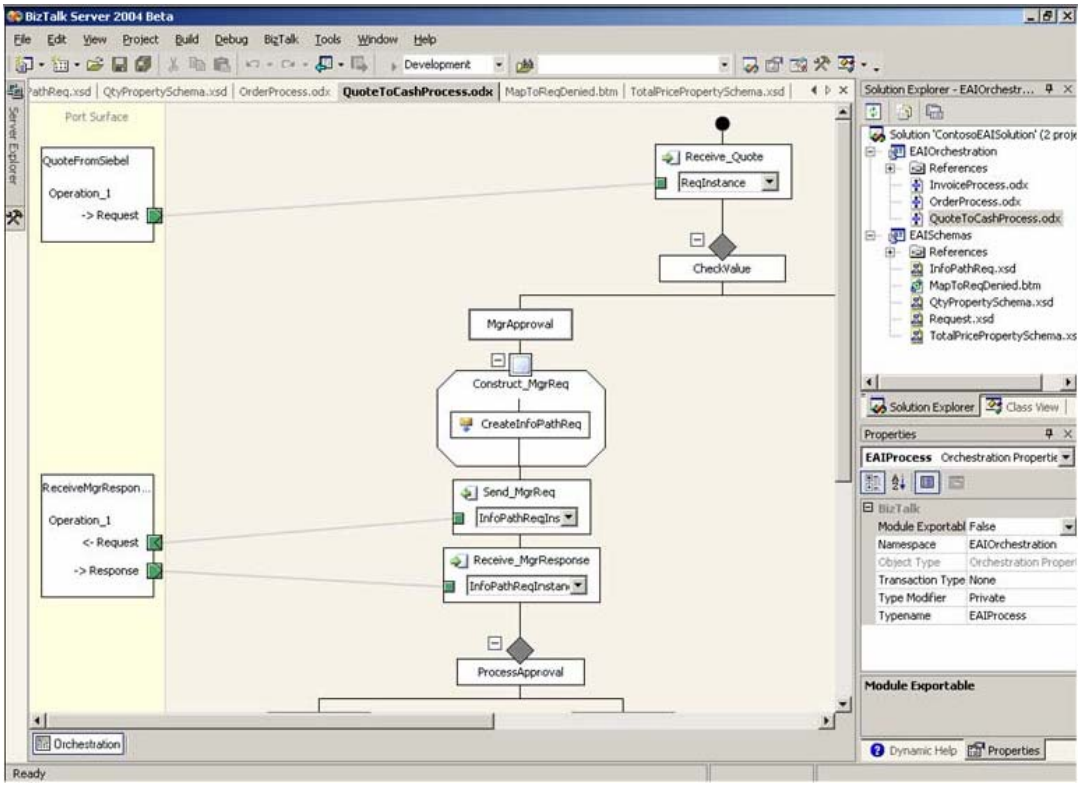
5 pav. pateiktas bendras BPWS4J redaktoriaus vaizdas.



5 pav. BPEL redaktorius Eclipse sistemopje

Komerciniai produktai:

Microsoft korporacija kuria naują BizTalk versiją, kodiniu pavadinimu „Jupiter“, kuri turėtų vadintis BizTalk 2004 ir kaip ir senosios versijos leis modeliuoti verslo procesus bei susieti jų realizacijas. Tai paprastai atliekama su grafiniu modeliavimo įrankiu, nereikalaujant didelių programavimo įgūdžių. BizTalk 2004 palaikys BPEL4WS 1.1 versiją ir bus Web servisų sąveikos serveriu t.y. leis vykdyti (execute) BPEL4WS procesus.

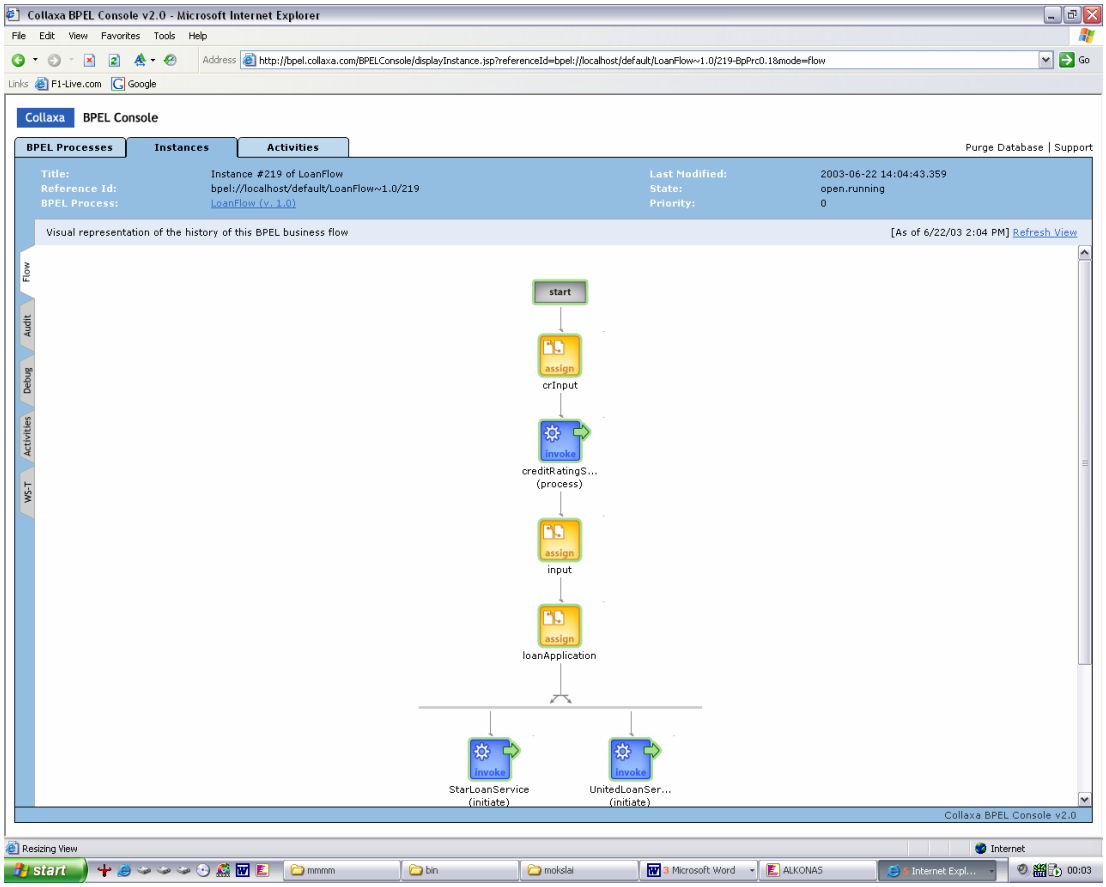


6 pav. Microsoft BizTalk 2004 Server

IBM sukurtas BPWS4J variklis BPEL4WS specifikacijai yra laisvai prieinamas internetu. Be to IBM pateikia ir anksčiau minėtą BPEL4WS modeliavimo įrankį. Taip pat BPEL4WS yra integruotas į Web Sphere sistemą.

Dauguma PĮ rinkos lyderių kuria sistemas palaikančias BPEL4WS, tačiau rinkoje kol kas jų nėra.

Viena pirmųjų BPEL4WS serverį rinką išleido Collaxa kompanija. Produktas turi puikią vartotojo sąsają, leidžia derinti (debug) BPEL4WS kodą, bei grafiškai stebėti proceso vykdymo eigą.

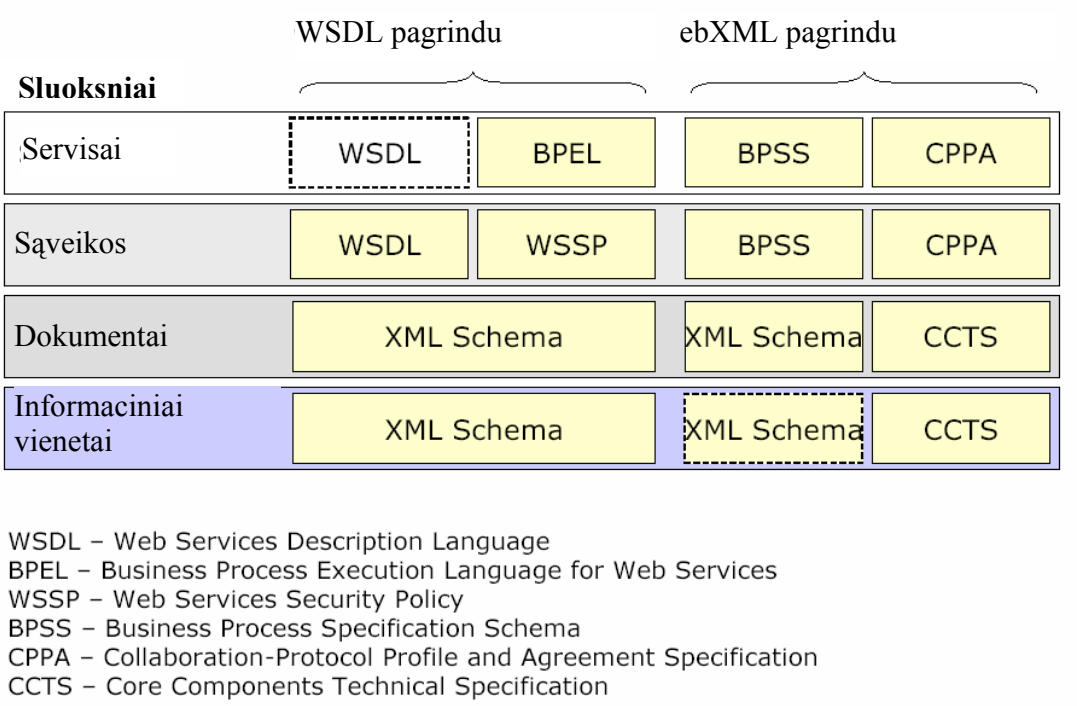


7 pav. Collaxa BPEL konsolė

2.4 Lyginamoji analizė

Rinkoje vyrauja dvi automatizuotų verslo procesų aprašymo ir jų vykdymo specifikavimui pagrindinės kryptys: veikiančios WSDL pagrindu bei ebXML. BPEL yra WSDL pagrindu veikianti kalba ir yra ryškiausia bei populiariausia savo šeimoje. Todėl palyginami bus WSDL pagrindu veikiantys metodai (turint omenyje BPEL) bei ebXML.

8 pav. pavaizduotos naudojamos technologijos kiekvienam proceso sluoksniui.



8 pav. procesu sluoksnius realizuojančios kalbos

Informaciniai vienetai (duomenų tipai) WSDL pagrindu veikiančiose kalbose apibrėžiami XML Schema, tuo tarpu ebXML „core components“ yra nepriklausomi nuo XML schemos ar kitų apibrėžimų kalbų ir teikia abstraktų net nuo proceso konteksto nepriklausomą esybių apibrėžimą. 1 lentelėje pavaizduotas tipų apibrėžimų palyginimas.

1 lentelė, duomenų tipai BPEL4WS ir ebXML

Kriterijus duomenų tipams	BPEL4WS	ebXML
Primityvūs duomenų tipai (iš anksto nustatyti)	+	+
Sudėtingi duomenų tipai	+	+
Tipų specializavimas	+	-
Verčių apribojimai	+	+
Eiliškumo apribojimai	+	-
Tipų bibliotekos	-	+

BPEL4WS Dokumentai apibrėžiami taip pat XML Schema, ebXML ir čia turi savo specifinius abstrakčius bei industrijai specializuotus būdus dokumentų apibrėžimui, taip pat nepriklausančius nuo apibrėžimą realizuojančios kalbos, tačiau naudojant reikia susieti su konkrečia kalba.

Sąveikoms apibrėžti BPEL4WS naudoja WSDL bei WS-Policy: WSDL apibrėžia sinchroninius asinchroninius pranešimų mainus bei RPC stiliaus interfeisus. WS-Policy – WS-SecurityPolicy – naudojama saugumo problemoms spręsti, transakcijoms vykdyti reikalingos WS-Transaction bei WS-Coordination specifikacijos. ebXML atskirai specifikuoja konkrečiam verslui būdingus bei nebūdingus sąveikos tipus bei teikia standartinius sąveikų šablonus. Sąveikos palygintos 2 lentelėje.

2 lentelė, sąveikų palyginimas

Kriterijus sąveikoms	BPEL4WS	ebXML
Funkcionalumo įvardinimas	+	+
Funkcionalumo prieš ir po sąlygos	-	+
Elgsena: iš anksto apibrėžti šablonai:		
Vienkryptis	+	+
Užklausimas/atsakymas	+	+
Keli sėkmingo atsakymo variantai	-	+
Skirtingi klaidų atsakymai	+	+
Konfigūruojami patvirtinimai	-	+

Transakcijų deklaravimas	-	+
Sąveikos tipų (šablonų) biblioteka	-	+

BPEL4WS servisų kompozicijai naudoja BPEL kalbą ir remiasiWSDL sąveikų apibrėžimais naudojamais specifikuoti tiek abstrakčius tiek vykdomuosius procesus. ebXML naudoja nuo verslo srities nepriklausomus servisų tipus bei nuo verslo srities priklausomus konfigūracinius šablonus. ebXML akcentuoja sąveiką tarp dveijų partnerių, kelių partnerių sąveika yra vykdoma išplečiant funkcionalumą. Servisų palyginimas pavaizduotas 3 lentelėje.

