KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Simas Grinkevičius
Vilius Levickas

Koordinatoriaus modelis programinės įrangos paslaugų sistemų kūrimui

Magistro darbas

Darbo vadovas

prof. L. Nemuraitė

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Simas Grinkevičius
Vilius Levickas

Koordinatoriaus modelis programinės įrangos paslaugų sistemų kūrimui

Magistro darbas

Recenzentas
doc.dr. Valentinas Kiauleikis
2007-05-

Vadovas
prof. L. Nemuraitė
2007-05-

Atliko
IFM-1/2 gr. stud.
Simas Grinkevičius
Vilius Levickas
2007-05-15

Kaunas, 2007

Turinys

Summary

In this work the coordinator model, designed for Internet services systems creation is presented. This model ensures strict system structure, reliability, testability, manageability and also easy modification and creation of different system features. The Internet publishing services system is created, which is designed to prove the appropriateness of the aforementioned coordinator model. The implementation of coordinator service required some effort, but this service greatly relieves the struggle of including new services or modification of existing and can be used when creating new systems. The created Internet publishing services system specification is described further. Afterwards, the system is tested against usability and compliance to coordinator model and is compared to other technologies such as BPEL and chaotic programming. The assumption is posed that coordinator model is best suited for development of small and medium information services systems. The performed research on the subject showed this was truly the case, where BPEL technology was best suited for big systems and not medium and small ones because of complexity and cost issues. The conclusion later is done that information systems are often developed in chaotic and rapidly changing environments so coordinator model is suited for them best because of its ease of changing and establishing of new system components. Therefore, the system created is universal product, which can be naturally adapted for many similar systems. The article was written on this work, which was presented at the 12th interuniversity conference of graduate students “Information society and university studies IVUS’07”.

1. Įvadas

Kadangi pastaruoju metu vis daugiau sistemų kuriama naudojant tinklo paslaugas, dar vadinamas Web servisais (angl. *Web services*), aktualu naudoti efektyvius paslaugų kūrimo metodus. Praktikoje nedidelės paslaugų sistemos dažniausiai kuriamos chaotiškai, o vienintelis plačiai pripažintas paslaugų sistemų kūrimo metodas paremtas BPEL[1] kalba. Tačiau BPEL kalba aprašytai sistemai funkcionuoti reikalinga speciali įranga (BPEL variklis), todėl BPEL kalbą netikslinga naudoti nedidelėms sistemoms. Todėl šiuo darbu siekiama išbandyti mažiau sąnaudų reikalaujantį koordinatoriaus modelį[23], kad jį būtų galima efektyviai taikyti praktikoje kuriant nedideles ir vidutinio dydžio paslaugų sistemas.

Darbo tikslas

- Patobulinti paslaugų sistemų architektūrą ir kūrimo procesą, pritaikant joms paslaugų koordinatoriaus šabloną.
- Tam reikia:
    - Realizuoti paslaugų sistemos prototipą, naudojant koordinatoriaus šabloną
    - Tuo pačiu ištirti koordinatoriaus tinkamumą paslaugų sistemų kūrimui, kad šį modelį būtų galima rekomenduoti projektuotojams, kuriantiems paslaugų informacines sistemas

Sistemos prototipu buvo pasirinkta leidyklos paslaugų sistema, kadangi buvo nustatytas tokio tipo paslaugų poreikis.
Pagrindiniai sistemos veikėjai yra autorius, koordinatorius, tikrintojas bei recenzentas. Autorius pateikia savo darbus leidybos agentūrai, koordinatorius priima darbus bei juos perduoda patikrinimui ir recenzijai, tikrintojas patikrina ar leidinys atitinka pradinius reikalavimus, peržiūri finansavimą, įvertina recenzijas, o recenzentas atlieka kūrinio recenziją.

Kūrinių recenzentų darbams keliama keletas reikalavimų: kritikos profesionalumas, savitumas ir originalumas. Knygų autoriai gali būti kitų autorių kūrinių (bet ne savo) recenzentais, už tai

[1] Business Process Execution Language – verslo procesų vykdymo kalba, pagrįsta XML technologija ir sukurta tam, kad standartizuotų integracijos logiką ir procesų automatizavimą tarp Web servisų.

jiems taikomos įvairios nuolaidos bei privilegijos ir taip tikimasi sumažinti įdarbinamų recenzentų kiekį bei jiems skiriamas sąnaudas.

Tyrimo sritis ir objektas

Tyrimo sritis – informacinių sistemų, sudarytų iš koordinuojamų Web servisų, kūrimo metodai.

Tyrimo objektas – koordinatoriaus modelio pritaikymas leidybos agentūrų sistemai, kurioje vyksta iš daugelio etapų susidedantys ir didelių organizacinių pastangų reikalaujantys procesai bei galimas koordinatoriaus modelio pritaikymas naujų sistemų kūrime.

Sprendžiama problema

Šis darbas turi atsakyti į klausimą, kaip reikėtų praktikoje projektuoti programinės įrangos paslaugų sistemas, kai daugelio versle vykdomų veiklos procesų taisyklės nuolat kinta ir reikalauja dažnų programinės įrangos pakeitimų[24, 215 p.], tačiau sistema nėra tokia didelė, kad jai vertėtų taikyti BPEL variklį. Pagrindinė problema iškyla tada, kai reikia restruktūrizuoti esamą programų sistemą, pridėti naujas funkcijas ar modifikuoti jau esančias. Tokiu atveju paprastai reikia daug darbo sąnaudų. Darbe buvo įgyvendintas koordinatoriaus modelis, kuris gerokai palengvina tokių pakeitimų atlikimą. Be to, šis modelis gerokai palengvina bei supaprastina ir visiškai naujų programinės įrangos paslaugų sistemų kūrimą.

Darbo uždaviniai

Pagrindiniai šio darbo uždaviniai yra šie: iš teorinės pusės išanalizuoti Web servisus bei technologijas, kuriomis jie remiasi, įvertinti jų privalumus ir trūkumus, palyginti sukurtą produktą su BPEL verslo procesų modeliavimo kalba, išanalizuoti jau esančius rinkoje produktus, aprašyti koordinatoriaus modelį iš mokslinės pusės, parinkti tinkamas jo realizavimo technologijas, pritaikyti koordinatoriaus modelį bei pasirinktas technologijas kuriant informacinę leidybos agentūros sistemą, aprašyti techninę sukurtos sistemos pusę, išanalizuoti, kokie galėtų būti jau sukurtos sistemos patobulinimai, atlikti programinės įrangos naudojimo eksperimentą, įvertinti jo rezultatus, pateikti naudojimo rekomendacijas.

Darbe naudoti metodai

Sistema remiasi doktorantės Linos Čeponienės sudarytu metodu[23] (veiklos paslaugų sistemos projektavimas, pradedant nuo reikalavimų apibrėžimo, kuris transformuojamas į projektą remiantis architektūriniu šablonu). Metodas remiasi modeliu, kuris apibrėžia pagrindinį servisą – koordinatorių, valdantį kitus, šalutinius ir papildomus, servisus. Koordinatorius atsakingas už užklausų priėmimą, *pre* ir *post* sąlygų patikrinimą, atitinkamos paslaugos iškvietimą, pranešimo siuntimą pradinės užklausos siuntėjui ir pan.

Sistema buvo kuriama evoliuciniu programavimo metodu. Buvo pradėta nuo programų sistemos prototipo, kai buvo sukurtas ir įgyvendintas koordinatoriaus servisas, o vėliau sistema buvo išplėsta bei sukurti papildomi Web servisai. Sukūrus sistemą, buvo atlikta lyginamoji analizė.

Kas padaryta

Sistemos kūrimo metu buvo išnagrinėta daug literatūros šaltinių bei išanalizuotas problemos sprendimas pasaulyje. Taip pat buvo įvertinta situacija Lietuvoje. Tuomet buvo pradėta projektuoti pati programų sistema. Buvo numatytos ir suprojektuotos programų sistemos funkcijos, vartotojo charakteristikos, aprašytos problemos, tikslai bei bendri apribojimai. Atlikti projekto įgyvendinimo planai ir kokybės vertinimo analizė. Buvo aprašyti reikalavimų specifikavimas, architektūros specifikacija bei detalios architektūros specifikacija.

Sistema buvo įgyvendinta pagal parengtus dokumentus ir specifikacijas, naudojant Microsoft .NET technologiją. Užbaigus pagrindinius programavimo darbus buvo atliktas visapusiškas sistemos testavimas panaudojant juodos ir baltos dėžės metodus. Taip pat buvo sukurta detali sistemos vartotojo dokumentacija bei atliktas sistemos kokybės vertinimas.

Darbo rezultatai

Buvo sukurta anksčiau minėtu metodu besiremianti programinė įranga – nedidelė eksperimentinė leidyklų agentūros sistema, kurioje naudojami Web servisai. Taigi, buvo sukurtas Web servisų sistemos modelis, ištirtas jo veikimas ir pasiūlytas šablonas, kurį galėtų naudoti daugelis projektuotojų, siekiantys kurti informacines sistemas, komponuodami jas iš elementarių, būsenų nesaugančių Web servisų.

Eksperimentas

Gauti rezultatai buvo eksperimentiškai patikrinti realybėje. Programinė sistema buvo pateikta darbui ir bandymams konkretiems vartotojams. Neskaitant keleto nedidelių nesklandumų, sistema buvo priimta gerai ir atsiliepimai apie ją buvo teigiami.

Asmeninis indėlis

Vilius Levickas buvo atsakingas už pagrindinį Web servisą (reikalavimų specifikavimas, architektūros specifikacija, detalios architektūros specifikacija, testavimas, vartotojo dokumentacija). Simas Grinkevičius – už sąsajas (reikalavimų specifikavimas, architektūros specifikaciją, detalios architektūros specifikacija, testavimas, vartotojo dokumentacija). Paslaugų Web servisai buvo daromi bendrai.

Rašant teorinę dalį, Vilius Levickas analizavo BPEL bei apdorojo apklausos duomenis. Simas Grinkevičius atliko Web servisų bei jų technologijų analizę ir organizavo apklausą. Prie koordinatoriaus bei jo šablono aprašymo dirbta bendrai.

Darbo privalumai

Tokia informacinė sistema turi nemažų perspektyvų konkuruoti tiek Lietuvoje, tiek ir pasaulyje, nes ypač Lietuvoje analogiškų sistemų nėra labai daug. Sukurta sistema skirta naudojimui leidyklose bei leidybos agentūrose. Dėka sukurto koordinatoriaus modelio, sistema nesunkiai pritaikoma skirtingose darbo aplinkose, t.y. skirtingos leidybos agentūros bei leidyklos gali lengvai ją pritaikyti savo reikmėms. Negana to, sistema su nedideliais pakeitimais gali būti naudojama ir kitose srityse, kuriose reikalinga centrinė vartotojų pateikiamos informacijos koordinacija ir valdymas.

Praktinis pritaikymas

Pagal Lietuvos leidėjų katalogą[23], vien Lietuvoje šiuo metu įregistruoti 1597 leidėjai. Daugiau nei 10 skirtingų pavadinimų knygų per metus išleidžia net 67 Lietuvos leidėjai. Be abejonės, užsienyje leidėjų yra gerokai daugiau. Sukurtam eksperimentiniam produktui egzistuoja plati rinka, tereikia sugebėti ją išnaudoti. Sistemos architektūra lanksti pokyčiams, todėl esamą sistemą būtų galima pritaikyti įvairiems leidybos taisyklių atvejams, o koordinatoriaus programinę įrangą – kitoms taikymo sritims.