3 lentelė, servisų palyginimas

Kriterijus servisams	**BPEL4WS**	**ebXML**
Sąveikų abipusė priklausomybė	+	+/-
Kintamieji	+	-
Duomenų srautai	+	-
Vartotojo apibrėžtas pranešimų koreliavimas	+	-
Lokalinė būsena ir veiklos	+	-
Sinchronizuota dvišalė būsena, veiklos	-	+
Daugiašalė sinchronizuota būsena, veiklos	-	-
Įvykiai	+	-
Klaidų gaudymas	+	+/-
Kompensavimas	+	+/-

BPEL4WS ir ebXML procesai tiesiogiai negali būti integruojami. Iš esmės BPEL4WS skirtas realizuoti konkrečias verslo užduotis tarp skirtingų verslo partnerių paskirstytoje aplinkoje, o ebXML yra labiau verslo procesus charakterizuojantis, grupuojantis standartas.

3. Verslo procesų vaizdavimo UML ir BPEL4WS metodika

Automatizuoto verslo proceso sąvoka naudojama turint omenyje laisvai susietų programinės įrangos komponentų sąveiką, realizuojančią santykinai ilgai trunkančias verslo užduotis. BPEL4WS kalba skirta automatizuotiems verslo procesams ir jų vykdymo semantikai apibrėžti. Ji naudoja XML žymėjimus.

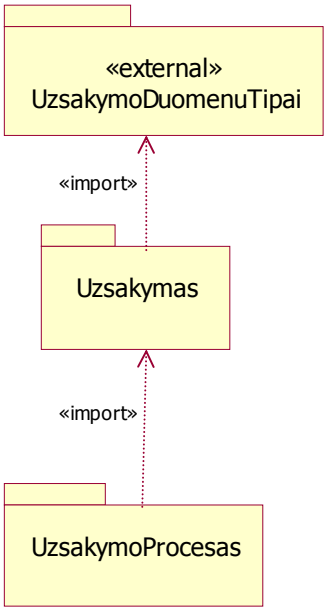
Šiame darbe nagrinėjamas automatizuoto verslo proceso pavyzdys yra pirkimo užsakymo vykdymo procesas. Nagrinėjamas procesas yra vykdomasis (angl. *executable*). Nagrinėjamas procesas, gavęs pirkimo užsakymą iš pirkėjo-tarpininko, sukuria tris lygiagrečias veiksmų sekas: galutinės kainos skaičiavimas, kurjerio parinkimas bei užsakymo įvykdymo ir pristatymo planavimas. Kai visos šios užduotys yra baigtos, siunčiama sąskaita pirkėjui.

3.1 Modelių struktūrizavimas

UML modeliai struktūrizuojami į paketus. UML paketai atitinka BPEL vardų erdves ar atitinkamus pakartotino panaudojimo vienetus. Paketai sugrupuoja tam tikrus komponentus suteikdami jiems bendrą vardų erdvę. Priklausomybės tarp paketų vaizduojamos <<import>> tipo priklausomybėmis.

Modelio paketų hierarchija atitinka XML vardų laukų hierarchiją. Modelio vardas atitinka aukščiausio paketo vardą ir atitinkamai aukščiausio lygmens paketo vardą. Generuojant kodą iš modelio sukuriami atitinkami failai tiems paketams kurie siejasi su XSD ar WSDL elementais. Protokolų paketams nauji failai nekuriami, o informacija talpinama WSDL faile atitinkančiame viršutinį paketą. BPEL procesas rašomas į atskirą failą. 9 pav. pavaizduota nagrinėjamo proceso paketų hierarchija.

Paketas „Uzsakymas“ bus sugeneruotas į failą Uzsakymas.wsdl, o „UzsakymoProcesas“ atitinka vardų lauko pavadinimą faile PirkimoUzsakymas.bpel.



9 pav. Užsakymo proceso modelio paketų hierarchija

10 pav. pavaizduotas sugeneruoto Uzsakymas.wsdl failo fragmentas iliustruojantis <<import>> priklausomybės tarp paketų realizavimą į vardų laukų importavimą.

```
<wsdl:definitions

targetNamespace="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Uzsa
kymas/"

xmlns:Uzsakymas="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Uzsa
kymas/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:UzsakymoDuomenuTipai="http://localhost/Tiekejas/PirkimoUzsakymoPr
ocesas/UzsakymoDuomenuTipai/">
    <wsdl:import
namespace="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDu
omenuTipai/"
```

```
location="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuo
menuTipai.wsdl"/>
..............
```

10 pav. WSDL vardų laukų deklaravimo fragmentas

Modelis gali būti priklausomas nuo elementų, kurie apibrėžiami už modelio ribų. Tokie elementai vadinami išoriniais. Išorinių paketų komponentai yra apibrėžiami kitame modelyje arba yra specifiniai nuo platformos priklausomi elementai. Išorinius paketus labai patogu naudoti vartotojo duomenų tipų sukūrimui. Nagrinėjamame procese išorinis paketas „UzsakymoDuomenuTipai“ turi stereotipą <<external>> reiškiantį, kad paketo elementai apibrėžti ne modelyje. Šiuo atveju generuojamas UzsakymoDuomenuTipai.xsd failas su reikalingų tipų vardais, o jų apibrėžimas paliekamas vartotojui. Tai suteikia modeliui lankstumo. Užsakymo proceso duomenų tipų deklaracijos pavaizduotos 11 pav.

```
<?xml version="1.0"?>
<xsd:schema

targetNamespace="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Uzsa
kymoDuomenuTipai/"

xmlns:UzsakymoDuomenuTipai="http://localhost/Tiekejas/PirkimoUzsakymoPr
ocesas/UzsakymoDuomenuTipai/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:complexType name="PirkimoUzsakymas"/>
    <xsd:element name="PirkimoUzsakymas" type="
UzsakymoDuomenuTipai:PirkimoUzsakymas"/>
    <xsd:complexType name="KlientoInfo"/>
    <xsd:element name="KlientoInfo" type="
UzsakymoDuomenuTipai:KlientoInfo"/>
    <xsd:complexType name="PristatymoInfo"/>
    <xsd:element name=" PristatymoInfo " type=" UzsakymoDuomenuTipai:
PristatymoInfo "/>
    <xsd:complexType name="Saskaita"/>
    <xsd:element name=" Saskaita " type="UzsakymoDuomenuTipai: Saskaita
"/>
    <xsd:complexType name="GrafikoInfo"/>
```

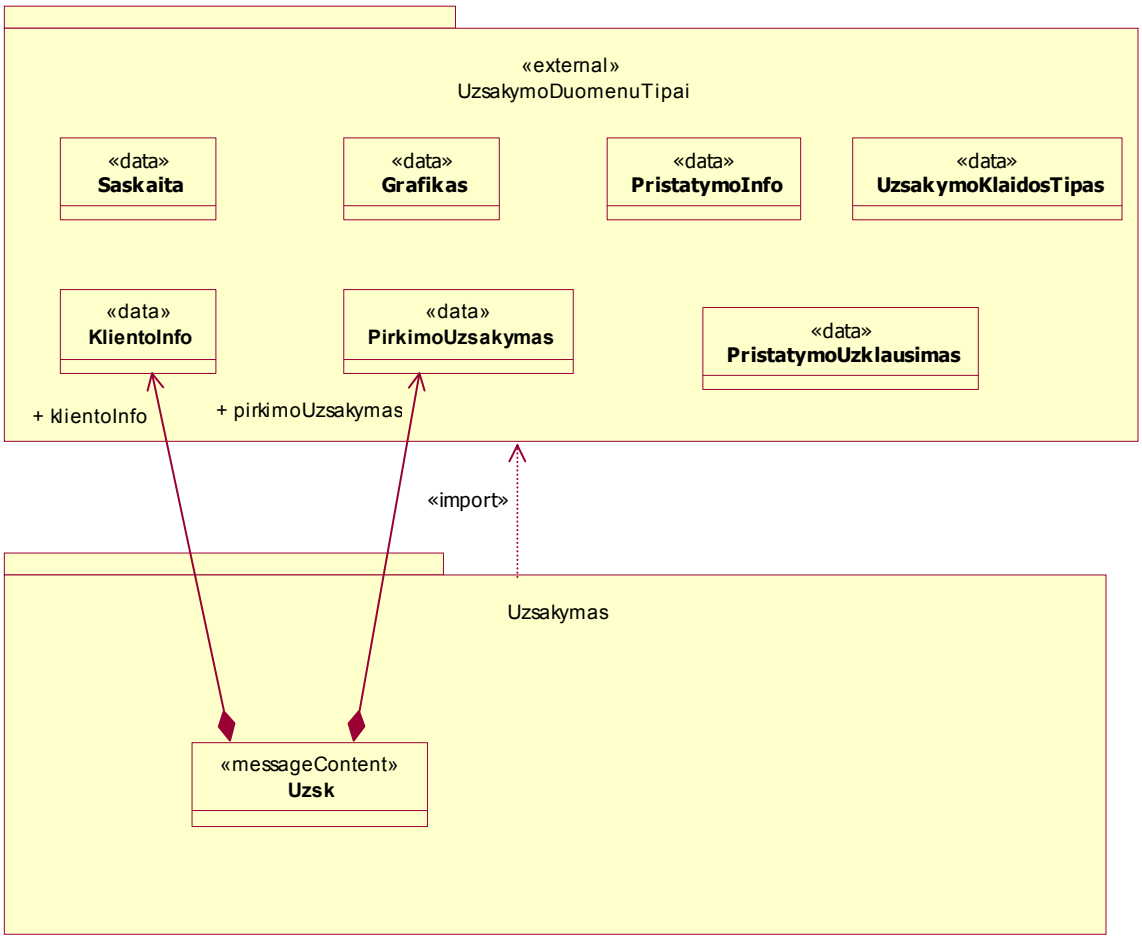
```
    <xsd:element name=" GrafikoInfo " type="UzsakymoDuomenuTipai:
GrafikoInfo "/>
</xsd:schema>
```

11 pav. „UzsakymoDuomenuTipai“ tipų deklaracijos

3.2 Web servisų vaizdavimas

Automatizuotų biznio procesų sritis remiasi Web servisų srities koncepcijomis bei infrastruktūra. Esminis automatizuotų procesų koncepcijos elementas yra tiesioginis (peer-to-peer) komunikavimas tarp Web servisų, kur ir procesas ir partneriai yra apibrėžti kaip atitinkami Web servisai.

Web servisų terminologijoje portų tipai apibrėžia susijusių operacijų aibę. Portų tipai UML diagramose vaizduojami kaip interfeisai. Operacijoms perduodami parametrai, kurių tipą nusako atitinkamo pranešimo tipas. Pranešimo tipas savo ruožtu turi kelias dalis, kurių tipas apibrėžtas duomenų tipu. Duomenų tipai modeliuojami kaip klasės su stereotipu <<data>>; pranešimų tipai modeliuojami kaip klasės su stereotipu <<messageContent>>. Pranešimų tipų modeliuoti nebūtina, jei jie susideda tik iš vienos dalies, tuomet tiesiog naudojamas duomenų tipas. Duomenų tipus derėtų kurti tik tuomet, kai jie prasmingai sugrupuoja keletą atributų ir naudojami ne tik vienoje operacijoje. Pranešimų tipus pravartu kurti kai grupavimas turi prasmę tik operacijos kontekste. <<data>> klasių atributų tipai gali būti baziniai arba sukurti vartotojo. 12 pav. pavaizduoti nagrinėjamo procese duomenų tipai, kurių atributai yra bazinių tipų. „Uzsk“ pranešimo tipas sudarytas iš dviejų vartotojo apibrėžtų dalių. <<data>> klasių atributai tiesiogiai siejami su XSD schemų atributais.



12 pav. Užsakymo proceso duomenų tipai

13 pav. pavaizduoti pranešimų tipų WSDL aprašai gauti generuojant kodą iš modelio

```
<wsdl:message name="Uzsk">
    <wsdl:part name="pirkimoUzsakymas"
type="UzsakymoDuomenuTipai:PirkimoUzsakymas"/>
    <wsdl:part name="klientoInfo"
type="UzsakymoDuomenuTipai:KlientoInfo"/>
</wsdl:message>

<wsdl:message name="PristatymoInfo">
    <wsdl:part name="part" type="
UzsakymoDuomenuTipai:PristatymoInfo"/>
</wsdl:message>
```

```
<wsdl:message name="Saskaita">
    <wsdl:part name="part" type=" UzsakymoDuomenuTipai:Saskaita"/>
</wsdl:message>

<wsdl:message name="UzsakymoKlaida">
    <wsdl:part name="problemosInfo" type="xsd:string"/>
</wsdl:message>

<wsdl:message name="GrafikoInfo">
    <wsdl:part name="part" type=" UzsakymoDuomenuTipai:GrafikoInfo"/>
</wsdl:message>

<wsdl:message name="PristatymoUzklausimas">
    <wsdl:part name="klientoInfo" type="
UzsakymoDuomenuTipai:KlientoInfo"/>
</wsdl:message>
```

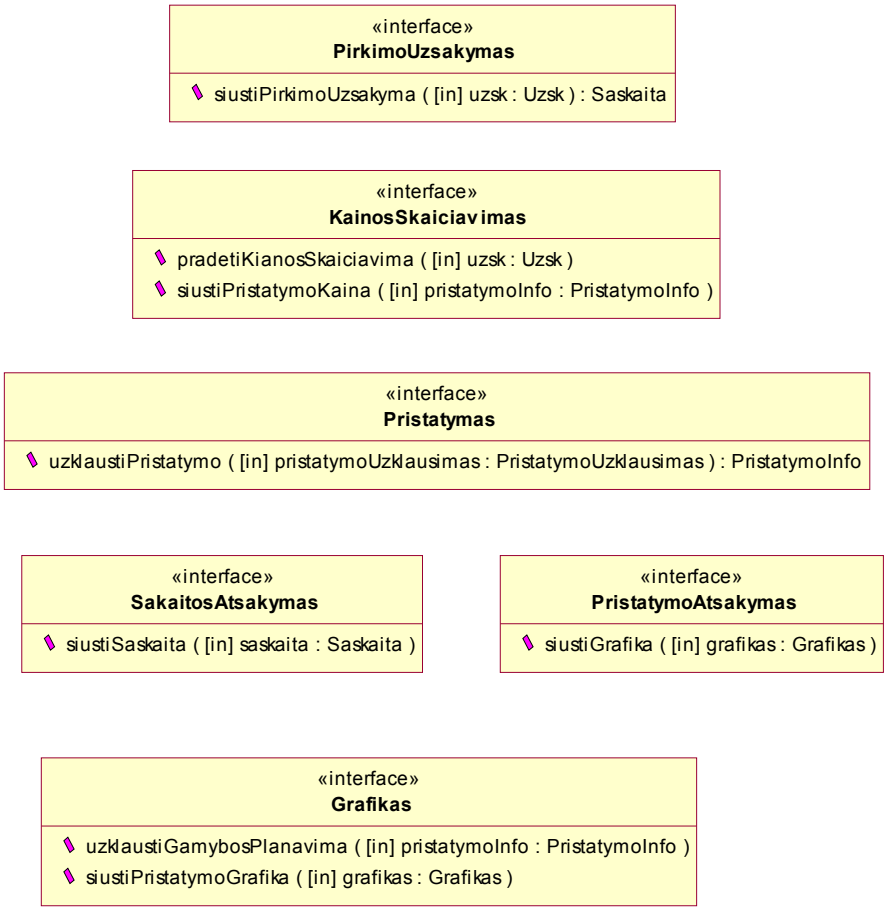
13 pav. WSDL operacijų pranešimai

Kiekvienas paketas, turintis interfeisų, yra susiejamas su atskiru WSDL dokumentu. Kiekvienas paketo interfeisas yra susiejamas su WSDL porto tipu atitinkamame WSDL dokumente. Kiekviena klasė su <<data>> stereotipu yra susiejama su XSD tipu, o kiekviena <<messageContent>> klasė siejama su WSDL pranešimo tipu. WSDL draudžia tiesiogiai naudoti XSD tipus kaip operacijų parametrus. Tam tikslui yra sugeneruojami pranešimų tipai su viena dalimi iš tų <<data>> klasių, kurios yra panaudotos kaip operacijų parametrai. Tačiau jei reikalingas skirtingas tipų panaudojimas, būtina sukurti skirtingus pranešimus, nes bus generuojamas tik vienas pranešimo tipas.

Procesas dalyvauja protokole per kiekvieną iš savo portų. Protokolas susieja porą rolių ir vaizduoja jų sąveiką. Kiekviena rolė gali turėti kelis interfeisus reikalingus kitai rolei. 12 pav. pavaizduota UML klasių diagrama, vaizduojanti proceso duomenų tipus, išreikštus UML klasėmis, kurios yra naudojamos kaip operacijų parametrų tipai. Operacijas turi arba jos yra iškviečiamos per rolių interfeisus.

Užsakymo pakete taip pat yra ir užsakymo procesui reikalingi ar jo teikiami interfeisai, jie pavaizduoti 15 pav. Klaidų metimui naudojami operacijų [out]

parametrai, su stereotipu <<fault>>, tačiau ši Rational Rose versija tinkamai nepalaiko šios galimybės, todėl modelyje to pavaizduoti neįmanoma.



15 pav. užsakymo proceso naudojami interfeisai

16 pav. Iš „Uzsakymas“ paketo atitinkamai sugeneruotame Uzsakymas.wsdl faile bus sukurti šie portų tipai atitinkantys pavaizduotus interfeisus su jų operacijomis:

```
<wsdl:portType name="KainosSkaiciavimas">
        <wsdl:operation name="pradetiKainosSkaiciavima">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:Uzsk"/>
        </wsdl:operation>
        <wsdl:operation name="siustiPristatymoKaina">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:PristatymoInfo"/>
        </wsdl:operation>
```

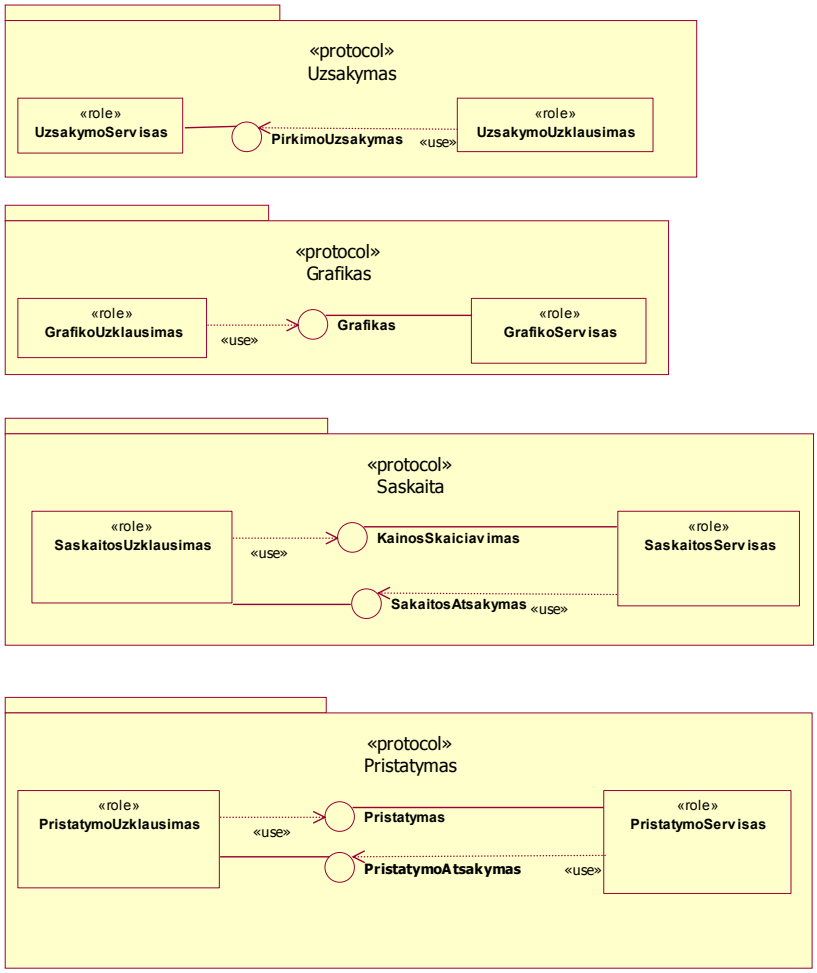
```
    </wsdl:portType>

    <wsdl:portType name="SaskaitosAtsakymas">
        <wsdl:operation name="siustiSaskaita">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:Saskaita"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:portType name="PirkimoUzsakymas">
        <wsdl:operation name="siustiPirkimoUzsakyma">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:Uzsk"/>
            <wsdl:output message="UzsakymoDuomenuTipai:Saskaita"/>
            <wsdl:fault name="cannotCompleteOrder" message =
"UzsakymoDuomenuTipai:UzsakymoKlaida"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:portType name="Grafikas">
        <wsdl:operation name="requestProductionScheduling">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:PO"/>
        </wsdl:operation>
        <wsdl:operation name="sendShippingSchedule">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:ScheduleInfoMessage"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:portType name="Pristatymas">
        <wsdl:operation name="requestShipping">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:ShippingRequest"/>
            <wsdl:output
message="UzsakymoDuomenuTipai:ShippingInfoMessage"/>
            <wsdl:fault name="cannotCompleteOrder" message =
"UzsakymoDuomenuTipai:OrderFault"/>
        </wsdl:operation>
    </wsdl:portType>
```

```
    <wsdl:portType name="PristatymoAtsakymas">
        <wsdl:operation name="sendSchedule">
            <wsdl:input name="input"
message="UzsakymoDuomenuTipai:ScheduleInfoMessage"/>
        </wsdl:operation>
    </wsdl:portType>
```

16 pav. „Uzsakymas“ paketo WSDL

17 pav. pavaizduoti protokolai, kuriuose pirkimo užsakymo procesas atliks tam tikras roles ir sąveikaus su partneriais atliekančiais savas atitinkamas roles. Užsakymo proceso atliekamos rolės pavaizduotos kairiau. Kiekvienas protokolas atitinka paketo „Uzsakymas“ atitinkamą subpaketą, į kurį talpinamos protokolo rolės. Paketas „Uzsakymas“ sudarytas ir iš protokolų naudojamų klasių bei interfeisų.

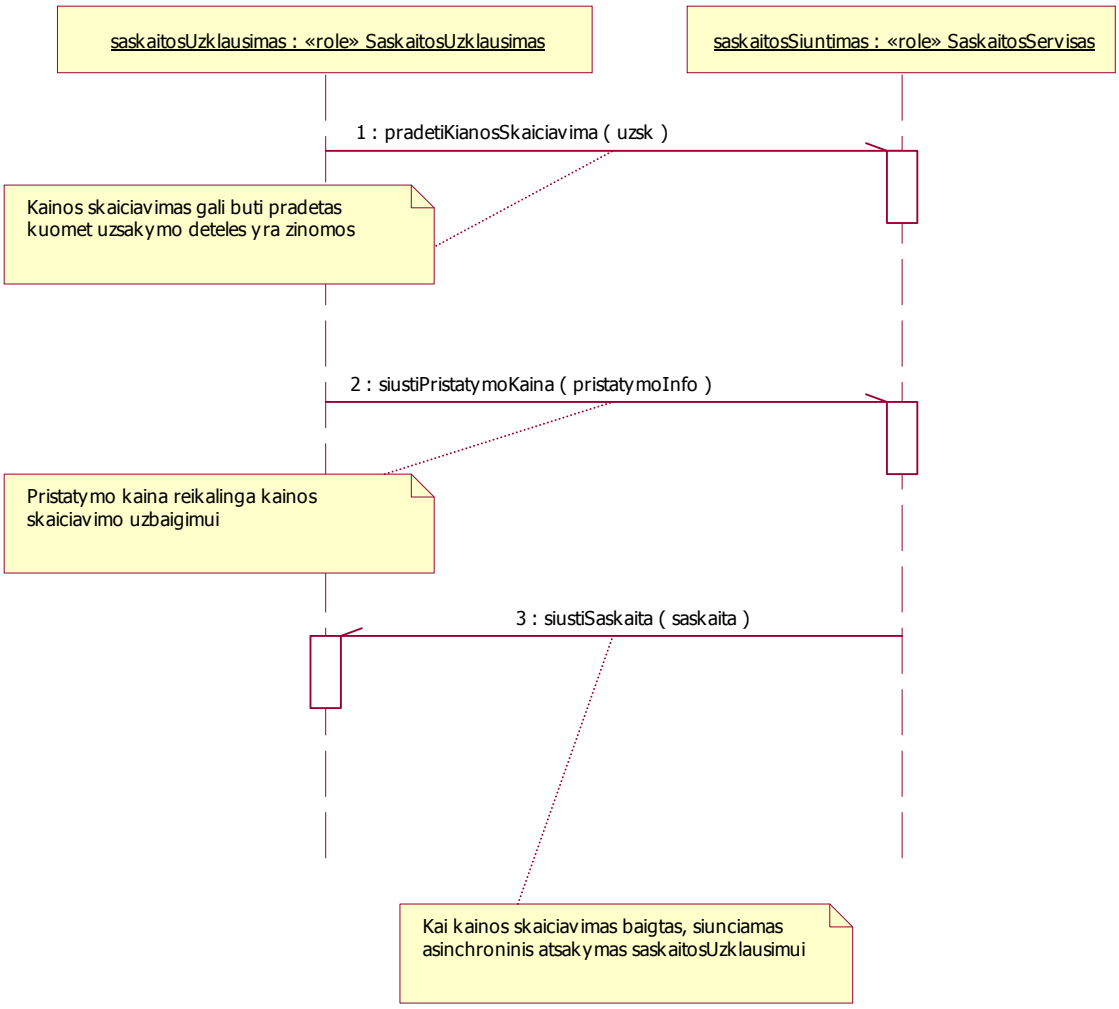
Protokolai „Uzsakymas“ bei „Grafikas“ turi tik po vieną rolę teikiančią interfeisą; kita rolė apibūdina serviso užklausimą. Protokolas „Saskaita“ turi „SaskaitosServisas“ rolę, kuri teikia „KainosSkaiciavimas“ interfeisą ir reikalauja, kad sąveikaujanti rolė teiktų „SaskaitosAtsakymas“ interfeisą būtiną asinchroniniam atsakymo pranešimui nusiųsti. Analogiška situacija protokole „Pristatymas“.