Darbo struktūra

Darbas susideda iš 7 pagrindinių dalių: įvado, analitinės, projektinės, tyrimo bei eksperimentinės dalių, išvadų ir literatūros. Įvade nusakomi pagrindiniai darbo tikslai bei uždaviniai, sprendžiamos problemos bei metodai, kuriais buvo remiamasi, gauti rezultatai bei darbo ateities perspektyvos. Analitinėje dalyje atliekama tiriamos srities analizė – nagrinėjama Web servisų architektūra ir paslaugų sistemų kūrimo būdai. Pagrindinėje dalyje pateikiamas koordinatoriaus šablonas ir jo veikimo modelis. Projektinėje dalyje pateikiamas koordinatoriaus modeliu paremtos leidybos sistemos projektas. Eksperimentinėje dalyje aprašomas sistemos taikymas realioje aplinkoje. Įvertinimo dalyje pateikiami ekspertų atlikto modelio vertinimo rezultatai. Išvadose pateikiamos atliekant darbą padarytos išvados. Paskutinėje, literatūros dalyje pateikiama darbo rašymo metu naudota literatūra.

Kur darbas pristatytas (konferencijos, straipsniai)

Šis darbas buvo pristatytas 12-oje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje “Informacinė visuomenė ir universitetinės studijos” (IVUS‘07) 2007 m. gegužės 16 d. Pristatymui buvo parengtas straipsnis bei paruošta prezentacija.

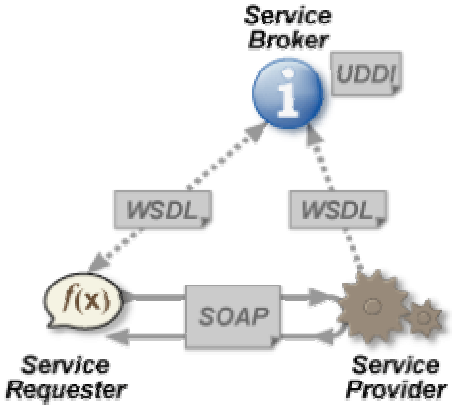
2. Programinės įrangos paslaugų architektūros ir koordinavimo metodų analizė

Analizės tikslas – išanalizuoti Web servisų architektūros principus, realizavimo technologijas, paslaugų koordinavimo mechanizmus ir padaryti prielaidą, koks koordinavimo metodas tinkamiausias nedidelių ir vidutinių paslaugų sistemų valdymui.

2.1. WEB servisų privalumų ir trūkumų analizė

Pasaulinis žiniatinklio konsorciumas (W3C), kuriantis žiniatinklio standartus, Web servisus apibūdina kaip „programinės įrangos sistemą, sukurtą užtikrinti suderintą dviejų mašinų bendradarbiavimą kompiuteriniame tinkle[1]“. Dažniausiai Web servisai yra tiesiog Web API[2] sąsajos, kurias galima pasiekti kompiuteriniu tinklu (pvz. Internetu) ir įvykdyti nutolusioje sistemoje, teikiančioje Web servisų paslaugas.

W3C pateiktas Web servisų apibūdinimas yra universalus, tačiau dažniausiai Web servisai naudoja WSDL apibrėžtas sąsajas, kurios priima SOAP formato XML pranešimus, paprastai perduodamus HTTP protokolu. Pavyzdžiui, WS-I konsorciumas, sukurtas užtikrinti Web servisų suderinamumą ir standartizavimą, pripažįsta tik tokio tipo Web servisus.



1 pav. Web servisų architektūra

Web servisas – abstrakti sąvoka, kuri turi būti įgyvendinta konkretaus agento. Agentas – tai

[2] Application Programming Interface

programa, veikianti asmens ar organizacijos labui[1]. Agentas yra konkreti programinė ar techninė įranga, siunčianti ir gaunanti pranešimus, tuo tarpu Web servisas yra resursas, charakterizuojamas jo teikiamų funkcijų aibės. Pavyzdžiui, galima sukurti agentą, naudojant vieną programavimo kalbą, o vėliau – kitą agentą, naudojant kitokią programavimo kalbą, tačiau taip, kad abu agentai būtų vieno ir to pačio Web serviso įgyvendinimas bei teiktų tą patį funkcionalumą.

Web serviso tikslas – atlikti tam tikras funkcijas jį sukūrusiam asmeniui ar organizacijai. Serviso tiekėjas (*service provider*) – tai asmuo ar organizacija, siūlanti konkretų agentą, įgyvendinantį konkretų Web servisą. Serviso naudotojas (*service requester*) – tai asmuo ar organizacija, norinti pasinaudoti serviso tiekėjo siūlomu Web servisu. Tam naudojamas užklausų agentas (*requester agent*), kuris gali keistis pranešimais su tiekėjo agentu.

Ne visada yra žinoma kur tinkle Web servisą galima rasti. Tokiais atvejais centralizuotoje tinklo vietoje yra sukuriamas tam tikras registras, kuriame klientai gali rasti vietiniame tinkle pasiekiamų Web servisų aprašus bei jų adresus. Tam paprastai naudojama UDDI technologija.

Apsikeitimo pranešimais mechanizmą nustato Web serviso aprašas (*Web Service Description (WSD)*). Tai – dokumentų rinkinys, aprašantis Web serviso sąsają ir semantiką. Web serviso aprašas kuriamas WSDL (*Web Services Description Language*) kalba ir nustato pranešimų formatą, naudojamus duomenų tipus, transporto protokolus bei serializacijos formatus, naudojamus tarp serviso tiekėjo agento bei užklausų agento. Taip pat Web serviso aprašas nustato vieną ar kelis adresus kompiuteriniame tinkle, kur tiekėjo agentas gali būti sužadintas, o taip pat gali aprašyti, koks labiausiai tikėtinas apsikeitimo pranešimais šablonas (*pattern*). Trumpai tariant, Web serviso aprašas apibūdina sąveikos su Web servisu principus ir mechaniką.

Web servisų funkcionavimas yra neatsiejamas nuo jų semantikos. Web serviso semantika – tai tam tikras „susitarimas“ tarp serviso tiekėjo ir naudotojo, nusakantis sąveikos tarp jų tikslus ir pasekmes[10]. Kitaip tariant, Web servisų semantika nusako, kokio atsako galima tikėtis nusiuntus Web servisui vienokį ar kitokį pranešimą.

Tačiau nors šis „susitarimas“ ir nusako pagrindinius abiejų pusių sąveikos principus, jis nebūtinai yra griežtai apibrėžtas. Web servisų semantika gali būti aiškiai aprašyta arba tik numanoma, nusakyta raštu arba žodžiu, aprašyta kompiuterine kalba arba orientuota į

žmoniškąjį supratimą, o taip pat šis „susitarimas“ gali būti teisiškai įformintas arba neformalus.

Nors Web serviso aprašas taip pat nustato tam tikrą „susitarimą“, apibūdinantį sąveiką su Web servisu, tačiau semantika ir aprašas nėra vienas ir tas pats. Web serviso aprašas (*WSD*) nurodo konkrečius būdus kaip komunikuoti su Web servisu, t.y. sąveikos su Web servisu mechaniką. Tuo tarpu semantika nusako šios sąveikos prasmę bei tikslą. Tačiau skiriamoji linija tarp šių aprašų nebūtinai turi būti griežtai nubrėžta. Kuo labiau „semantiškesnė“ kalba naudojama Web servisui aprašyti, tuo daugiau semantinių elementų pereina iš neformalios semantinės pusės į konkretų serviso aprašą ir tuo daugiau darbo, reikalingo užtikrinti sėkmingą Web serviso ir jo klientų sąveiką, gali būti automatizuota.

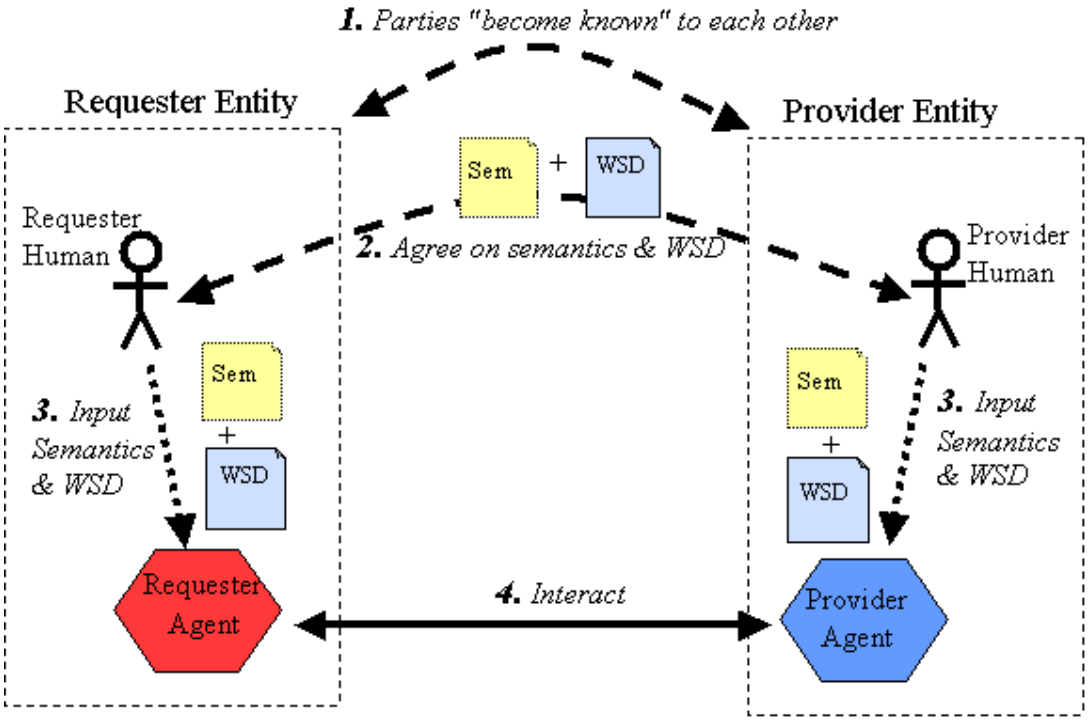
Web serviso naudotojas ryšį su Web servisu gali užmegzti keliais būdais. Tačiau dažniausiai naudojama tokia veiksmų seka (2 pav.):

1. Serviso naudotojas ir serviso tiekėjas užmezga tarpusavio ryšį ir taip tampa žinomi vienas kitam (jei tai neįmanoma, bent viena pusė turi „prisistatyti“ kaip serviso tiekėjas/naudotojas kitai pusei).

2. Web serviso naudotojas ir tiekėjas vienu ar kitu būdu susitaria dėl Web serviso aprašo ir semantikos, apibrėžiančių sąveikos su Web servisu taisykles.

3. Web serviso naudotojo ir tiekėjo agentai vienodai „supranta“ Web serviso aprašą ir semantiką. Paprastai šis supratimas yra agento dalis, suprogramuota jo kūrimo metu.

4. Web serviso naudotojo ir tiekėjo agentai apsikeičia pranešimais. Dažniausiai tai būna SOAP formato pranešimai.



2 pav. Ryšio su Web servisu iniciavimas

2.1.1. Paslaugų architektūra SOA (Service Oriented Architecture)

Paskirstytoji sistema (*distributed system*) – tai tokia sistema, kurioje skirtingi programinės įrangos agentai dirbdami kartu siekia tam tikrų tikslų ir uždavinių. Be to, šie agentai veikia skirtingose aplinkose, todėl tarpusavyje komunikuoti turi naudodamiesi kompiuteriniu tinklu. Tai lemia, kad vidinė komunikacija paskirstytose sistemose yra lėtesnė bei mažiau patikima nei „vientisose“ programose. Tai turi įtakos paskirstytųjų sistemų architektūrai, nes sistemos kūrėjai turi atsižvelgti į padidėjusį ryšio laiką tarp nutolusių sistemos dalių bei įvertinti pilno ar dalinio ryšio nutrūkimo galimybę[26].

Paskirstytosios objektinės sistemos – tai tokios paskirstytosios sistemos, kuriose objektų sukūrimas ir metodų iškvietimas padaromi prienami nutolusioms sistemoms per tam tikrą standartizuotą ar patentuotą mechanizmą. Paprastai (nors ir ne visada) paskirstytosios objektinės sistemos yra nusakomos gana sudėtingos vidinės struktūros, palaikančios jų metodus, objektų, tiksliai apibrėžtos sąveikos tarp objekto ir jį naudojančios programos bei bendrų komponentų architektūros ir sąsajų hierarchijos akcentavimu tarp objekto ir jį

naudojančios programos.