17 pav. pirkimo užsakymo protokolai bei rolės

Taigi, ryšiui tarp biznio proceso ir partnerio realizuoti gali prireikti vienpusio ar abipusio pranešimų siuntimo. Protokolas susieja dvi viena kitą papildančias roles ir specifikuoja interfeisus, kuriuos teikia ar reikalauja kiekvieną rolė.

Rolių struktūros modeliavimas nėra būtinas, tačiau dėl aiškumo kartais naudinga pavaizduoti teisingas pranešimų sekas. Sąveikų diagramos puikiai tinka tokiam atvaizdavimui. 18 pav. pavaizduota sekų diagrama detalizuojanti „Saskaita“ protokolą. Šiuo atveju tėra viena teisinga pranešimų seka, todėl vienos diagramos pakanka. Analogiška pranešimų seka vykdoma protokole „Pristatymas“.



18 pav. Protokolo „Saskaita“ pranešimų sekų diagrama

Kiekvienas protokolo paketas UML modelyje susiejamas su BPEL serviso ryšio tipu. Protokolo rolės tampa serviso ryšio tipo rolėmis, o jų interfeisai tampa serviso ryšio tipo rolių portų tipais. 19 pav. pavaizduoti sugeneruoti servisų ryšio tipai

```
<slnk:serviceLinkType name="PirkimoUzsakymoProcesas:Saskaita">
    <slnk:role name="SaskaitosServisas">
        <slnk:portType name="Uzsakymas:KainosSkaiciavimas"/>
    </slnk:role>
    <slnk:role name="SaskaitosUzklausimas">
        <slnk:portType name="Uzsakymas:SaskaitosAtsakymas"/>
    </slnk:role>
</slnk:serviceLinkType>
```

```
<slnk:serviceLinkType name="PirkimoUzsakymoProcesas:Uzsakymas">
    <slnk:role name="UzsakymoServisas">
        <slnk:portType name="Uzsakymas:PirkimoUzsakymas"/>
    </slnk:role>
</slnk:serviceLinkType>

<slnk:serviceLinkType name="PirkimoUzsakymoProcesas:Grafikas">
    <slnk:role name="Grafikas">
        <slnk:portType name=" Uzsakymas:GrafikoServisas"/>
    </slnk:role>
</slnk:serviceLinkType>

<slnk:serviceLinkType name="PirkimoUzsakymoProcesas:Pristatymas">
    <slnk:role name="PristatymoServisas">
        <slnk:portType name=" Uzsakymas:Pristatymas"/>
    </slnk:role>
    <slnk:role name="PristatymoUzklausimas">
        <slnk:portType name=" Uzsakymas:PristatymoAtsakymas"/>
    </slnk:role>
</slnk:serviceLinkType>
```

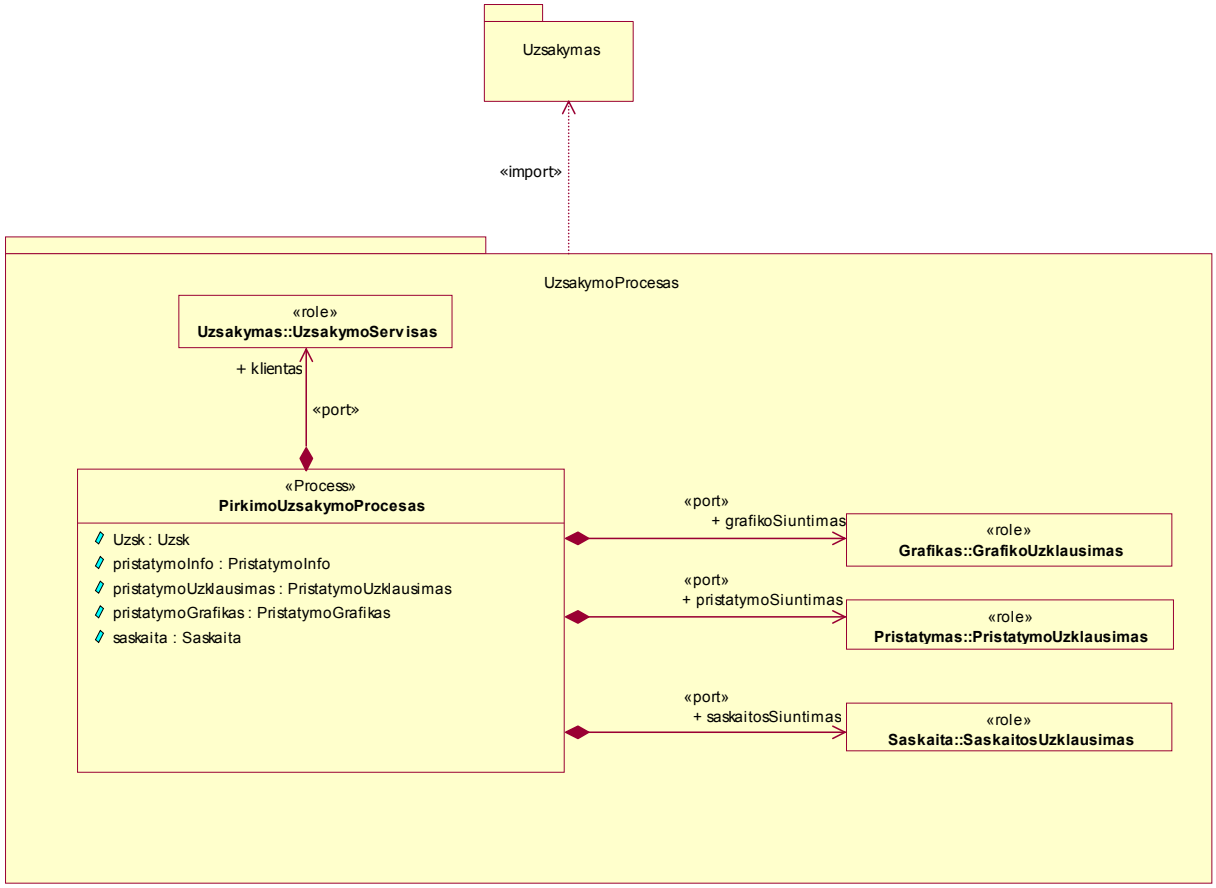
19 pav. Užsakymo proceso servisų ryšių tipai

3.3 Procesų vaizdavimas

Automatizuotas biznio procesas apibūdina kelių partnerių atliekamas su verslu susijusias užduotis. Procesas išlaiko būseną sąveikaudamas su keliais partneriais per apibrėžtus protokolus. Procesas gali sąveikauti per kelis protokolus su skirtingais partneriais, bei per tą patį protokolą su keliais partneriais.

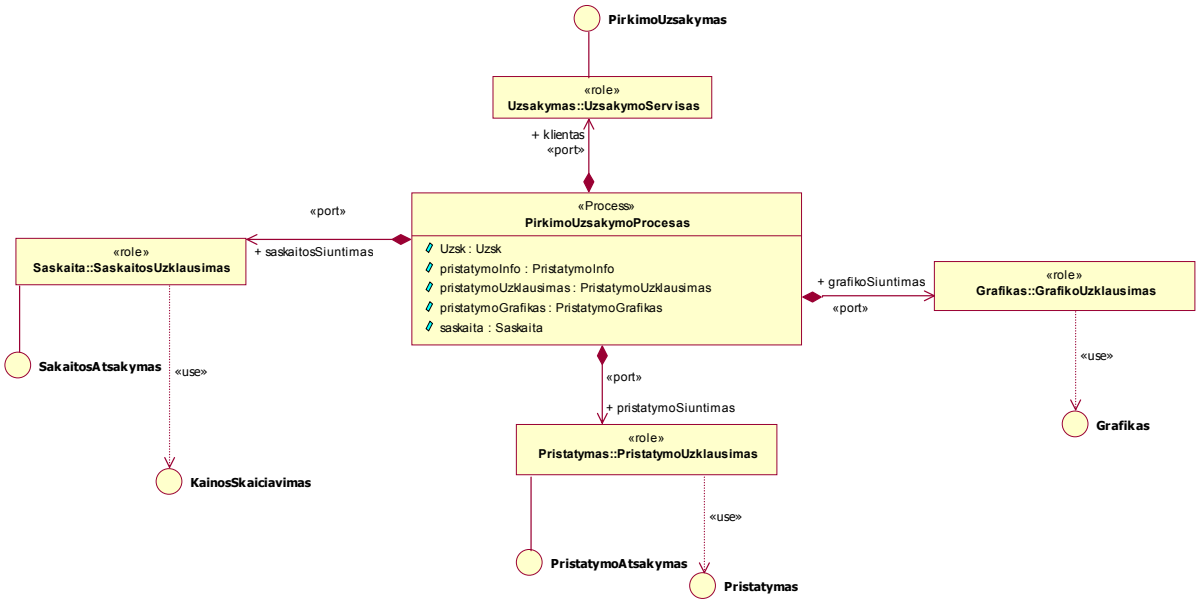
20 pav. vaizduoja klasę „PirkimoUzsakymoProcesas“ su stereotipu <<process>>. Ši klasė atvaizduoja pirkimo užsakymo procesą. Procesas turi keturis portus, kurie apibrėžti ankstesniuose modeliuose. Portai atitinka keturis partnerius su kuriais sąveikauja procesas. Kiekvienas portas yra atitinkamo partnerio ryšio taškas su procesu. Per portą teikiami ar reikalaujami interfeisai yra apibrėžiami role, kurią procesas vykdo protokole. Portas vaizduojamas agregavimo ryšiu su stereotipu <<port>> nukreiptu į rolę, kurią vykdo procesas. Proceso būsena atvaizduota kaip

„PirkimoUzsakymoProcesas“ klasės atributai, kurių tipai nurodyti pakete „Uzsakymas“ ar importuoti iš paketo „UzsakymoDuomenuTipai“.



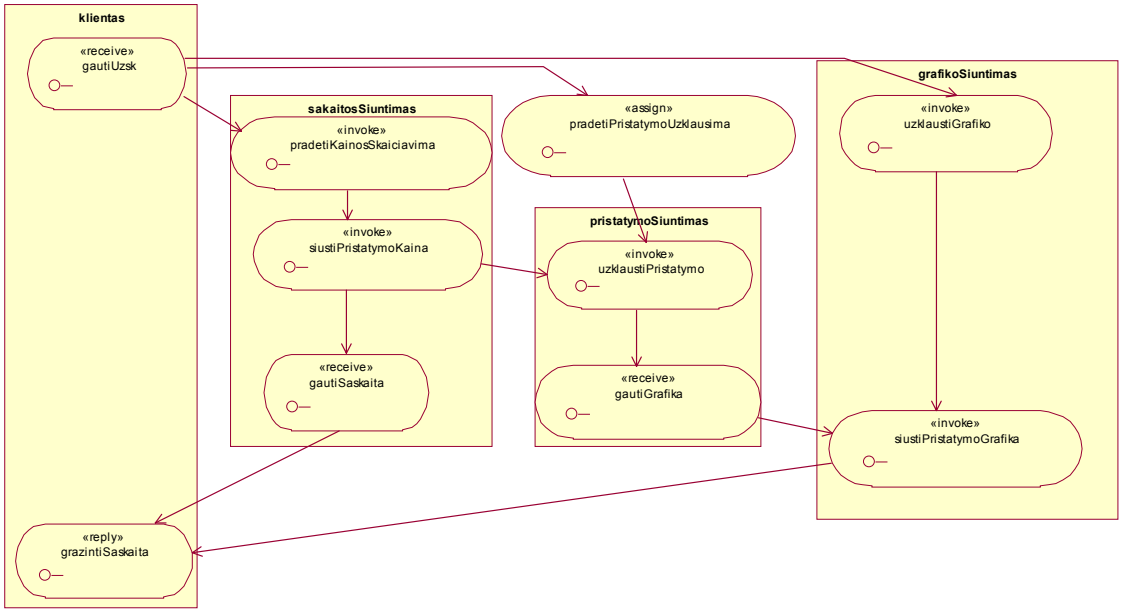
20 pav. Užsakymo proceso struktūrinis modelis

Vaizduojant portus, rolės gali būti „išskleistos“ pavaizduoti interfeisams, kuriuos jos teikia ar reikalauja. Ši informacija pateikiama protokolų modeliuose, tačiau patogiau yra turėti viską viename modelyje kaip pavaizduota 21 pav.



21 pav. Proceso struktūros diagrama su interfeisais.

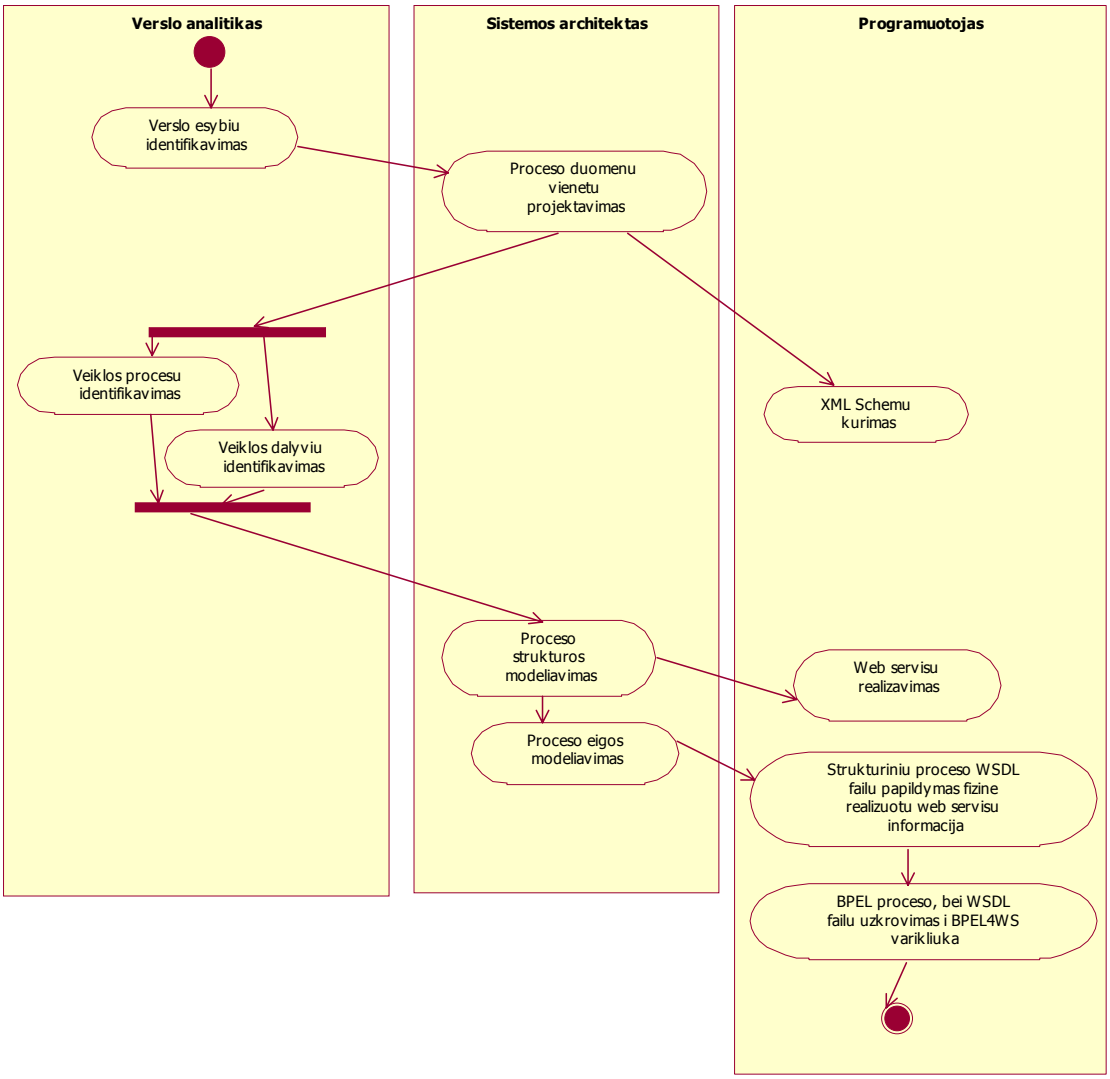
22 paveiksle pavaizduota užsakymo proceso UML veiklos diagrama vaizduoja nagrinėjamo BPEL proceso portus, atitinkančius veiklos diagramos juostas (angl. *swimlane*). Veiklos, kurios apima pranešimų siuntimą ar gavimą partnerių Web servisams per kurį nors iš 4 portų (klientas, saskaitosSiuntimas, pristatymoSiuntimas, grafikoSiuntimas) yra pavaizduotos atitinkamoje porto juostoje. Rodyklės vaizduoja proceso veiklų vykdymo seką.



22 pav. Užsakymo vykdymo proceso UML veiklų diagrama

3.4 UML modelio sudarymo metodika

Ankstesniuose šio skyriaus paragrafuose nagrinėti BPEL modelio komponentų atvaizdavimo būdai UML diagramomis. Šiame paragrafe nagrinėjama modelio sudarymo metodika. 23 pav. pavaizduota proceso realizavimo veiklų diagrama. Diagramoje yra trys veiklų juostos atitinkančios proceso sudarymui reikalingas roles: verslo analitikas, sistemos projektuotojas bei programuotojas. Proceso kūrimui veiklos realiai yra vykdomos nuosekliai (reikalavimų išgavimas, projektavimas, realizavimas), tačiau diagramoje siekta atvaizduoti projektavimo etapo prasmę verslo srities bei realizavimo atžvilgiu.



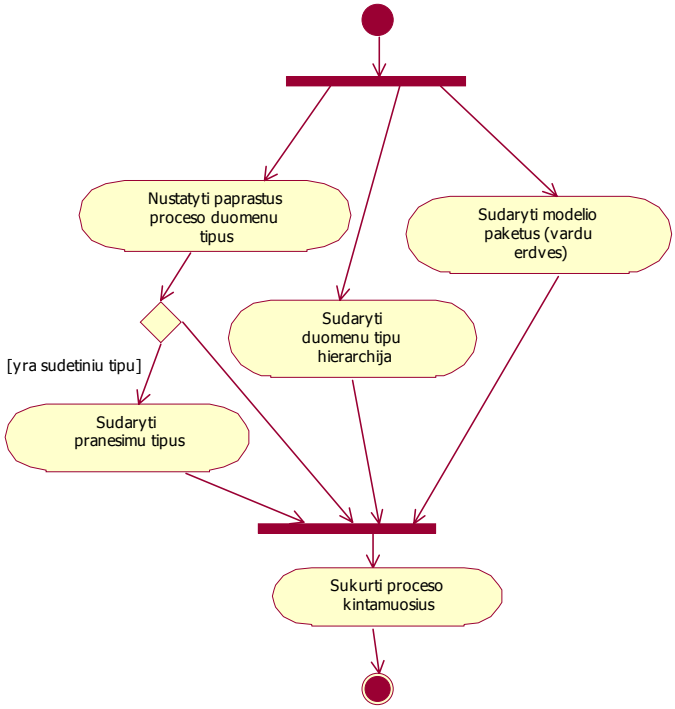
23 pav. BPEL4WS proceso modeliavimo bei realizavimo metodikos veiklų diagrama

Verslo analitikas yra dalykinės srities ekspertas ir teikia žinias apie proceso duomenų tipus, dalyvius, vykdomas užduotis.

Sistemos projektuotojas, kuria proceso modelį, pagal kurį po to kiekvienas iš partnerių realizuoja savo web servisus, ar esamus pritaiko prie proceso interfeisų.

Verslo analitikui identifikavus verslo esybes, sistemos projektuotojas jau gali projektuoti BPEL4WS proceso duomenų struktūras. Detalizuota duomenų vienetų kūrimo veiklų diagrama pavaizduota 24 pav. Kuriami paprasti duomenų tipai, t.y. jei jie nėra sudėtiniai, tai jų atskirai modeliuoti nereikia, nes WSDL operacijų parametrai

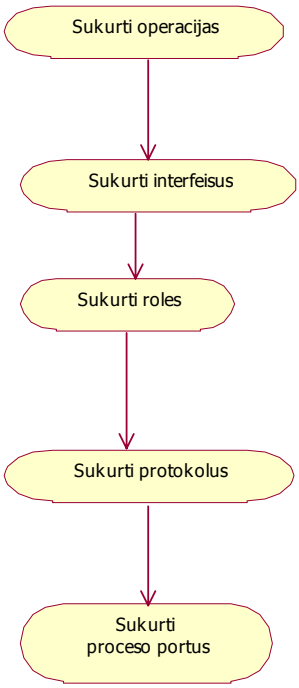
sugeneruojami automatiškai. Sukurti tipai priskiriami vardų erdvei. Proceso būsenai išlaikyti sukuriami kintamieji.



24 pav. detalizuota duomenų vienetų sudarymo diagrama

Suprojektavus duomenų struktūrą programuotojas gali kurti XML Schemas realiems duomenų tipams.

Verslo analitikas pateikia duomenis apie proceso dalyvius (partnerius) bei kokias užduotis jie vykdo. Iš šios informacijos kuriama proceso struktūra. 25 pav. pavaizduota proceso struktūros sudarymo veiklų seka. Pirmiausia nustatomos operacijos, jų įėjimo bei išėjimo parametrai, vėliau operacijos grupuojamos į interfeisus, atitinkančius WSDL portų tipus. Interfeisai toliau grupuojami roles. Rolės naudojamos protokolams kurti, kurie atitinka BPEL servisų ryšio tipus. Toliau procesui priskiriant atitinkamas roles sukuriami portai – BPEL proceso sąveikos su kitais proceso partneriais taškus.



25 pav. proceso struktūros sudarymo veiklų seka.

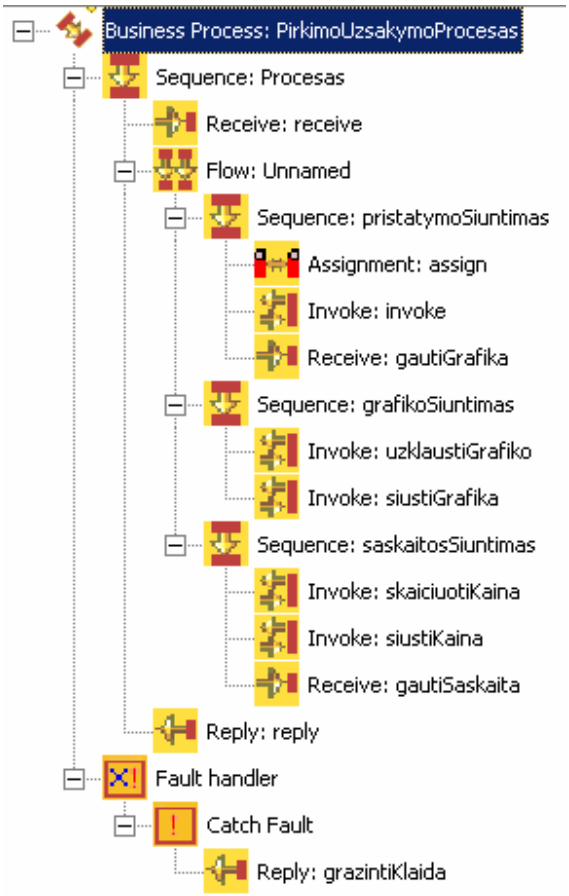
Sukūrus proceso struktūrą, programuotojas turi pakankamai informacijos apie web servisų interfeisus ir gali realizuoti verslo funkcionalumą.

Toliau kuriamos veiklų sekos – BPEL veiklų vykdymo tvarka. Nubraižius proceso veiklų diagramą, procesas yra pilnai sumodeliuotas ir programuotojas gali papildyti WSDL aprašus <bind> bei <service> elementais, charakterizuojančiais realią serviso vietą ir “įkrauti” juos kartu su proceso .bpel failu į BPEL4WS varikliuką.

4. Eksperimentinio B2B proceso projektas ir jo realizacija

4.1. Užsakymo proceso veiklų BPEL modelis BPWS4J modulyje

3-ame skyriuje nagrinėtas pirkimo užsakymo proceso modelio sudarymas. Naudojant nagrinėtą metodiką sukurtas proceso modelis. 26 pav. pavaizduotas proceso veiklų modelis IBM Eclipse platformoje naudojant BPWS4J modulį.



26 pav. BPWS4J modulyje pavaizduota pirkimo užsakymo diagrama.