Į servisus orientuota architektūra (SOA) yra atskira paskirstytųjų sistemų architektūros forma. Į servisus orientuota architektūra paprastai pasižymi šiomis savybėmis[18, 43 p]:

- Loginis požiūris. Servisas – tai abstraktus loginis požiūris į konkrečias programas, duomenų bazes, verslo procesus ir pan., nusakantis, kokia tiksliai veikla yra vykdoma.

- Orientacija į pranešimus. Servisas formaliai aprašomas pranešimų, kuriais keičiasi tiekėjo agentas ir užklausų agentas, kontekste, bet ne pačių agentų savybių kontekste. Vidinė agento struktūra (programavimo kalba, kuria jis parašytas, proceso ar duomenų bazės struktūra ir pan.) yra griežtai atskirta nuo SOA. Naudojant SOA, nebūtina ir netgi nereikia žinoti nieko apie vidinę agento struktūrą ir jo veikimo principus. Užtenka žinoti, kokie pranešimai turi būti perduoti vieną ar kitą pusę[21].

- Orientacija į aprašymą. Servisas aprašomas meta-duomenimis, kuriuos gali apdoroti kompiuteris. Aprašas atspindi viešąją SOA pusę: tik tos savybės, kurios pasiekiamos viešai ir yra svarbios serviso naudojimui turėtų būti įtrauktos į aprašą. Taip pat serviso aprašas turėtų tiesiogiai arba netiesiogiai dokumentuoti serviso semantiką.

- Moduliškumas. Servisai turėtų naudoti santykinai mažą operacijų skaičių su santykinai didelėmis ir sudėtingomis operacijomis.

- Orientacija į veiklą tinkle. Servisai turėtų būti orientuoti į veiklą kompiuteriniame tinkle, tačiau tai nėra būtina sąlyga.

- Nepriklausomybė nuo platformos. Pranešimai yra siunčiami nuo platformos nepriklausančiu standartizuotu formatu per servisų sąsajas. Dažniausiai pranešimams siųsti naudojamas XML formatas.

2.1.2. Web servisai ir architektūros tipai

Visos paskirstytosios objektinės sistemos susiduria su įvairiomis problemomis, nulemtomis jų architektūros specifikos:

- Problemos, sukurtos komunikacinės terpės (dažniausiai kompiuterinio tinklo), kuria perduodami pranešimai, nepatikimumo ir užlaikymų joje.

- Problemos, sukurtos bendros atminties srities tarp serviso ir į jį besikreipiančiojo agento nebuvimo.

- Didelis skaičius problemų, įtakotų dalinės nesėkmės situacijų.

- Problemos kaip lygiagrečiai paskirstyti resursus vienu metu jų pareikalaujantiems klientams.

- Problemos, sukeltos paskirstytųjų sistemų jautrumo bet kokiems nesuderinamiems pokyčiams klientų pusėje.

Šios problemos iškyla nepriklausomai nuo to, kokiu būdu (COM/CORBA, Web servisais ar kitu) paskirstytoji objektinė sistema yra įgyvendinta. Jei tokia sistema atitinka pagrindinius paskirstytosios objektinės architektūros reikalavimus, nėra didesnio skirtumo kokiu būdu ji bus įgyvendinta. Web servisams pirmenybė gali būti reikiama tuo atveju, kai reikia nepriklausomybės nuo naudojamos platformos ir programinės įrangos tiekėjo, o veikimo sparta nėra aukščiausio prioriteto tikslas.

Web servisų panaudojimas įgyvendinant paskirstytąją sistemą dar nepaverčia paskirstytųjų objektų architektūros į SOA. Be to, Web servisai nebūtinai yra pats geriausias būdas SOA įgyvendinimui. Gali būti, kad SOAP/WSDL naudojimo privalumai ne visada atsvers lėtesnio veikimo bei kitus trūkumus. Paprastai įgyvendinti SOA naudojant Web servisus geriausia šiais atvejais:

- Sistemose, veikiančiose Internete, kur patikimumas ir veikimo sparta negali būti garantuojama.

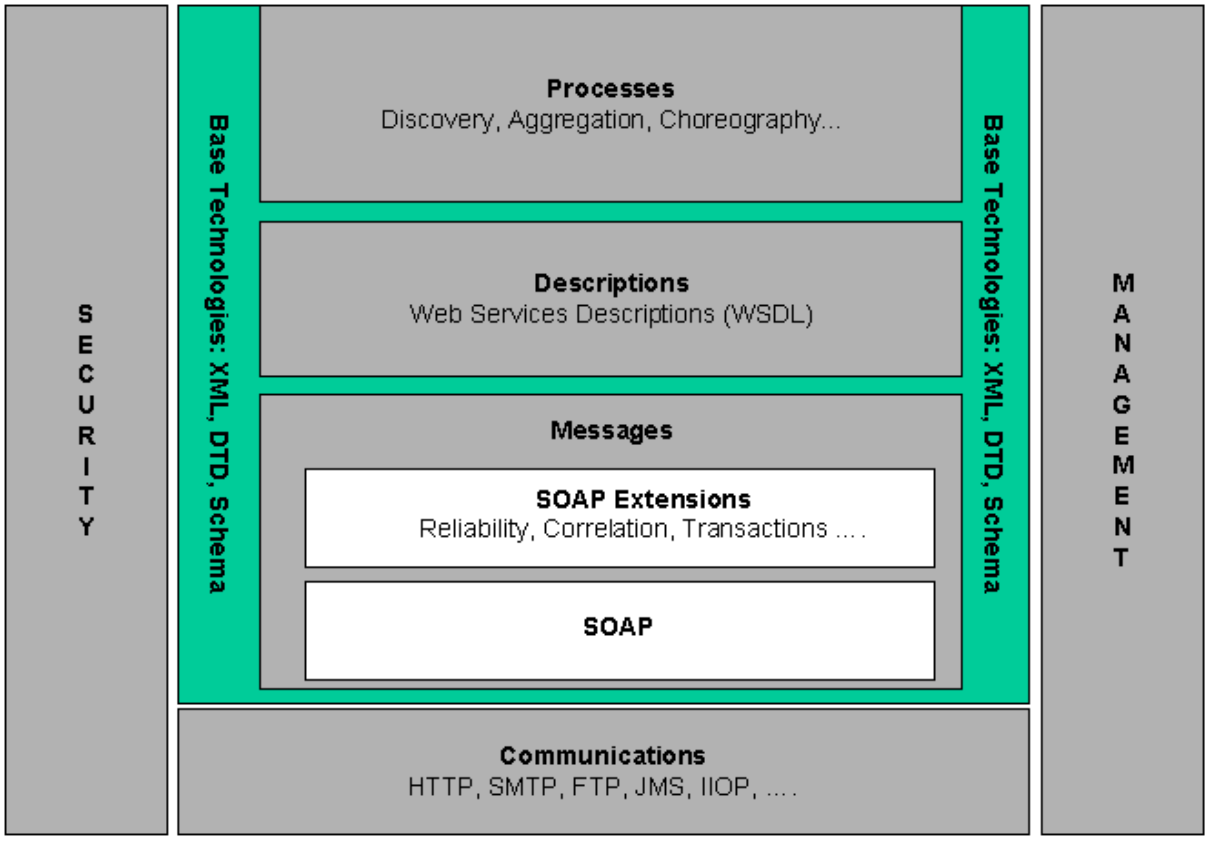
- Sistemose, kuriose nėra galimybės valdyti atnaujinimus, kad visi servisų tiekėjų ir naudotojų agentai būtų atnaujinti kartu.

- Sistemose, kuriose paskirstytųjų sistemų komponentai veikia skirtingose platformose su skirtingų programinės įrangos gamintojų produktais.

- Tokiais atvejais, kai jau sukurta programa turi būti pritaikyta veikti tinkle ir gali būti pateikta Web serviso forma.

2.1.3. Web servisų technologijos

Web servisų architektūrą sudaro daug sudėtingų ir tarpusavyje susijusių technologijų. Yra daug būdų pavaizduoti jų tarpusavio sąryšį ir vienas iš tokių būdų pateiktas 3 pav.



3 pav. Web servisų architektūra

Pagrindinės Web servisų technologijos yra XML, SOAP ir WSDL. Taip pat Web servisų paieškai ir agentų tarpusavio bendravimo būdo nustatymui dažnai naudojama UDDI technologija. Žinoma, yra dar daugybė technologijų, naudojamų su šiomis pagrindinėmis (EbMS–EbXML, EbCPPA–EbXML, kiti OASIS formatai, su saugumu susijusios ir kitos technologijos[28]), tačiau jų nagrinėjimas atsiduria už šio darbo apimties ribų.

Iš esmės Web servisų architektūrą sudaro keturi lygiai[29]:

- Serviso transporto lygis. Šis lygis atsakingas už pranešimų perdavimą tarp tinkle veikiančių programų. Jis remiasi HTTP, SMTP, FTP, o taip pat ir santykinai nauju BEEP (*Blocks Extensible Exchange Protocol*) protokolais[7].

- XML pranešimų lygis. Šis lygis atsakingas už pranešimų kodavimą XML formatu, kad kiekviena apsikeitime pranešimais dalyvaujanti šalis galėtų vienodai juos suprasti. Šiuo metu šiame lygyje veikia XML-RPC, SOAP ir REST protokolai.

- Serviso aprašo lygis. Šis lygis naudojamas konkretaus Web serviso viešai prieinamai sąsajai (*public interface*) aprašyti. Paprastai tam naudojamas WSDL formatas.

- Servisų suradimo (*discovery*) lygis. Šis lygis atsakingas už Web servisų centralizaciją vienoje tinklo vietoje. Web servisai gali registruoti savo vietą tinkle ir aprašą konkrečioje tarnybinėje stotyje taip padėdami sužinoti kokie servisai yra prieinami konkrečiame kompiuteriniame tinkle. Servisų suradimo lygyje veikia UDDI technologija.

Be jau minėtų technologijų, Web servisų architektūra apima ir daugiau, dar gana naujų, protokolų – BPEL, SOAP-DSIG ir kt.

2.1.3.1. XML

Dėl savo universalumo XML technologija tinka ir yra naudojama daugelyje sričių. Web paieškoje bei Web uždavinių automatizavime naudojant XML galima nurodyti dokumente saugomos informacijos tipą, taip pagerinant grąžinamų rezultatų tikslumą. Pavyzdžiui, ne XML knygų kataloge knygos autorius, pavadinimas, leidykla ir pan. nebus išskirti pagal informacijos tipą. Tuo tarpu naudojant XML įmanoma paiešką vykdyti tik tarp autoriaus tipo žymių. XML taip pat gerai tinka elektroniniam verslui, kur informacijos perdavimas ir apdorojimas yra labai svarbus. Be to, XML naudojamas programų, dirbančių su metaduomenimis (pvz. UML modeliavimo programos naudoja XML modelių savybėms aprašyti), mobiliuose įtaisuose veikiančių programų (WML, VoiceXML ir pan.) bei apskritai programų, dirbančių su duomenimis ir juos apdorojančių. Labai dažnai XML naudojamas Web leidyboje, kur XML standartu saugomus duomenis labai lengva atvaizduoti skirtinguose įtaisuose ir aplinkose, XSL/XSLT procesoriaus pagalba pagal saugomus duomenis sugeneravus kiekvienam klientui tinkamą atvaizdavimo kodą.

XML puikiai tinka ir Web servisams. Vietoje to, kad atskiriems servisams pagal kiekvieno jų poreikius kurti skirtingus komunikacijos mechanizmus ar net protokolus, XML suteikia standartais paremtą, lankstų ir pagal apibrėžimą laisvai keičiamą duomenų formatą.[14]

XML turi daug pranašumų, lyginant su kitomis technologijomis. Keletas iš šių pranašumų yra tokie:

- Žmogui suprantamos kalbos naudojimas. XML aprašytas dokumentas paprastai yra nesunkiai suprantamas žmogaus. Be to, rašyti XML kodą yra ne sunkiau nei HTML.[3]

- Visiškai nepriklausomas nuo platformos formatas. Bet kuri programa, sugebanti apdoroti XML, veikianti bet kurioje operacinėje sistemoje, gali apdoroti XML forma pateiktą informaciją. Be to, nepriklausomybė nuo platformos padaro XML kur kas mažiau jautresnį technologijų pokyčiams[5].

- Išplečiamumas. XML sukurta taip, kad bet kada galima pakeisti, pridėti ar pašalinti duomenis aprašančias žymes (*tags*) ir atributus. Jei duomenų struktūra keičiasi, labai paprasta tiesiog pakeisti XML struktūrą taip, kad ji atspindėtų įvykusius pokyčius.

- Žymių ir duomenų sąsajos. Skirtingai nei HTML, XML žymių pavadinimai atspindi su jomis susijusių duomenų reikšmę. Kiekviena XML žymė yra prieš ir po su ja susijusiais duomenimis. Taip XML tampa kur kas lengviau skaitomas žmonėms. Be to, griežta XML struktūra supaprastina mašininį XML kodo apdorojimą bei palengvina darbą su duomenimis.

- Apsikeitimas sudėtingomis duomenų struktūromis tarp sistemų, naudojančių skirtingus duomenų formatus. Naudojant XML, skirtingos programos, naudojančios skirtingus vidinius duomenų formatus, gali sėkmingai apsikeisti informacija, neprarasdamos duomenų.

- Lankstumas saugojant duomenis. Duomenys duomenų bazėje gali būti saugomi XML formatu arba XML formato dokumentas iš konkrečių duomenų gali būti suformuojamas realiu laiku tada, kai to reikia.

- Unikodo (*Unicode*) palaikymas. Unikodo palaikymas leidžia naudoti praktiškai bet

kokią informaciją parašytą bet kuria žmonių vartojama kalba. Pavyzdžiui, XML dokumentas gali turėti lotyniškomis raidėmis užrašytą antraštę (*header*), japoniškais hieroglifais užrašytas žymes ir kirilica įrašytus duomenis ir visiškai atitikti standartus, keliamus tvarkingam XML dokumentui.

- Duomenų atvaizdavimas. Naudojant XML galima atvaizduoti visas pagrindines kompiuteriuose naudojamas duomenų struktūras: įrašus, sąrašus ir medžius. Hierarchinė XML struktūra tinkama daugumai (nors ir ne visiems) dokumentų.

- Paplitimas. Dėl savo universalumo XML plačiai naudojamas duomenų saugojimui ir apdorojimui įvairiose aplinkose. Be to, XML paplitimą nemaža dalimi lėmė ir tai, kad jis paremtas atvirais ir nemokamais tarptautiniais standartais.

Tačiau, žinoma, XML turi ir trūkumų. Keletas jų būtų tokie:

- XML sintaksė yra pernelyg perkrauta ir besikartojanti lyginant su dvejetainiais formatais, atvaizduojančiais tuos pačius duomenis. Šis perkrovimas gali neigiamai įtakoti (ir dažnai įtakoja) programų veikimo efektyvumą, gerokai pakeldamas duomenų saugojimo, perdavimo ir apdorojimo kaštus[20].

- Nėra vidinio duomenų tipų palaikymo. Skaitinė, tekstinė, loginė ir pan. informacija XML neišskiriama.

- Hierarchinis XML modelis yra gana ribotas, lyginant su kai kuriais kitais duomenų atvaizdavimo modeliais (pvz. reliaciniu). Be to, gana sudėtinga pavaizduoti nehierarchinius ryšius tarp duomenų.

- Kartais XML vardų sritis (*namespace*) gali būti sudėtinga naudoti ir teisingai apdoroti.

Web servisams svarbiausi yra šie XML aspektai: pati XML sintaksė bei XML duomenų aibės (*XML Infoset*), XML schemos (*XML Schema*) ir XML vardų srities (*XML Namespace*) sąvokos.

XML duomenų aibė pats savaime nėra duomenų formatas, o tik formali informacinių vienetų aibė, apibūdinanti XML dokumento abstraktų duomenų modelį. XML duomenų aibės specifikacijos duomenų aprašai yra skirti naudoti kitose specifikacijose, kurios tiesiogiai

kreipiasi į duomenis XML dokumente.

XML duomenų aibės informacijos aprašų serializacija gali būti vykdoma XML 1.0 pagalba. Tačiau tai nėra privalomas reikalavimas – gali būti panaudoti ir kiti formatai. Ateityje vis dažniau gali būti naudojamas XML duomenų aibės dvejetainis pavidalas, kuris būtų efektyvesnis bei labiau tinkamas komunikacijoms tarp kompiuterių.

2.1.3.2. SOAP

SOAP (*Simple Object Access Protocol*, nors dabar vis dažniau vartojamas ir *Service Oriented Architecture Protocol* terminas) – tai protokolas, skirtas XML formato pranešimų apsikeitimui kompiuterių tinklais atlikti, paprastai naudojantis HTTP protokolo teikiamomis galimybėmis. Tai – lengvai išplečiama ir lanksti sistema, naudojama XML pranešimų enkapsuliacijai. SOAP sudaro vieną iš pamatinių Web servisų architektūros sluoksnių, ant kurio laikosi aukštesnio lygio sluoksniai. Nuo 1.2 versijos SOAP standartas antraščių pagalba leidžia agentams apsikeisti savo galimybių aprašais. Tai užtikrina geresnį suderinamumą tarp agentų.

Pirmoji SOAP 1.2 specifikacijos dalis[30] apibrėžia XML paremtą apsikeitimo pranešimais sistemą: apdorojimo (*processing*) ir išplečiamumo (*extensibility*) modelius. Specifikacija nurodo, kad SOAP pranešimai gali būti perduodami įvairiais tinklo protokolais: HTTP, SMTP, FTP, RMI/IIOP ar uždarais patentuotais (*proprietary*) protokolais.

Antroji specifikacijos dalis aprašo tris papildomus komponentus: kodavimo taisyklių, leidžiančių pavaizduoti programos nustatytus duomenų tipus, rinkinį, susitarimą, kaip atvaizduoti RPC užklausas ir atsakymus bei SOAP naudojimo su HTTP/1.1 taisyklių rinkinį.

Nors SOAP gali veikti kelių tinklo technologijų pagrindu, dažniausiai tam naudojamas HTTP protokolas. Iš esmės jis paprastai naudojamas dėl gero suderinamumo su šiandienos Interneto infrastruktūra, o konkrečiau dėl to, kad lengvai apeina ugniasienes. Tai vienas iš SOAP privalumų prieš tokius paskirstytuosius protokolus kaip GIOP/IIOP ar DCOM, kuriuos ugniasienės paprastai blokuoja.

Be to, HTTP naudojimas leidžia nekurti dar vieno atskiro protokolo saugiam duomenų perdavimui užtikrinti. SOAP gali naudotis jau esamu HTTPS protokolu, nes tai iš esmės yra

tas pats HTTP protokolas taikomajame tinklo lygyje, besinaudojantis šifruotu transporto lygio protokolu.

Žinoma, SOAP turi ir trūkumų. Pagrindiniai jų yra šie (kadangi SOAP yra Web servisų dalis, kai kurie iš šių trūkumų iš dalies sutampa su Web servisų trūkumais):

- Dėl tekstinio ir perteklinio XML formato SOAP gali būti gerokai lėtesnis nei dvejetainiai formatai. Tačiau, jei siunčiami nedideli pranešimai, skirtumas tampa praktiškai nejuntamas. Be to, tam tikrais atvejais labai efektyvus tampa vis labiau pripažįstamas dvejetainis XML (*binary XML*) formatas[31].

- Egzistuoja daug SOAP realizacijų, kurios riboja persiunčiamos informacijos kiekį.

- Tiek SOAP, tiek HTTP protokolai priklauso aukščiausiam – taikomajam tinklo lygiui. Tačiau SOAP naudoja HTTP kaip transporto lygio protokolą SOAP pranešimams perduoti. Tačiau tai gali sukelti problemų, nes HTTP nėra transporto lygio protokolas. Todėl nėra būdo sužinoti ar HTTP naudojamas metodas informacijai perduoti yra tinkamas atliekamai operacijai. Be to, toks veikimo principas apsunkina programos veikimo analizę, neigiamai įtakoja Web servisų darbo efektyvumą (pavyzdžiui, jei naudojama POST užklausa, kai HTTP lygyje geriau tiktų GET užklausa) bei gali bei gali įtakoti nekorektišką programos veikimą.

SOAP specifikacijos nereikia painioti su SOAP priemonėmis. Dauguma programuotojų patys nerašo SOAP kreipinių, tačiau naudojasi jau parengtomis priemonėmis (*toolkits*). Deja, tokių priemonių yra gana daug ir skirtingai įgyvendintų ir ne visos jos yra pilnai suderinamos viena su kita. Tačiau tai jau ne pačios SOAP specifikacijos, o jos netikslaus įgyvendinimo trūkumas.

2.1.3.3. WSDL

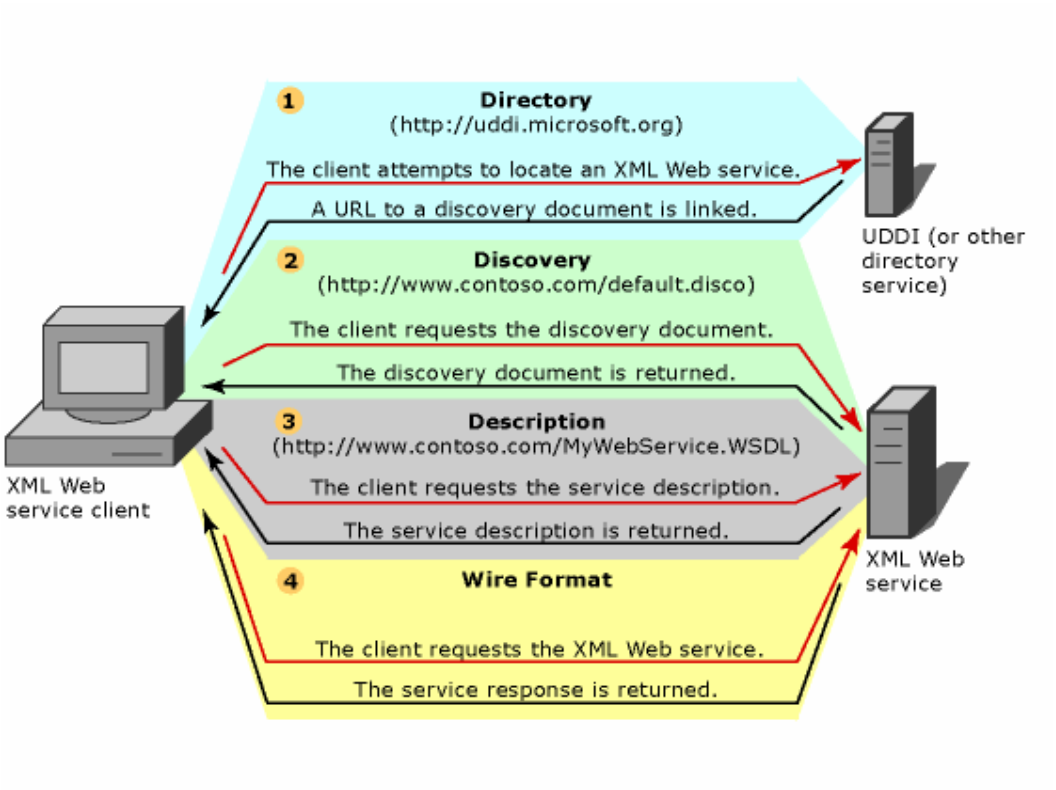
WSDL (*Web Services Description Language*) yra XML pagrįsta kalba, skirta Web servisams aprašyti. Paprastai aprašomi Web servisų pranešimai ir prievadai (*ports*). Pranešimai ir prievadai aprašomi abstrakčiai, atskirai nuo konkrečių jų panaudojimo atvejų. Tai leidžia pakartotinai panaudoti šiuos aprašus vėliau.