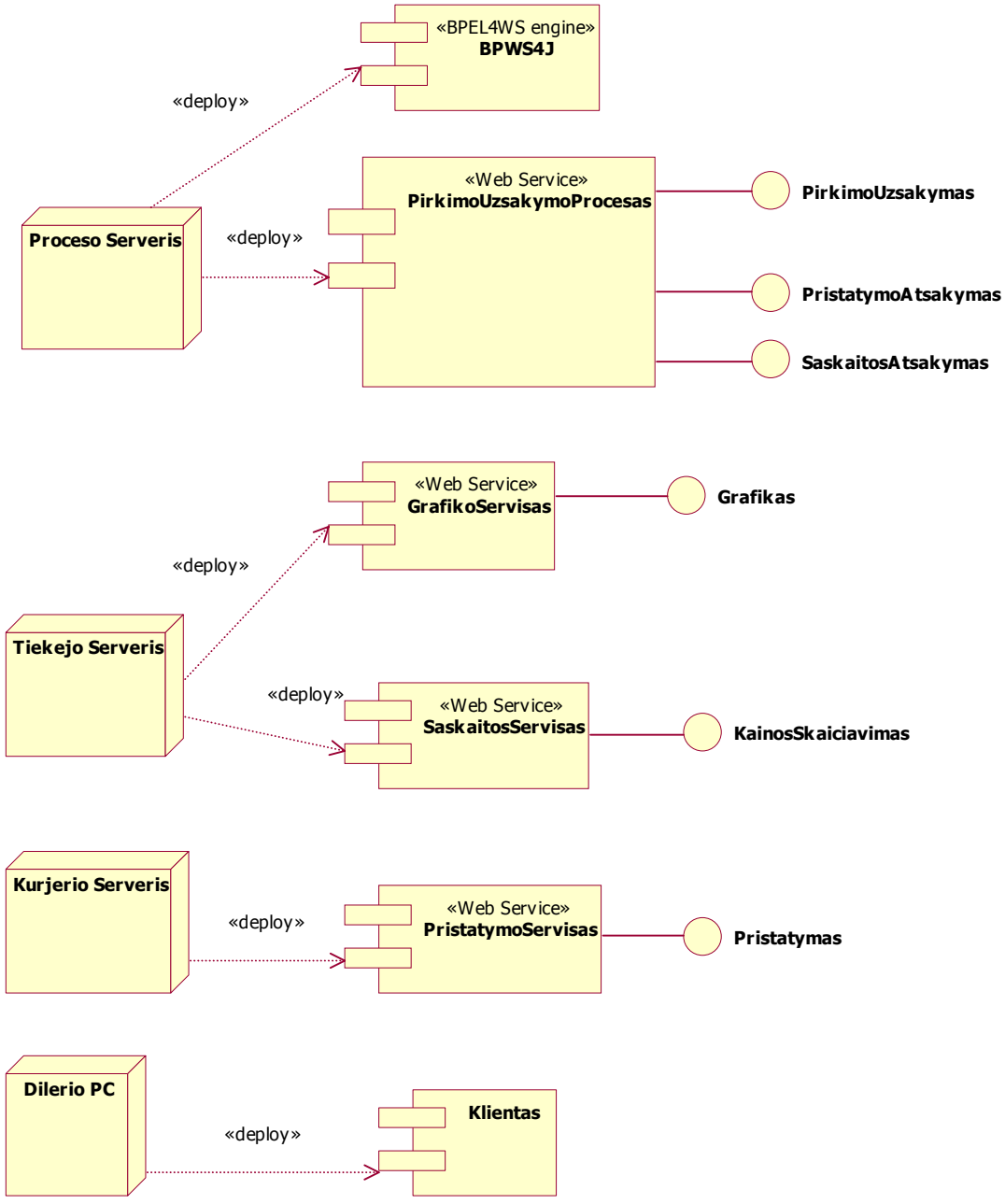
4.2. Proceso web servisų realizacija

Kadangi automatizuotų verslo procesų sistemos yra sąveikaujančios kitos sistemos – web servisai, tai tiesioginio jų vartotojo nėra. Vartotojas gali sąveikauti tik su tam tikrais proceso komponentais t.y. kurio nors partnerio serviso vykdyme. 27 pav. pavaizduota panaudojimo atvejų diagrama vaizduoja užsakymo proceso dalyvių sąveiką.

27 pav. užsakymo proceso panaudojimo atvejų diagrama

28 pav. pavaizduoti užsakymo proceso komponentų ir jų įdiegimo diagrama. Eksperimentinė sistema realiai yra viename kompiuteryje, tačiau turi analogišką loginį

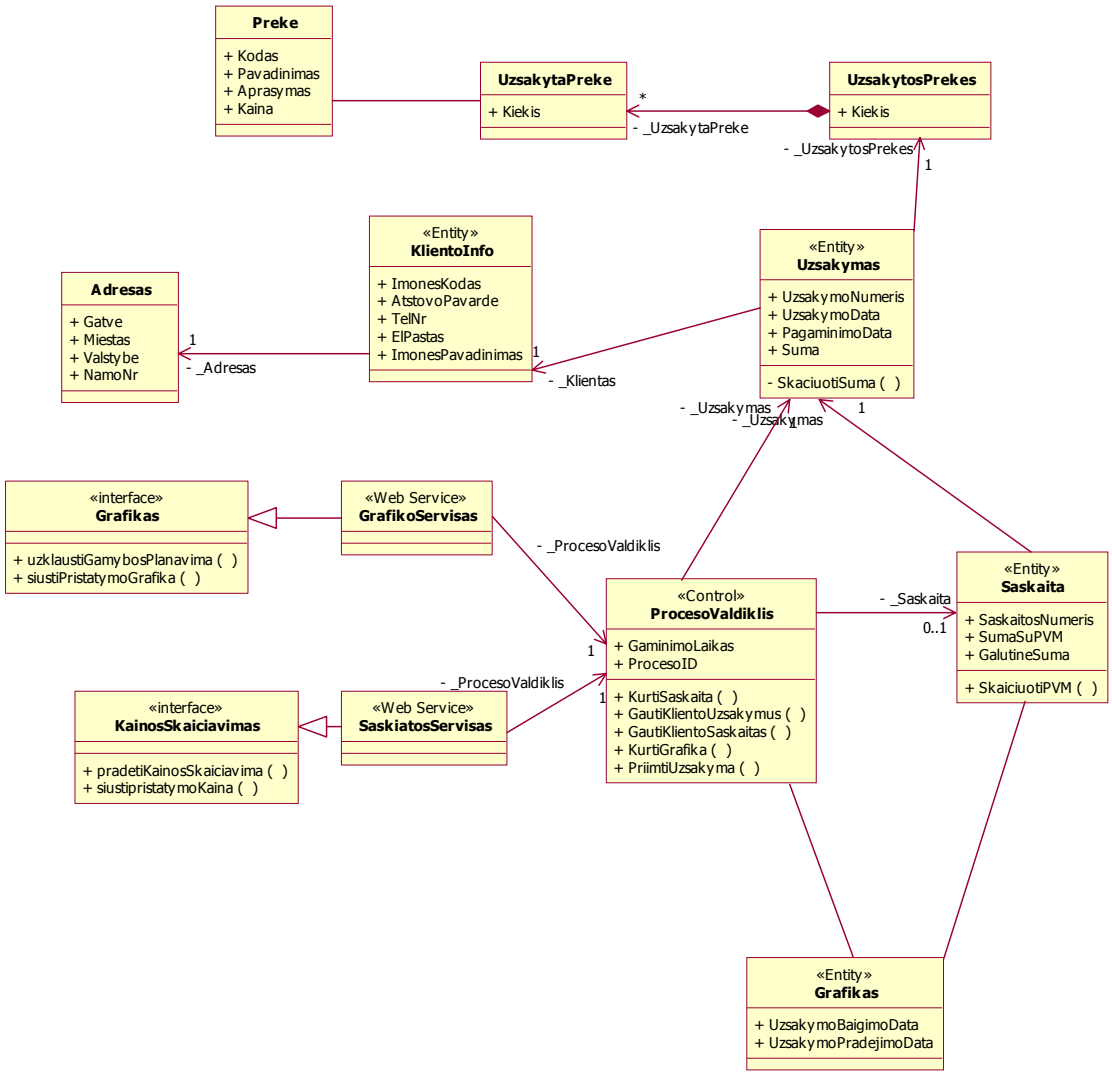
suskaldymą. Komponentai atitinka 3 dalyje nagrinėtas proceso roles, jie realizuoja atitinkamus interfeisus.



28 pav. komponentų diagrama

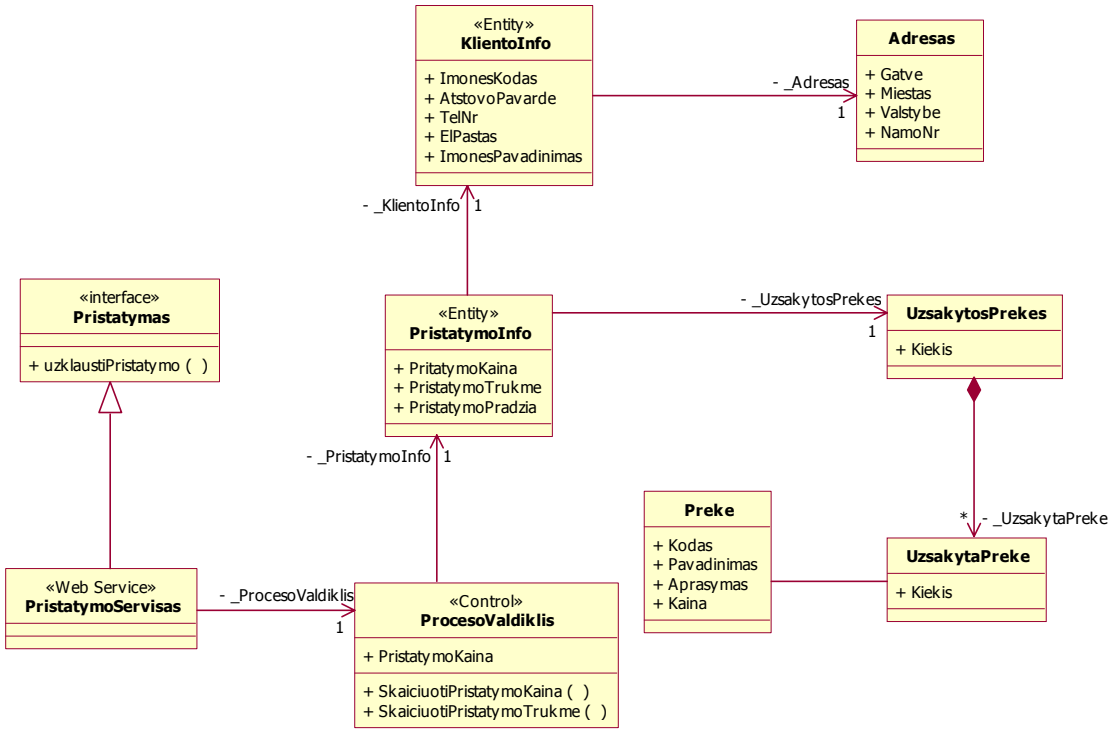
Matome, kad realiai procesas sąveikauja tik su trimis web servisais. Klientas yra tik modeliuojamas kaip web servisas, tačiau jokių užduočių neatlieka, todėl yra tik proceso kaip web serviso vartotojas.

Tiekėjas procese atlieka dvi roles: skaičiuoja galutinę sąskaitos kainą bei nustato užsakymo įvykdymo bei pristatymo terminus. 29 pav. pavaizduota tiekėjo servisus realizuojančių klasių diagrama. Stereotipu <<entity>> pažymėtos klasės reiškia esybes, kurios apibrėžtos BPEL procese. Web servisų klasės realizuoja atitinkamus proceso interfeisus, o funkcionalumas atliekamas per "ProcesoValdiklis" klasę. Proceso valdiklio parametrai naudojami verslo sprendimams (užsakymo gaminimo terminui) vykdyti. Jie yra nuskaitomi iš XML failo.



29 pav. tiekėjo klasių diagrama

Analogiškai sukurtas ir kurjerio web servisas. 30 pav. pavaizduota kurjerio web serviso klasių diagrama.