Prievadas yra apibrėžiamas tinklo adresą susiejant su naudojama jungtimi (*binding*). Prievadų visuma apibrėžia servisą. Taip pat WSDL aprašo pranešimus, kuriais tarpusavyje bendrauja serviso tiekėjo ir užklausų agentai. WSDL pranešimai – tai abstraktūs pranešimuose naudojamų duomenų aprašymai, kai tuo tarpu prievadai – tai serviso palaikomų operacijų abstrakčios kolekcijos. WSDL viešą Web serviso sąsają aprašo pranešimus bei operacijas susiedama su konkrečiu tinklo protokolu ir pranešimų formatu.

Užklausų agentas, jungdamasis prie Web serviso, gali perskaityti WSDL aprašą ir taip sužinoti, kokias paslaugas teikia Web servisas. Bet kokie papildomai Web serviso naudojami duomenų tipai XML schemos pavidalu yra saugomi WSDL faile. Perskaitęs šį aprašą, klientas, naudodamasis SOAP, gali iškviesti bet kurią iš WSDL aprašytų Web serviso funkcijų.

2.1.3.4. UDDI

UDDI (*Universal Description Discovery and Integration*) – tai OASIS konsorciumo, atsakingo už Web servisų ir elektroninio verslo standartų plėtrą, remiamas standartas, skirtas viešai publikuoti Web servisų aprašams ir aprašantis kaip Web servisai bei programinė įranga sąveikauja tarpusavyje Internetu. Tai lyg tam tikras viešas Web servisų sąrašas, į kurį kreipiantis galima sužinoti kokie Web servisai jame yra, kokias funkcijas jie atlieka bei kaip juos pasiekti. UDDI informacija perduodama SOAP pranešimais, o Web servisai pasiekiami per WSDL dokumentus, nurodančius konkretaus sąraše esančio Web serviso naudojamus duomenų formatus bei sąsajas.



4 pav. Web servisų veikimas naudojant UDDI

Dažnai UDDI yra naudojama kompanijų ir organizacijų viduje, taip palengvinant vidinių Web servisų suradimą ir naudojimąsi jais. Tačiau nepaisant jos paplitimo, daugelis autorių nelaiko UDDI priklausančia pagrindinėms Web servisų technologijoms[22, 71 p.].

2.1.4. Web servisų naudojimas

Web servisų privalumai:

- Santykinis paprastumas. Web servisų veikimas pagrįstas Web technologijomis, kurios, kaip parodė laikas, tinka tiek nedidelėms, tiek milžiniškoms sistemoms. Todėl Web servisas gali būti ir visai nedidelė programa, užimanti mažą atminties dalį. Be to, Web servisų naudojamas HTTP protokolas leidžia gana lengvai išvengti ugniasienių problemos, nes daugumoje ugniasienių HTTP prievadai paprastai būna atidaryti.

- Paplitimas. Kadangi pasaulinis žiniatinklis šiuo metu yra visuotinai paplitęs, o Web servisai dažniausiai veikia naršyklėje, galima sakyti, jog Web servisais nesudėtinga

naudotis visur, kur yra tinklo prieiga.

- Nepriklausomybė nuo naudojamos programavimo kalbos. Naudodami XML tarpusavyje „bendrauti“ gali bet kuria kalba (C, C++, Java, .NET, Python, Perl, ...) parašyti Web servisai. Be to, XML yra nepriklausoma ne tik nuo programavimo kalbos, bet ir programavimo modelio – visiškai nesvarbu, ar Web servisas sukurtas objektiniu programavimo principu, ar neobjektiniu. Skirtingos programos gali būti visiškai suderinamos jei tik jos tenkina Web serviso sąsajos reikalavimus.

- Moduliškumas ir pasiskirstymas tinkle. Ši Web servisų savybė gali būti laikoma ir pliusu, ir minusu. Platus išsidėstymas tinkle (pats servisas ir juo besinaudojantys užklausų agentai gali būti išsimėtę po visą pasaulį) suteikia globalumo ir lengvo naudojamumo bei pasiekiamumo požymių. Tačiau kartu tai gerokai sulėtina pranešimų perdavimo spartą bei sumažina šio perdavimo patikimumą.

- Programinės įrangos gigantų palaikymas. Šiuo metu Web servisai vis dažniau naudojami didžiųjų korporacijų ir verslo organizacijų duomenų perdavimui ir apdorojimui vidiniuose tinkluose. OASIS (*Organization for the Advancement of Structured Information Standards*) konsorciumas yra atsakingas už Web servisų, elektroninės komercijos, saugumo, XML duomenų apdorojimo ir kitų versle naudojamų standartų, į kuriuos kasmet investuojama milijardai JAV dolerių, kūrimą ir palaikymą.

Be minėtų pranašumų, Web servisai turi šiuos pagrindinius trūkumus:

- Saugumo bei privatumo užtikrinimas. Šiuo metu nėra plačiau paplitusio standarto, užtikrinančio Web servisų bei jų perduodamų duomenų saugumą. Žinoma, jei toks saugumas reikalingas, programų kūrėjai visada gali sukurti savo sprendimą, tačiau dėl standarto nebuvimo jis greičiausiai nebus suderinamas su atitinkamais kitų programuotojų sprendimais. Tokios tradiciškai naudojamos saugumo priemonės kaip IPSec, VPN, SSL/TLS ar S/MIME yra skirtos taškas-taškas (*point-to-point*) ryšiui ir todėl Web servisams yra nepakankamos. Reikalingas standartas, kuris užtikrintų pakankamus autentifikacijos ir autorizacijos mechanizmus, duomenų bei pranešimų perdavimo tarp Web servisų ir klientų integralumą bei konfidencialumą, saugumo privilegijų suteikimą ir kt.

- Efektyvus pranešimų perdavimas ir veikimo sparta. Kadangi Web servisai yra paskirstytoji sistema, veikianti kompiuteriniame tinkle, kuris gali būti lėtas ir nepatikimas, efektyvus pranešimų perdavimas ne visada gali būti užtikrinamas. Paprastai Web servisai veikia lėčiau ar net gerokai lėčiau nei atitinkamos programos, pranešimus perduodančios vietinėje kompiuterinėje sistemoje, pranešimais besikeičiančios per bendrą sistemos atmintį.

- Patikimumas. Pranešimų perdavimo kompiuteriniu tinklu patikimumas gali nukentėti ne tik dėl problemų tinkle, bet ir dėl netinkamo pranešimo suformavimo. Pranešimas ne tik turi būti gavėjo gaunamas lygiai toks, koks jis buvo išsiųstas siuntėjo, bet ir griežtai atitikti apibrėžtą formatą bei veiklos taisykles, kad abi pusės jį galėtų suprasti vienareikšmiškai. Turint omenyje, kad Web servisai paprastai yra plačiai paplitę tinkle ir agentai parašyti skirtingomis kalbomis ir veikia daugelyje skirtingų platformų, šių sąlygų kartais yra gana sunku laikytis.

- Valdymas. Web servisų valdymas apima jų stebėjimą (*monitoring*), kontrolę bei ataskaitų apie įvairius parametrus (pvz. Web servisų darbo kokybę, našumą, pasiekiamumą, naudojimą ar pan.) formavimas. Tam įgyvendinti paprastai reikalingi įvairūs skaitikliai ir metrikos, o taip pat ir būdas nustatyti bendras taisykles kaip tai padaryti (*policies*) pasirinktiems Web servisams. Šiuo metu yra dirbama ties keliais pasiūlymais, tačiau patvirtinto standarto, apibrėžiančio Web servisų valdymą, nėra.

- Pranešimų stebėjimas (*tracking*). Daugelyje verslo scenarijų pranešimų srauto stebėjimas labai dažnai yra tiesiog būtinas. Tai padeda išvengti nesusipratimų ar nesąžiningumo atvejų („Kodėl jūs dar nesumokėjote? Mes jau sumokėjome!“) ar tiesiog sužinoti kas konkrečiu metu vyko tarp Web serviso ir konkretaus agento. Tačiau Web servisai neturi pranešimų sekimo mechanizmo. Tai laikoma dideliu Web servisų trūkumu, o tokio mechanizmo sukūrimas – vienu iš prioritetų tolimesniame Web servisų tobulinime ir vystyme.

Web servisų pasirinkimą lėmė tai, kad ši architektūra ypač gerai tinka kuriamai programų sistemai įgyvendinti. Kadangi leidybos agentūros informacine sistema naudojasi autoriai ir recenzentai, kurie gali būti bet kurioje pasaulio vietoje, pasaulinio žiniatinklio prieinamumas ir paplitimas ypač gerai pasitarnauja programų sistemos poreikiams.

Web servisų moduliškumas leidžia išdėstyti sistemos komponentus skirtingose kompiuterinio

tinklo dalyse, taip padidinant patikimumą (paskirstytoji sistema nėra priklausoma nuo centrinės tarnybinės stoties patikimumo) bei pasiekiamumą.

Vartotojai informacinę sistemą gali ne tik pasiekti iš bet kurios pasaulio vietos, kur yra Interneto ryšys, tačiau dėl Web servisų paprastumo tai padaryti jiems reikia minimalių pastangų. Vartotojams nereikia diegti jokios programinės įrangos – prie sukurtos informacinės sistemos galima prisijungti naudojant bet kurią šiuolaikinę operacinę sistemą ir W3C standartus atitinkančią naršyklę.

2.2. Paslaugomis realizuojamų veiklos procesų vykdymo kalbos ir įrankių analizė

2.2.1. Kas yra BPEL

BPEL (angl. Business Process Execution Language), taip pat žinoma kaip BPEL4WS ir WSBPEL - verslo procesų vykdymo kalba, pagrįsta XML technologija, kuri buvo sukurta tam, kad standartizuotų integracijos logiką ir procesų automatizavimą tarp Web servisų[32]. BPEL siūlo lankstaus bendravimo tarp sistemų standartą.

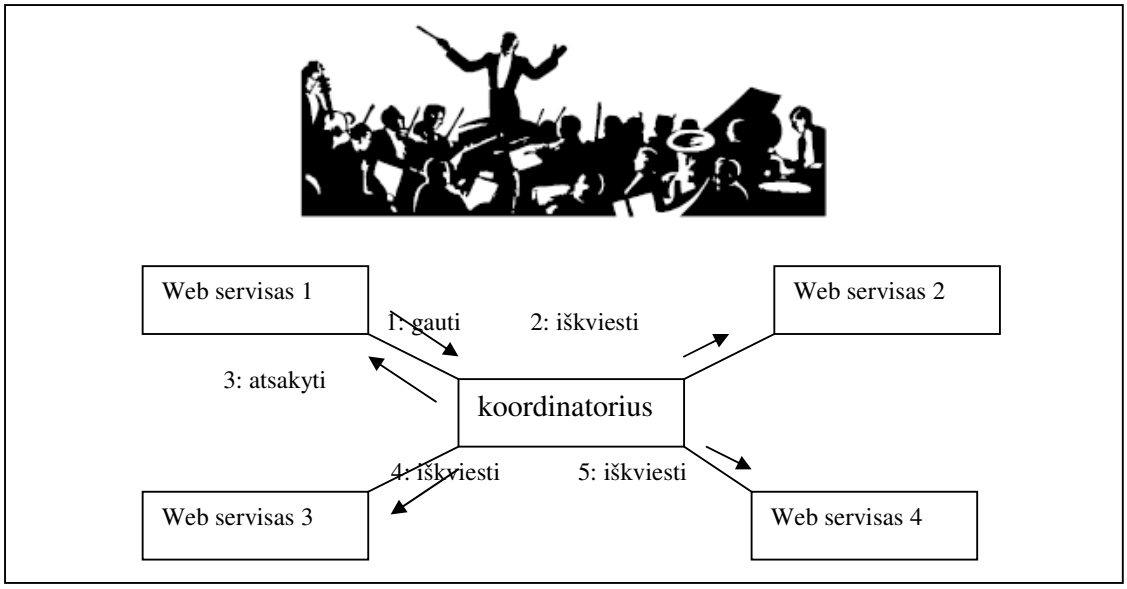
BPEL taikoma didelio masto programavimui[33] (kai sistemą kuria daugelis žmonių ir tai užtrunka, sistemos kodo neįmanoma suprasti be principo „skaldyk ir valdyk“).

BPEL naudojama apibrėžti ir valdyti orkestravimą (orchestration) ir procesus[34]**.**

2.2.1.1. Orkestravimas

Orkestravime pagrindinis servisas (koordinatorius) kontroliuoja visa bendradarbiavimą.
Naudojant BPEL pagrindinis servisas yra BPEL procesas, nes BPEL procesai yra įgyvendinti kaip servisai. BPEL procesas kontroliuoja visą seką ir iškviečia bendradarbiaujančius procesus kai reikia.

Toks BPEL procesas gali būti sinchroniškas arba asinchroniškas.



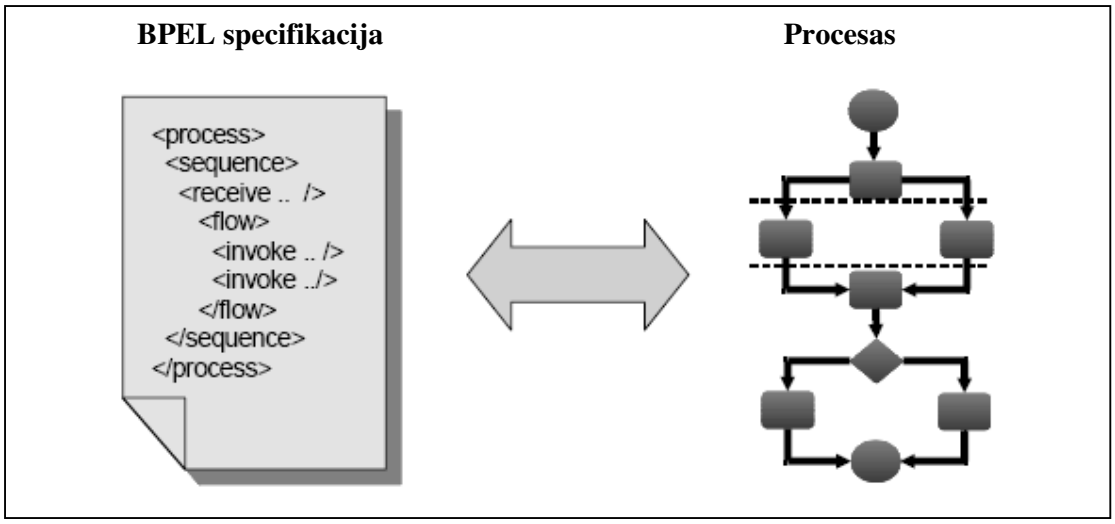
5 pav. Orkestravimo struktūra

2.2.1.2. Technologinis pagrindas.

BPEL technologiniai pagrindai yra XML ir Web servisai. XML – standartinė žymėjimo kalba. BPEL naudoja XMP procesų aprašymams koduoti ir perduoti.

Taigi BPEL kalba pasako kaip siusti XML pranešimus nutolusiems servisams, valdyti XML duomenų struktūras, priimti XML pranešimus sinchroniškai/asinchroniškai, valdyti įvykius ir išimtis.

2.2.1.3. Struktūra



6 pav. BPEL struktūra

Proceso pavyzdys:

```
<process>
   <receive createInstance="yes" />
   <switch>
     <case name="usa">
       <invoke name="install_firmware" />
     </case>
     <case name="france">
       <invoke name="install_firmware" />
     </case>
   </switch>
   <reply variabele="status" />
</process>
```

2.2.2. Kada reikia BPEL

Turint didelę sistemą, ir ją keičiant, papildant naujomis funkcijomis, procesas tampa sudėtingas ir brangus. Dėl to ir buvo sukurta technologija, palengvinanti integraciją, paverčianti vystymo procesą automatiniu.

2.2.2.1. BPEL palaikymas

BPEL buvo pristatytas keleto didesnių kompanijų. Šalia jų yra nemažai kompanijų, kurios dirba su BPEL programine įranga.

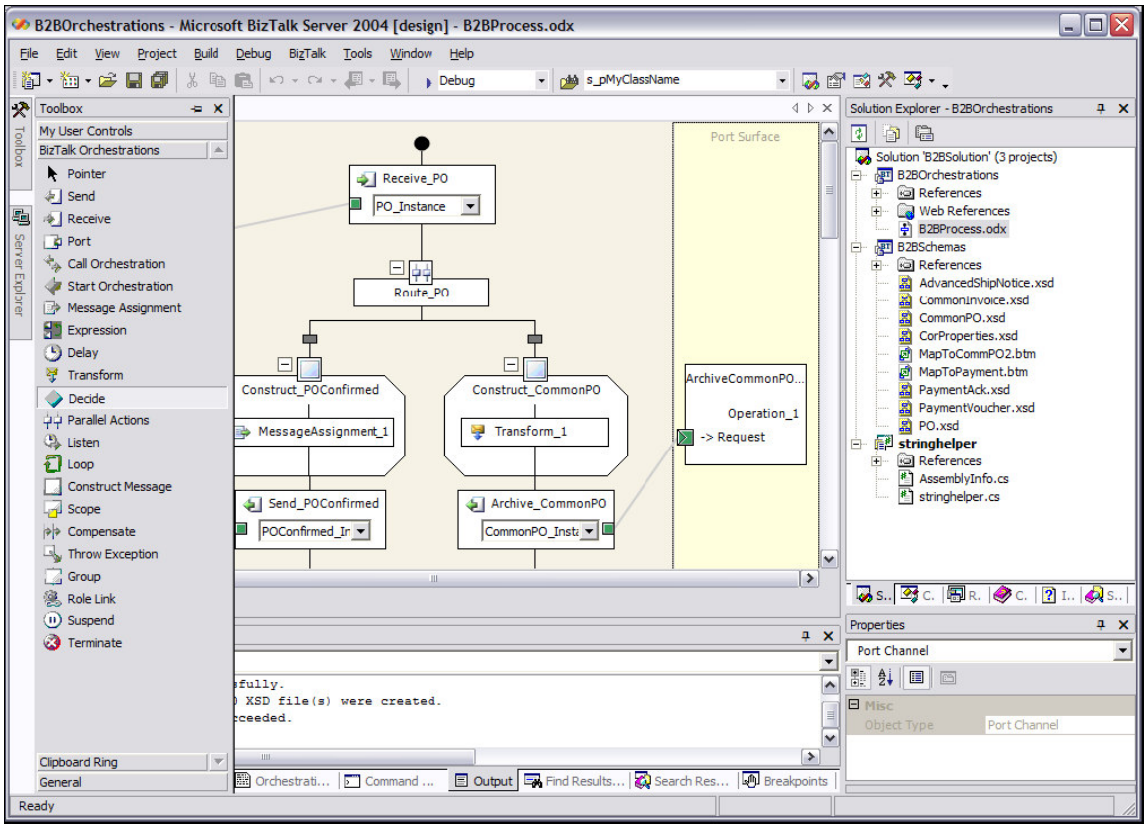
BPEL iniciatoriai[35]**:**

- BEA - www.bea.com,
- IBM - www.ibm.com,
- Microsoft - www.microsoft.com,
- SAP AG - www.sap.com,
- Siebel - www.siebel.com,
- Oracle - www.oracle.com.

BPEL redaktoriai[36]

- Sukurti ne tik PĮ žinovams, bet ir kitiems darbuotojams, neišmanantiems programavimo kalbų..
- Kūrimo procesai vyksta ir grafiniame ir kodo vaizde.
- Daugeliuose redaktorių nėra būtina liesti pačio kodo.
- Redaktoriai taip pat pateikia pagalbos įrankius:
    - proceso siuntimo į BPEL variklį,
    - BPEL kodo validavimo,
    - Web serviso įkėlimo,
    - procesų derinimo.

Redaktoriaus pavyzdys:



7 pav. Redaktoriaus vaizdas

2.2.2.2. BPEL varikliai

BPEL procesų aprašymai siunčiami į BPEL variklį, kuris vykdo procesus. Procesas gali būti iškviečiamas kaip Web servisas. Kai kurie varikliai turi savo testavimo aplinkas.

Keletas savybių:

- Procesų suderinimas (debugging).
- Procesų dehidratacija (ilgai veikiančių procesų saugojimas duomenų bazėje).
- Procesų stebėjimas.
- Versijų kontrolė.
- Išimčių valdymas ir klaidų pranešimai.

Keletas BPEL variklių[37]:

1. lent. BPEL varikliai

	BPEL Process	WebSphere	BPWS4J	Biztalk	ActiveBPEL	Twister	Bexee

	Manager						
Organizacija	Oracle	IBM	IBM	Microsoft	Active Endpoints		
Platforma	J2EE	J2EE	J2EE	.NET	J2EE	J2EE	J2EE
Palaikymas	telefonu, internetu	telefonu, internetu	internetu	telefonu, internetu	internetu	telefonu	internetu
Licenzijos tipas	komercinė	komercinė	komercinė	komercinė	nemokama	nemokama	nemokama

Visgi nekomerciniai varikliai nėra taip gerai išvystyti kaip komerciniai.

2.2.3. BPEL privalumai ir trūkumai

BPEL privalumai

- Plačiai patvirtintas standartas (daug dokumentacijos).
- Nepriklauso nuo platformos.
- Gerai tinka didelėms sistemoms.
- Lengva pridėti naujus modulius.

BPEL trūkumai

- BPEL netinka mažoms sistemoms.
- Reikia įsisavint BPEL kalbą.
- Sudėtinga įdiegti.
- BPEL nekomerciniai varikliai nėra taip gerai išvystyti kaip komerciniai – aukšta kaina.

Galima teigti, kad jei sistema didelė, sudėtinga, BPEL naudojimas pasiteisina. Jei sistema maža – ne.

2.3. Analizės išvados

Šiame skyriuje buvo apžvelgta ir išanalizuota Web servisų bei BPEL kalbos teorinė dalis: jų sudėtis, architektūra, veikimo principai bei pagrindas. Taip pat buvo apžvelgtos BPEL, Web servisų bei juos palaikančių technologijų (XML, SOAP, WSDL ir kt.) stipriosios bei silpnosios pusės bei padaryta BPEL variklių lyginamoji analizė.

Atlikta analizė leido padaryti išvadą, kad kuriama sistema turi būti pakankamai didelė, kad BPEL variklio taikymas būtų efektyvus. Mažesnėms sistemoms tikslinga išbandyti teorinį koordinatoriaus šabloną, kuris reikalauja mažiau investicijų, tačiau turėtų būti efektyvesnis negu chaotiškas() kūrimas.

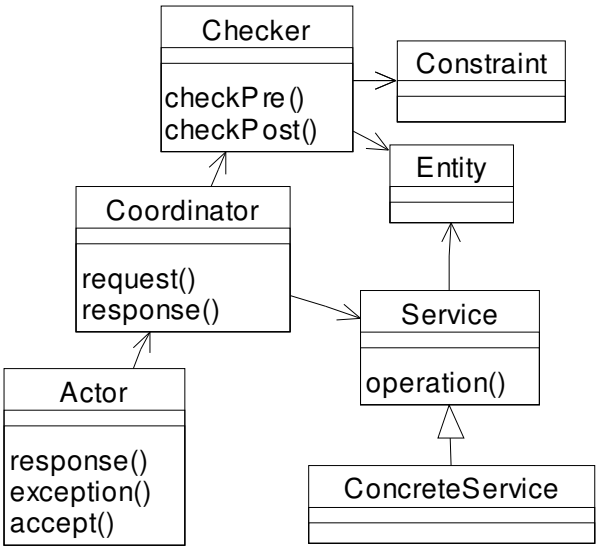
Koordinatoriaus šablono išbandymui buvo pasirinkta leidybos paslaugų sistema, kadangi pastebimas tokio tipo sistemų trūkumas ir tai yra tipiška paslaugų sistema, kuriai geriausiai tinka Web servisų architektūra. Web servisai sistema naudotis leidžia iš bet kurios pasaulio vietos, o naudojimuisi kelia ypač mažus reikalavimus – pakanka kompiuterio su Interneto naršykle ir prieigos prie pasaulinio tinklo. Toks Web servisų globalumas, paprastumas naudotis bei moduliškumas labai gerai tenkina specifinius leidybos agentūrų ir kitų panašių sistemų poreikius.