30 pav. kurjerio web servisą realizuojančių klasių diagrama

5. Išvados

5.2 Darbo rezultatai

- Darbe analizuojama Web servisų komponavimo į automatizuotus verslo procesus problema, kurios sprendimas padėtų išnaudoti naujausių interneto technologijų potencialą ir paversti internetą globalia bendrai vykdomo verslo platforma.
- Apžvelgtos tam naudojamos automatizuotų verslo procesų vykdymo kalbos ir pasirinkta nauja BPES4WS kalba, kuri šiuo metu turi didžiausias perspektyvas web servisų komponavimui, nes apima kelių ankstesnių kalbų savybes, o už labiausiai išvystytą ebXML standartą ji pranašesnė tuo, kad bendro verslo automatizavimui nereikalauja tiek daug standartų ir papildomų išteklių.
- Išanalizuota BPEL4WS specifikacija bei vykdomų procesų modeliavimas su BPWS4J moduliu IBM Eclipse platformoje. Nors ši kalba turi dideles perspektyvas, į ją gilinasi programuotojai, mėgstantys naujoves, arba teoretikai. Kol kas nėra sukauptos jos naudojimo patirties nei taikymo metodikos, nėra praktinių pavyzdžių. Skirtingai nuo darbų sekų tipo procesų modeliavimo priemonių, ši kalba nėra intuityviai suprantama.
- Norint padaryti šią kalbą prieinamesnę analitikams, programuotojams ir projektuotojams, išnagrinėtos BPEL4WS procesų modeliavimo galimybės standartine UML kalba ir Rational Rose paketu ir sudaryta tam skirta metodika.
- Sukūrus tam tikslui programines priemones, iš sudarytų verslo proceso modelių BPEL4WS scenarijus būtų galima generuoti automatiškai. Šiame darbe UML modeliai buvo panaudoti tradiciškai kaip gairės programuotojui.
- Realizuotas eksperimentinis trijų dalyvių verslo procesas, kuris veikia BPWS4J varikliuke. Web servisai realizuoti Microsoft .NET platformoje ir ištestuoti tam sukurtu trasavimo mechanizmu.
- BPEL4WS kalba ir modeliavimo įrankiai turėtų būti toliau tobulinami ir plečiami. Reikalinga vieninga proceso elementų notacija, nes vien UML stereotipai modelius daro painiais, o įvairių paketų (BizTalk, Collaxa, IBM Web Sphere) notacijas reikia suprasti kiekvieną atskirai. Būtina tobulinti ir

modeliavimo aukštesniame lygyje metodiką, padaryti ją intuityviau suprantamą, užtikrinti modelio teisingumo tikrinimo galimybes. Didieji PĮ gamintojai rodo didelį susidomėjimą, tačiau realiai veikiančių produktų kol kas nėra, todėl rinka vis dar atvira BPEL4WS modeliavimo produktams, procesų vykdymo varikliukams ar serveriams.

5.2 BPEL4WS privalumai

- Gryna sąveikaujančių web servisų koncepcija.

Procesas yra web servisas ir standartinėmis priemonėmis sąveikauja su kitais servisais.

- Duomenų tipų ir jų struktūrų laisvas sudarymas, paveldėjimas bei atskyrimas nuo proceso.

Vartotojas gali laisvai keisti duomenų tipus nekeisdamas proceso struktūros. Duomenų tipai apibrėžiami XML Schema.

- Galimybė naudoti kaip EAI priemonę

Nors BPEL4WS nėra skirtas programų integravimui spręsti, jį galima naudoti šių problemų sprendimams.

- Proceso aprašas „programuotojiško“ pobūdžio: WSDL, XPath, XSD

Viskas aprašoma XML, todėl nereikalingos papildomos technologinės žinios ar programinė integravimo įranga

- Koordinuojamas asinchroninis komunikavimas tarp servisų

Labai svarbus aspektas Web servisų kompozicijoje vykdant ilgai trunkančias veiklas, nes sinchroninis komunikavimas realaus pasaulio programose būtų neįmanomas.

- Procesus galima modeliuoti naudojant standartinę UML ir gauti sugeneruotą BPEL scenarijų

Nėra būtinumo įsisavinti naujas modeliavimo aplinkas, pakanka suprasti BPEL4WS struktūrą ir specifinius UML išplėtimus.

- LRT (Long Running Transaction) – ilgai vykdomų transakcijų palaikymas

Nors pati BPEL4WS specifikacija nepalaiko transakcijų, tačiau ją galima išplėsti WS-Transaction bei WS-Coordination specifikacijomis ir gauti šį būtiną automatizuotiems verslo procesams funkcionalumą.

- Procesas aprašo būtinus interfeisus

Sumodeliuotas procesas jau pateikia būtinus integravimosi interfeisus, todėl sprendžiant partnerio integracijos problemas tereikia sukurti web servisą realizuojantį reikalingus interfeisus.

- Nereikalauja didelės specializuotos kvalifikacijos.

BPEL4WS sąlyginai paprastas lyginant su ebXML, kuris teikiantis daug abstrakčių procesų, duomenų tipų ir servisų šablonų bei specializuotų verslo konteksto procesų šablonų reikalauja gilaus ne tik pačios specifikacijos bet ir šablonų, bei duomenų tipų bibliotekų žinių.

- Pranešimų mainų koreliavimas tarp verslo partnerių

Pranešimų koreliavimas leidžia sąveikauti su proceso egzemplioriumi vykdant kelis procesus iš karto.

- Realizuojamas lygiagretus veiklų vykdymas

Automatizuotiems verslo procesams vykdyti svarbu ne tik veiksmų vykdymo seka, bet ir jų vykdymas lygiagrečiai. Kartais kai kurios veiklos reikalauja duomenų iš kitų veiklų kurios vykdomos paraleliai. Šią galimybę BPEL4WS taip pat palaiko.

- Teikiamas klaidų apdorojimo mechanizmas

Kadangi Web servisų kompozicija pagrįsti procesai yra priklausomi nuo keleto servisų vykdomų pas skirtingus partnerius, egzistuoja didele klaidų tikimybė. Gebėjimas jas apdoroti yra svarbus privalumas.

5.3 BPEL4WS trūkumai

- Sunku testuoti

Modelio testavimas atliekamas tik atliekant visa procesą ir lyginant rezultatus su tikėtinais gauti duomenimis. Nėra trasavimo ir tarpinių duomenų stebėjimo galimybės, kadangi pati kalba neteikia šios galimybės.

- Nėra tiesioginio transakcijų palaikymo

Būtina BPEL4WS scenarijų papildyti WS-Transaction bei WS-Coordination specifikacijomis, norint vykdyti transakcijas.

- Modeliavimo procesas nėra intuityvus

Partnerių išskyrimas, duomenų tipų ir paties proceso sudarymas reikalauja specifinio suvokimo.

- Nėra modelio teisingumo tikrinimo priemonių

Modelio teisingumas tikrinamas tik testų pagalba, o toks modelio kūrimas ilgai trunka ir padidina klaidų tikimybę.

- Sudėtingėjant procesui BPEL modelį darosi sunku suprasti.

- Nėra duomenų transformavimo specifikavimo.

- Nėra duomenų konvertavimo iš EDI ar binarinio formato.

- Nespecifikuojama žmogaus veiksmų seka.

- Nėra partnerių susitarimų ar derybų specifikavimo, kuris yra svarbus verslo procesuose

5.4 BPEL4WS nauda

Įmonėms BPEL4WS suteikia sąveikavimo terpę su verslo partneriais vykdyti verslo procesus. Įmonių paslaugos gali būti ne tik publikuojamos internete web servisų pagalba, bet ir dalyvauti automatizuotuose procesuose.

Sistemų projektuotojams galimybė BPEL4WS modelius kurti standartinėmis priemonėmis bei programuotojams naudoti standartines technologijas leidžia nesunkiai įsisavinti šią kalbą, o didelis PĮ rinkos lyderių susidomėjimas garantuoja visuotiną šios specifikacijos palaikymą bei paplitimą.

6. Literatūros sąrašas

Specifikacija. Business Process Execution Language for Web Services (BPEL4WS) 1.1 Prieiga per Internetą: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Paul Brown. BPEL for Programmers and Architects [interaktyvus] 2003 [žiūrėta 2003 10 22]. Prieiga per Internetą: http://blog.fivesight.com/prb/space/BPEL/BPEL4ProgArchies.pdf

Mike Lehmann. BPEL: Business Processes with Web Services [interaktyvus]. 2003 10 02, [žiūrėta 2003 11 15]. Prieiga per Internetą: http://blog.fivesight.com/prb/space/Other+sources+of+information+on+BPEL/Oracle_BPEL_08_13_03.pdf

IBM Corp. Business processes with BPEL4WS: Learning BPEL4WS. Iš IBM, developerWorks [interaktyvus]. 2003, [žiūrėta 2003 12 22]. Prieiga per Internetą: http://www-106.ibm.com/developerworks/views/webservices/articles.jsp

Barbara Darrow, Elizabeth Montalbano. IBM Exec Touts Need For BPEL Support, SOAs [interaktyvus]. 2003, [žiūrėta 2003 12 22]. Prieiga per Internetą: http://www.crn.com/sections/BreakingNews/dailyarchives.asp?ArticleID=44275

MacDonald M. Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting. – Microsoft Press, 2003. – 717 p.

Michael C. Daconta, Leo J. Obrst, Kevin T. Smith. The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. – John Wiley & Sons, 2003. – 281 p.

Doug Kaye. Loosely Coupled: The Missing Pieces of Web Services. – RDS Press, 2003. – 334 p.

David Jorgensen. Developing .NET Web Services with XML. – Syngress Publishing, 2002. – 457 p.

Scott Short. Building XML Web Services for the Microsoft .NET Platform. – Microsoft Press, 2002. – 426 p.

Nemuraitė L., Oželis A. E.verslo įgyvendinimo metodika naudojant procesų vykdymo ir tinklo paslaugų apibrėžimo kalbas // Informacinės technologijos '2004: konferencijos pranešimų medžiaga [Kaunas, 2004m. sausio 28, 29 d.]. Kaunas, 2004.

7. Summary

Today Web services can communicate with each other, advertise themselves, and be discovered and invoked using industry-wide specifications. However, linking these services together into a business process or a composition gave the user a number of conflicting specifications to choose from – as was the case with WSFL from IBM and XLANG from Microsoft. The Business Process Execution Language for Web Services (BPEL4WS) represents the merging of WSFL and XLANG, and with luck, will become the basis of a standard for Web service composition. BPEL4WS combines the best of both WSFL (support for graph oriented processes) and XLANG (structural constructs for processes) into one cohesive package that supports the implementation of any kind of business process in a very natural manner. In addition to being an implementation language, BPEL4WS can be used to describe the interfaces of business processes as well – using the notion of abstract processes.

8. Priedai

8.1. Eksperimentinio užsakymo proceso BPEL scenarijaus kodas

```
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         name="PirkimoUzsakymoProcesas"

targetNamespace="http://localhost/Tiekejas/PirkimoUzsakymas/PirkimoUzsakymoProcesas/"
         xmlns:tns="http://localhost/Tiekejas/PirkimoUzsakymas/PirkimoUzsakymoProcesas/"
         suppressJoinFailure="yes"
         enableInstanceCompensation="no"
         abstractProcess="no">
  <partners>
    <partner name="Klientas"
xmlns:ns1="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns1:Uzsakymas" myRole="UzsakymoServisas"/>
    <partner name="SaskaitosSiuntimas"
xmlns:ns2="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns2:saskaita" myRole="SaskaitosUzklausimas"
partnerRole="SaskaitosServisas"/>
    <partner name="pristatymoSiuntimas"
xmlns:ns3="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns3:Pristatymas" myRole="pristatymoUzklausimas"
partnerRole="Pristatymas"/>
    <partner name="grafikoSiuntimas"
xmlns:ns4="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/"
serviceLinkType="ns4:Grafikas" partnerRole="Grafikas"/>
  </partners>
  <variables>
    <variable name="Uzsk"
xmlns:ns5="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuomenuTipai/"
messageType="ns5:PO"/>
    <variable name="saskaita"
xmlns:ns6="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuomenuTipai/"
messageType="ns6:InvoiceMessage"/>
    <variable name="uzskKLaida"
xmlns:ns7="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuomenuTipai/"
messageType="ns7:OrderFault"/>
    <variable name="pristatymoGrafikas"
xmlns:ns8="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuomenuTipai/"
messageType="ns8:ScheduleInfoMessage"/>
    <variable name="pristatymoInfo"
xmlns:ns9="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuomenuTipai/"
messageType="ns9:ShippingInfoMessage"/>
```

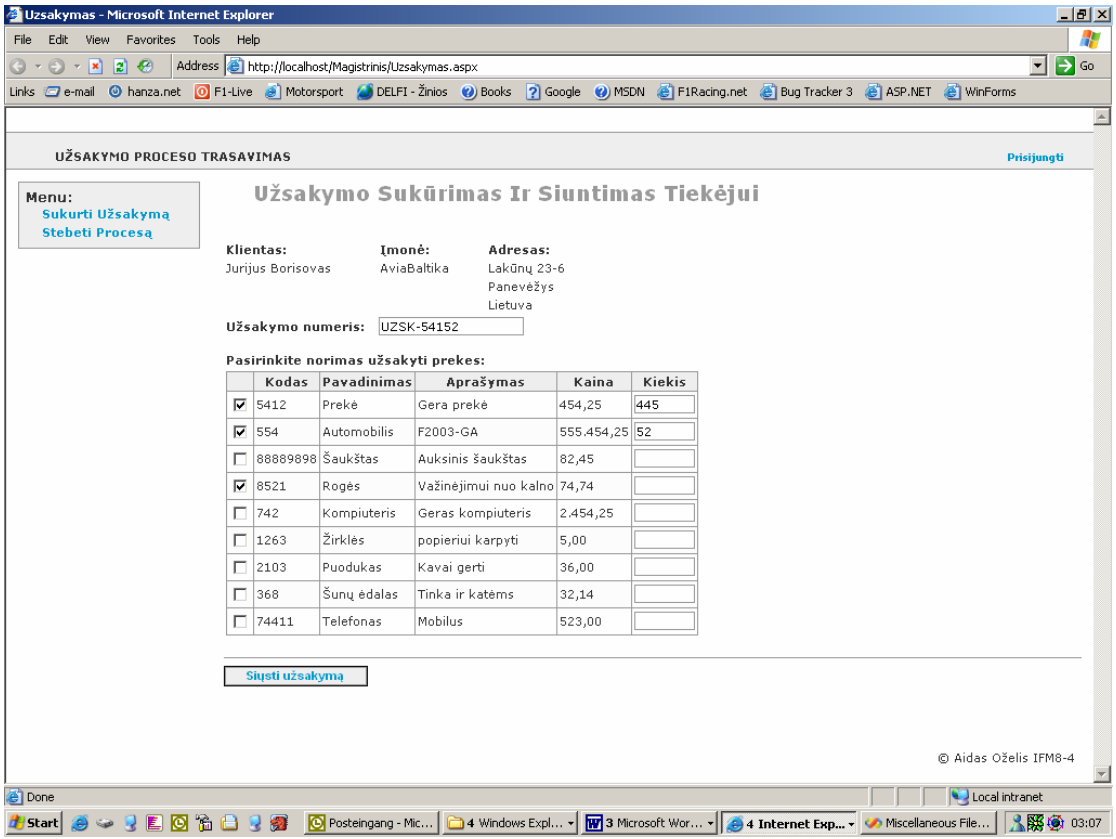
```
    <variable name="pristatymoUzklausimas"
xmlns:ns10="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/UzsakymoDuomenuTipai/"
messageType="ns10:ShippingRequest"/>
  </variables>
  <faultHandlers>
    <catch xmlns:ns11="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
faultName="ns11: uzsakymasNebaigtas" klaidosKintamasis="uzskKLaida">
      <reply name="grazintiKlaida"
             partner="Klientas"
xmlns:ns12="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns12:PurchaseOrderService" operation="sendPurchaseOrder"
             variable="uzskKLaida"
xmlns:ns13="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
faultName="ns13:cannotCompleteOrder">
      </reply>
    </catch>
  </faultHandlers>

  <sequence name="Procesas">
    <receive name="receive"
              partner="Klientas"
xmlns:ns14="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns14:PurchaseOrderService" operation="siustiPirkimoUzsakyma"
             variable="Uzsk" createInstance="yes">
    </receive>
    <flow>
      <links>
        <link name="invoke_to_sendPrice"/>
        <link name="shippingProvider_to_sendSchedule"/>
      </links>
      <sequence name="pristatymoSiuntimas">
        <source linkName="shippingProvider_to_sendSchedule"/>
        <assign  name="assign">
          <copy>
            <from variable="Uzsk" part="customerInfo"/>
            <to variable="pristatymoUzklausimas" part="customerInfo"/>
          </copy>
        </assign>
        <invoke name="invoke"
                partner="pristatymoSiuntimas"
xmlns:ns15="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns15:ShippingService" operation="uzklaustiPristatymo"
                inputVariable="pristatymoUzklausimas" outputVariable="pristatymoInfo">
          <source linkName="invoke_to_sendPrice"/>
        </invoke>
        <receive name="gautiGrafika"
                  partner="pristatymoSiuntimas"
xmlns:ns16="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns16:ShippingCallback" operation="sendSchedule"
                 variable="pristatymoGrafikas">
```

```
        </receive>
      </sequence>
      <sequence name="grafikoSiuntimas">
        <invoke name="uzklaustiGrafiko"
                partner="grafikoSiuntimas"
xmlns:ns17="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns17:SchedulingService" operation="requestProductionScheduling"
                inputVariable="Uzsk">
        </invoke>
        <invoke name="siustiGrafika"
                partner="grafikoSiuntimas"
xmlns:ns18="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns18:SchedulingService" operation="sendShippingSchedule"
                inputVariable="pristatymoGrafikas">
          <target linkName="shippingProvider_to_sendSchedule"/>
        </invoke>
      </sequence>
      <sequence name="saskaitosSiuntimas">
        <invoke name="skaiciuotiKaina"
                partner="SaskaitosSiuntimas"
xmlns:ns19="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns19:ComputePriceService" operation="initiatePriceCalculation"
                inputVariable="Uzsk">
        </invoke>
        <invoke name="siustiKaina"
                partner="SaskaitosSiuntimas"
xmlns:ns20="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns20:ComputePriceService" operation="sendShippingPrice"
                inputVariable="pristatymoInfo">
          <target linkName="invoke_to_sendPrice"/>
        </invoke>
        <receive name="gautiSaskaita"
                  partner="SaskaitosSiuntimas"
xmlns:ns21="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns21:InvoiceCallback" operation="sendInvoice"
                 variable="saskaita">
        </receive>
      </sequence>
    </flow>
    <reply name="reply"
           partner="Klientas"
xmlns:ns22="http://localhost/Tiekejas/PirkimoUzsakymoProcesas/Purchase/"
portType="ns22:PurchaseOrderService" operation="sendPurchaseOrder"
           variable="saskaita">
    </reply>
  </sequence>
</process>
```

8.2 Eksperimentinio BPEL4WS proceso trasavimo klientinė dalis

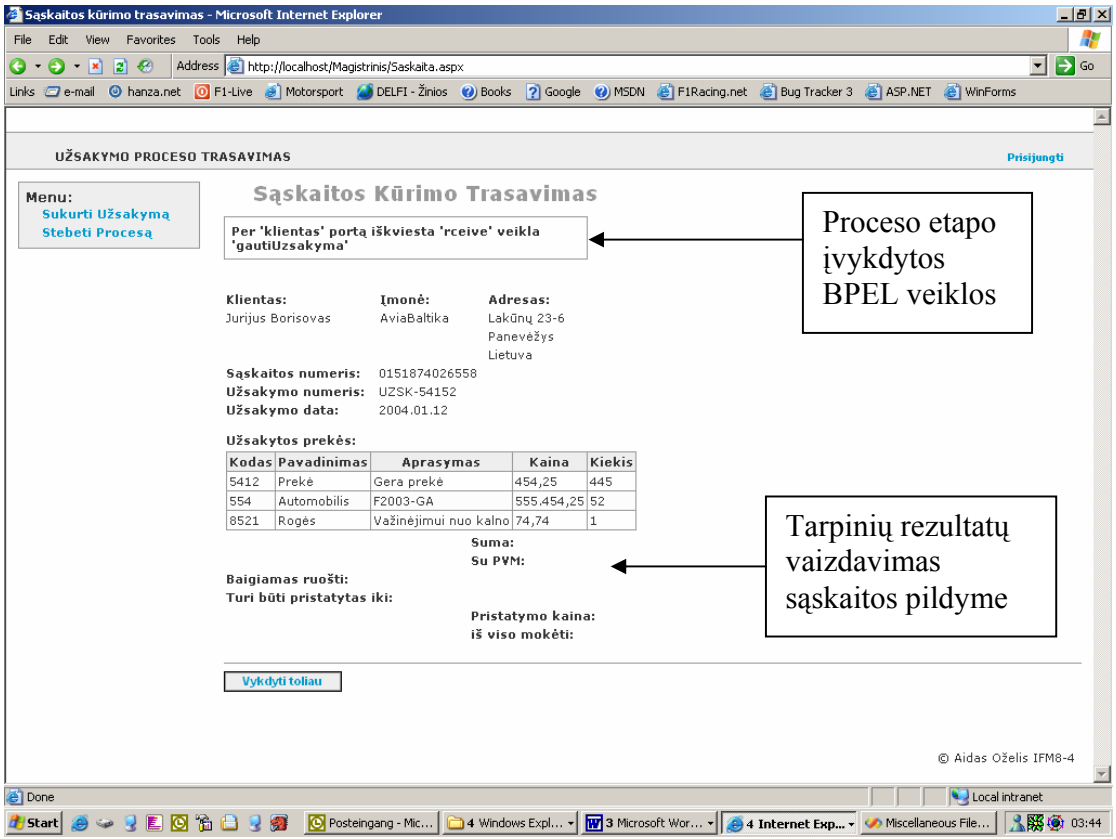
Kadangi sukurtas eksperimentinis BPEL4WS procesas yra pilnai automatizuotas, siekiant pademonstruoti jo veikimą buvo iliustraciniais motyvais sukurtas proceso trasavimo mechanizmas. Klientinė dalis realizuota ASP. NET ir turi interneto sąsają. Šis mechanizmas įmanomas tik kai web servisai veikia tame pačiame kompiuteryje, nes web servisus realizuojantys komponentai turi sąveikauti .NET vykdymo aplinkoje, kad būtų galima siųsti ir gauti pranešimus apie įvykius bei gauti tarpinius rezultatus.

Prieš vykdant proceso atitinkamas veiklas, kurių rezultatas yra proceso sąveika su web servisu, web servisas siunčia įvykį trasavimo mechanizmui ir stabdo savo vykdymą laukdamas įvykio iš kliento tęsti darbą toliau. 31 pav. pavaizduotas proceso inicijavimas: sukuriamas užsakymas ir siunčiamas tiekėjui.



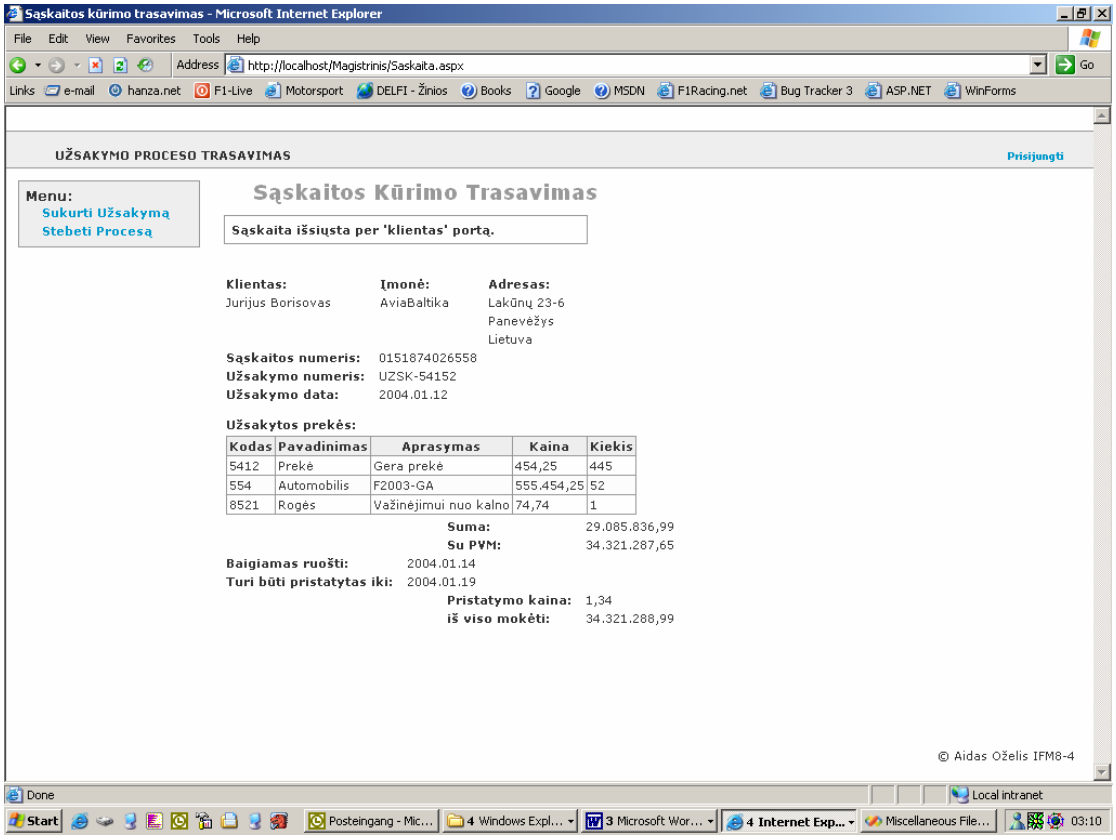
31 pav. proceso inicijavimas.

Inicijavus procesą, jo tolimesnis vykdymas iliustruojamas sąskaitos pildymu. Pradžioje sąskaita yra tuščia, užpildyti tik tie laukai, kurie yra žinomi iš užsakymo (32 apv.).



32 pav. proceso trasavimas

Toliau vartotojas stebi proceso veiklą gaudamas informaciją kokios veiklos įvyko ir kokie gauti jų rezultatai. Stebėdamas procesą vartotojas trasavimą vykdo pažingsniui siųsdamas komandą vykdyti procesą toliau. Siunčiant komandą vykdyti procesą, web servisų komponentai gauna įvykius tęsti darbą ir vykdo procesą toliau iki sekančio etapo kuomet vykdymas sustabdomas ir rezultatams stebėti.



33 pav. baigtas proceso vykdymas.

8.3 CD turinys

Kataloge “Dokumentacija” yra elektroninis šios dokumentacijos variantas MS Word faile.

Kataloge “BPEL4WS” yra eksperimentinio proceso WSDL failai bei proceso BPEL failas.

Kataloge “Realizacija” yra eksperimentinio užsakymo proceso web servisų realizacijos išeities tekstai.

Kataloge “Skaidres” yra prezentacijos skaidrės.