3. Koordinatoriaus modelis Web servisų sistemų kūrimui

Šiame skyriuje pateikiamas Web servisų koordinatoriaus šablonas, kurį galėtų naudoti daugelis projektuotojų, siekiantys kurti informacines sistemas, komponuodami jas iš elementarių, būsenų nesaugančių Web servisų. Koordinatoriaus modelis buvo pasiūlytas Informacijos sistemų katedroje[23], šiame darbe siekiama atlikti jo eksperimentinę realizaciją ir atlikti tyrimą dėl modelio tinkamumo Web servisų sistemoms kurti. Koordinatoriaus šablonas ir sukurtas servisas gali būti alternatyva Web servisų varikliams (pavyzdžiui, BPEL), juos galėtų naudoti informacinių sistemų kūrėjai.

3.1. Koordinatoriaus šablonas

Koordinuojama Web servisų sistema – tai tokia sistema, kurioje galima išskirti dvi sistemos dalis: atskirus Web servisus ir juos koordinuojantį pagrindinį Web servisą. Pagrindinė sistema atskiriems prisijungusiems servisams teikia paslaugas, atlieka kontrolę ir pan. Pagrindinio, atskirus Web servisus koordinuojančio serviso pagrindas yra koordinatorius. Juo ir remiasi pagrindinis Web servisas bei iš esmės visa programų sistema. Koordinatoriaus šablonas atrodo taip:



8 pav. Būsenų koordinatoriaus šablonas

Koordinatorius gauna ir reaguoja į dviejų tipų pranešimus – užklausas (request) ir atsakus (response). Užklausos yra gaunamos iš aktorių ar paslaugų, kurie siekia iškviesti konkrečios

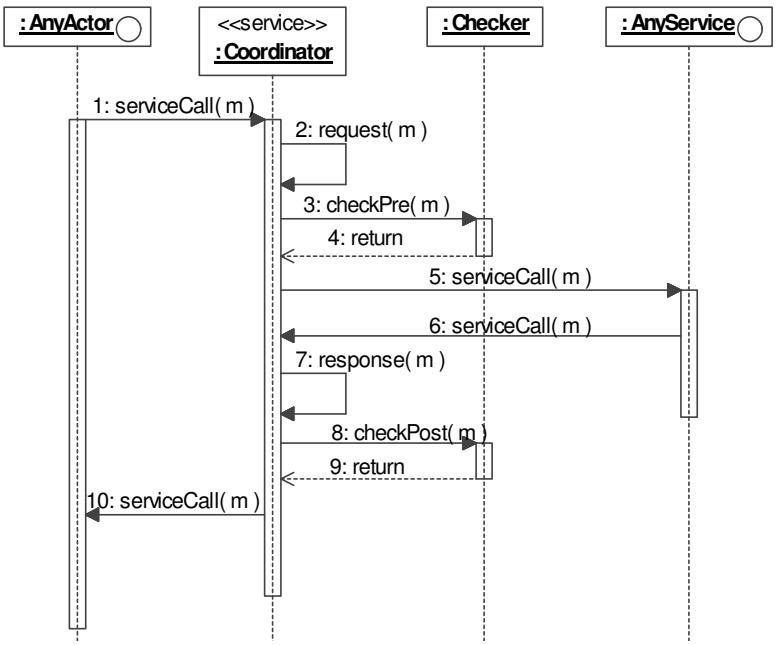
paslaugos operaciją. Atsakas gaunamas iš paslaugos, kuri tą operaciją atliko. Kadangi koordinatorius yra paslauga, bendraujanti pranešimais paremtu bendravimo modeliu, į jį kreipiamasi iškviečiant operaciją serviceCall, kurios argumentas – pranešimas (MessageEntity), kur nurodyta pageidaujama iškviesti operacija, jos argumentai ir jų reikšmės, kas ją nori iškviesti ir t.t.

Gavęs užklausos pranešimą, koordinatorius pirmiausia iškviečia tikrintojo klasę (Checker), kad būtų patikrinta reikalaujamos operacijos prieš sąlyga. Operacijų prieš ir sąlygos saugomos apribojimų bazėje (arba realizuojamos kaip atskiros operacijos), apribojimai aprašomi naudojant dalykinės srities esybių atributus ir ryšius. Jei tikrintojo klasė praneša, kad operacijos prieš sąlyga negalioja, užklausos siuntėjui grąžinamas atmetimo (exception) pranešimas. Jei sąlyga galioja, koordinatorius persiunčia užklausos pranešimą konkrečiai paslaugai.

Kai koordinatorius gauna atsakymą iš paslaugos apie operacijos įvykdymo rezultatus, jis vėl kviečia tikrinimo klasę, kuri iš apribojimų bazės (ar iškvietęs konkrečią tikrinimo operaciją) turi pateikti operacijos po sąlygoje aprašytus siunčiamus pranešimus. Tai dažniausiai pranešimo siuntimas vienam paslaugos interfeisui, bet gali būti ir pranešimų sekos siuntimo aprašas, ar reikalavimas siųsti keletą lygiagrečių pranešimų skirtingiems gavėjams ir pan. Visi šie pranešimų siuntimo būdai aprašomi panaudojant OCL. Jei reikia siųsti atsako ar priėmimo patvirtinimo (acceptance) pranešimą, koordinatorius persiunčia tokį pranešimą aktoriaus paslaugai. Jei reikia siųsti užklausos pranešimą, koordinatorius vėl elgiasi kaip gavęs užklausą operacijai atlikti.

3.2. Koordinatoriaus veikimo modelis

Koordinatoriaus reakcijas į gaunamus pranešimus galima pavaizduoti tokiu modeliu:



9 pav. Koordinatoriaus sekų diagrama

Pavyzdys. Tegul koordinatoriaus šablonas bus taikomas leidybos agentūrai, kur pagrindiniai sistemos veikėjai yra šie:

- Autorius – pateikia savo darbus leidybos agentūrai
- Koordinatorius – priima darbus, juos perduoda patikrinimui, recenzijai.
- Tikrintojas – patikrina ar leidinys atitinka pradinius reikalavimus, peržiūri finansavimą, įvertina recenzijas.
- Recenzentas – atlieka recenzavimą.

Paimkime pavyzdžiu panaudojimo atvejį „Pateikti leidini“, bei, kad būtų aiškiau, supaprastinkime iki vieno žingsnio:

Panaudojimo atvejis „Pateikti leidinį“:

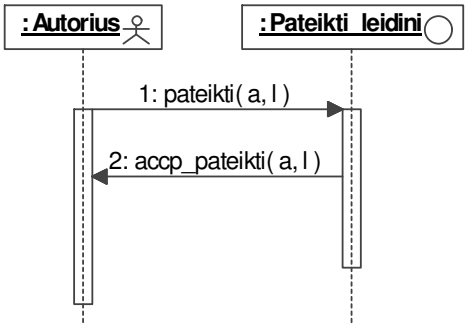
1. Autorius nurodo sistemai pateikti Leidinį svarstymui

   Įvykio Prieš sąlyga: Pateikiamas Leidinys yra būsenoje „Įkeltas“.

   Įvykio Sužadinimo sąlyga: Autorius nori pateikti Leidinį svarstymui

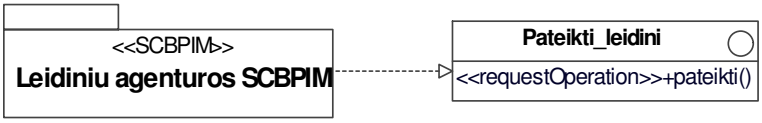
   Įvykio Po sąlyga: Leidinys yra būsenoje „pateiktas“. Autoriui nusiųstas Leidinio pateikimo patvirtinimas.

Tokį panaudojimo atvejį reikalavimų aprašymo etape atitiktų tokia sekos diagrama:



10 pav. Panaudojimo atvejo „Pateikti leidinį“ fragmentas

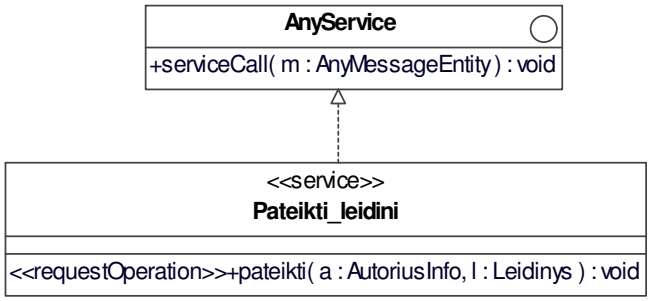
Leidinių agentūros sistema turėtų interfeisą:



11 pav. Leidinių agentūros interfeiso struktūra

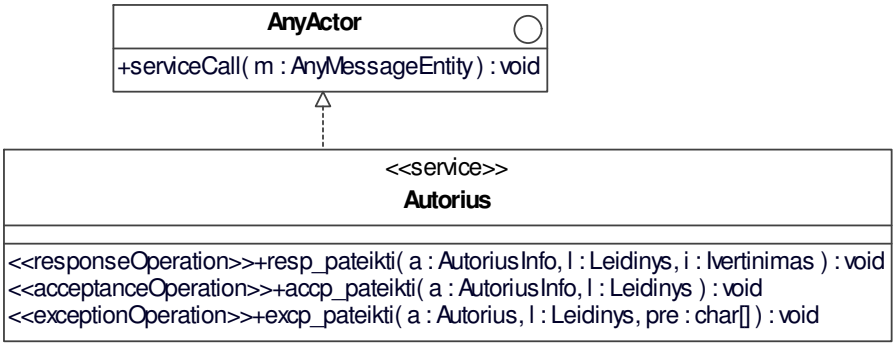
Pritaikius koordinatoriaus šabloną, viskas turėtų atrodyti taip:

Pirmiausia interfeisą „Pateikti leidinį“ turėtų realizuoti servisas, kuris taip pat turėtų ir koordinatoriaus šablono interfeisą AnyService:



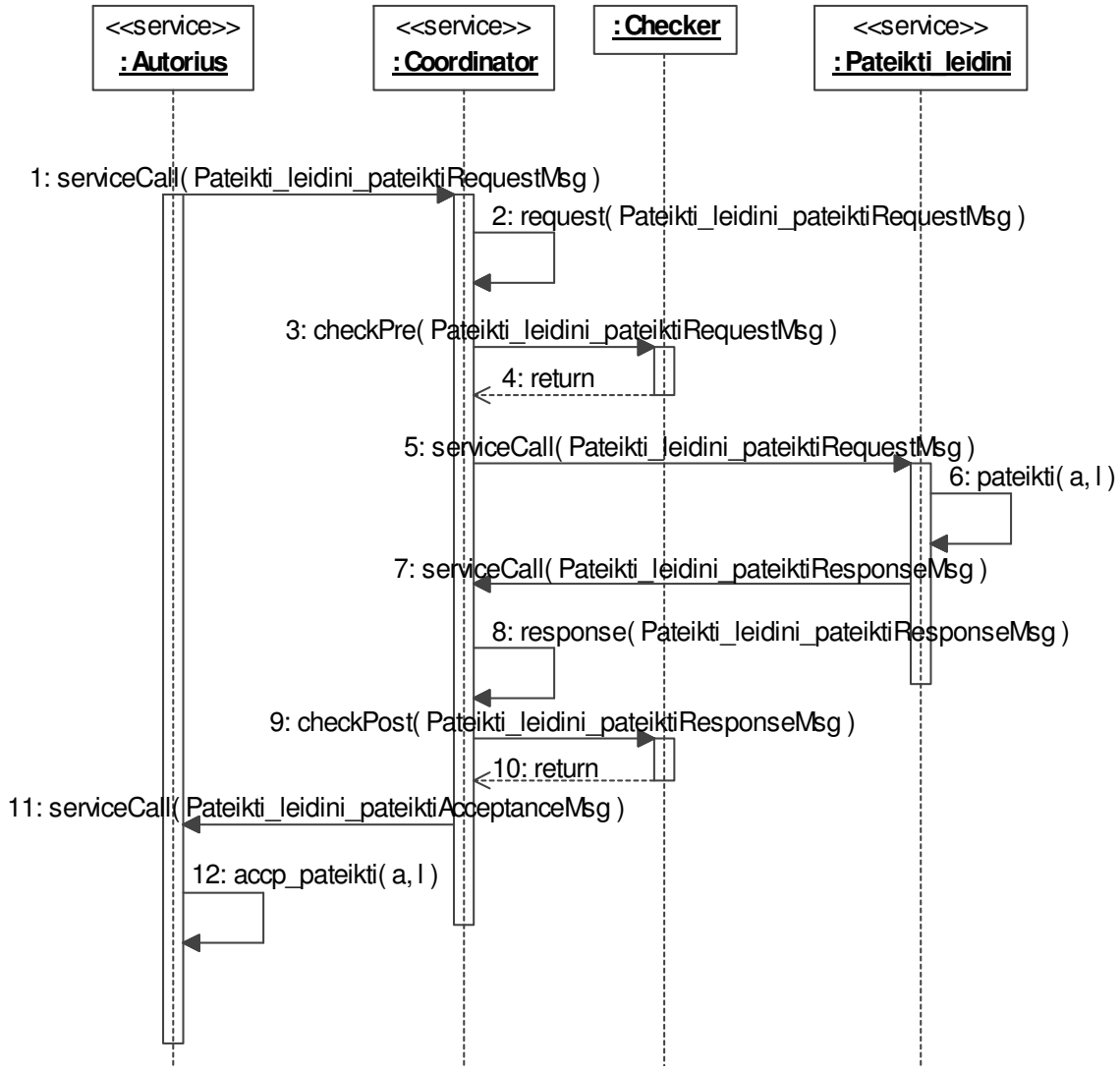
12 pav. Interfeisas AnyService

Autorius irgi tampa servisu, turinčiu šablono interfeisą AnyActor:



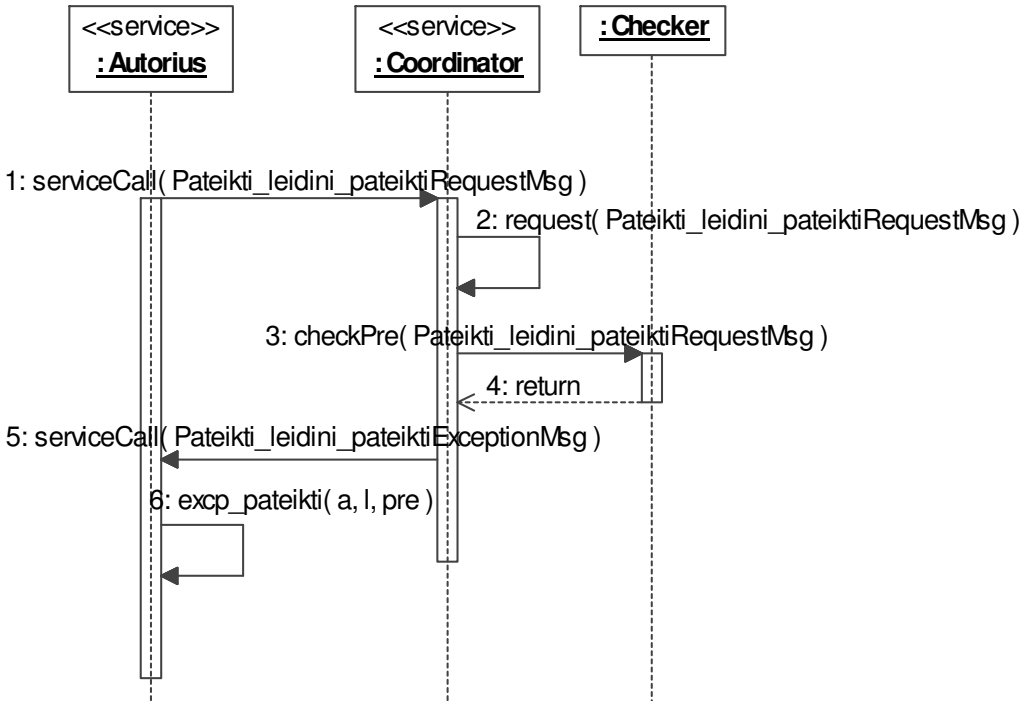
13 pav. Interfeisas AnyService

O buvusi sekos diagrama su interfeisais, pritaikius koordinatoriaus šabloną turėtų atrodyti taip:



14 pav. Sekos diagrama su interfeisais

Šioje sekos diagramoje matosi, kaip elgiasi koordinatorius, jeigu checker patikrina *prieš* sąlygą ir gauna rezultatą, kad ji negalioja:



15 pav. Koordinatoriaus elgesys, kai checker patikrina prieš sąlygą ir gauna rezultatą, kad ji negalioja

3.3. Kūrimas naudojant koordinatorių

3.3.1 Panaudojimo atveju specifikavimas, aprasant pries ir po salygas kiekvienam ivykiui

Kuriant sistemą pagal koordinatoriaus modelį, panaudos atveju specfikavimas truputį skiriasi nuo standartinės formos. Kiekvienam panaudos atveju bei įvykiui reikia nurodyti pre ir post sąlygas, kurias išpildžius, sistema veikai toliau, kitu atveju vartotojui gražinamas klaidos pranešimas. Visgi pre ar post salygą gali būti tuščia, jei tam tikram panaudos atvejui ar įvykiui jos nenaudojamos.

Panaudos atvejo pavyzdys:

Panaudojimo atvejis „Prisijungti“

Vartotojas / Aktorius: Autorius

Asmuo pateikia savo duomenis (el.pašto adresą ir slaptažodį) norėdamas prisijungti prie sistemos.

Prieš sąlyga: Asmuo su pateiktais duomenimis yra registruotas sistemoje ir jis yra atsijungęs

Sužadinimo sąlyga: Asmuo nori prisijungti

Po sąlyga: Asmuo yra prisijungęs; Asmeniui pateiktas prisijungimo

3.3.2 Koordinatoriaus projektavimas

Koordinatoriaus aprašymai yra skyriuose 3.1 bei 3.2.

3.3.3 Web servisų projektavimas

Tikrinimo sąlygu projektavimas

Koordinatorius norėdamas patikrint pre ar post sąlygas, kreipiasi i Tikrintojo klasę „*Tikrintojas*“, kuri turi 2 funkcijas:

1. tikrinti_pre(string paslauga, object[] parametrai)
2. tikrinti_post(string paslauga, object[]parametrai)

„*Tikrintojas*“ klasėje funkcijos, atitinkamai pagal paslaugą, patikrina pre bei post sąlygas ir jei jos tenkinamos – patvirtinimą kad viskas gerai (reikšmių pavyzdys: *true*, *ok*), jei ne - gražina klaidą.

Servisų projektavimas

Kiekvienas servisas bendrauja su sistema per Koordinatorių. Norint vykdyti paslauga, kreipiamasi į Koordinatorių ir jam perduodama tokia informacija:

1. paslaugos pavadinimas
2. parametrai

Gaunamą atsakymą sudaro: klaidos pranešimas arba įvykdytos paslaugos atsakymas.

Pranešimų projektavimas

Kadangi įvairūs servisai su savo funkcijom ir skirtingais kintamaisiais bei reikalavimais kreipiasi į koordinatorių, kuris turi priimti duomenis ir nusiūsti atsakymą, pas praktiškai nežinodamas kokią paslaugą aptarnaują, pranešimų formatas turėtų būti gana abstraktus. Pranešimo koordinatoriui struktūros pavyzdys:

1. paslaugos pavadinimas

2. masyvas, kurį sudaro visi perduodami kintamieji.

Koordinatoriaus atsakymo struktūros pavyzdys:

DataSet objektas, kurs gali susidaryti iš daugelio objektų:ir klaidų pranešimų, ir duomenų lentelių.

3.3.4 Naujos paslaugos sukūrimas

Norint pridėti naują paslaugą, reikia:

1. specifikuoti panaudos atvejus, aprašant pre ir post salygas,
2. suprojektuoti WEB sevisą,
    a. realizuoti tikrinimo sąlygas
    b. realizuoti patį web servisą

4. Koordinatoriaus šablono realizacija leidybos agentūros paslaugų sistemai

Šiame skyriuje aprašoma konkreti sistema, sukurta leidybos agentūros paslaugoms teikti, kuri įgyvendina koordinatoriaus šabloną praktikoje. Bus aprašomos sistemos atliekamos funkcijos, jos architektūra, koordinatoriaus sąveikos su išoriniais servisais, pati paslaugų sistema ir kt. Be to, bus detaliai aprašyti paslaugų sistemos komponentai bei klasės.

4.1. Leidybos agentūros funkcijos

Šiuolaikinė leidybos agentūra ar leidykla – sudėtinga organizacija, kurioje kartu turi dirbti daug žmonių. Autoriai pateikia medžiagą spausdinimui, redaktoriai stengiasi atsirinkti tik tai, kas geriausia, tačiau ne visada spėja perskaityti visus autorių pateiktus kūrinius, todėl į pagalbą kviečiasi recenzentus. Žinoma, nereikia pamiršti ir finansinės pusės, kai leidykla gali sau leisti už savo lėšas spausdinti tik geriausius darbus, tačiau autoriai ne visada nori padengti leidybos išlaidas. Nenuostabu, kad viso leidybos proceso efektyvus koordinavimas dažnai tampa sudėtingu uždaviniu net ir geriausiems leidėjams. Sukurtos paslaugų sistemos tikslas ir yra palengvinti leidybos proceso koordinavimą bei efektyvų valdymą. Taigi, tam, kad sistema būtų efektyvi, būtina aprašyti pagrindines leidybos agentūros funkcijas.

Autoriai, naudodamiesi Interneto tinklu, gali prisiregistruoti prie informacinės sistemos bei pateikti leidybos agentūrai savo darbus ir kūrinius bei prašyti finansavimo juos išleidžiant. Agentūroje dirbantys žmonės bei recenzentai gali peržiūrėti ir įvertinti autorių darbus bei nuspręsti tolesnį jų likimą – finansuoti jų išleidimą už leidybos agentūros lėšas ar pasiūlyti autoriui sumokėti už leidybą pačiam.

Sistemos vartotojai:

Autorius

> Vartotojo sprendžiami uždaviniai: prisijungti prie sistemos ir atsijungti nuo jos; pateikti savo darbą ar kūrinį leidybai; prašyti finansavimo savo darbo išleidimui.
>
> Patirtis dalykinėje srityje: įprastas darbuotojas .

Patirtis informacinėse technologijose: naujokas.

Papildomos vartotojo charakteristikos:

- atsakingas požiūris į darbą;
- bendri lietuvių kalbos įgūdžiai;

Vartotojų prioritetai. – pirmaeiliai.

Komitetas

Vartotojo sprendžiami uždaviniai: skirti paramą autoriui; gauti ataskaitas apie autorių kūrinius bei recenzentų darbą; spręsti finansavimo klausimus.

Patirtis dalykinėje srityje: srities specialistas.

Patirtis informacinėse technologijose: patyręs.

Papildomos vartotojo charakteristikos:

- atsakingas požiūris į darbą;
- bendri lietuvių kalbos įgūdžiai;

Vartotojų prioritetai – svarbiausi.

Recenzentas

Vartotojo sprendžiami uždaviniai: prisijungti ir atsijungti nuo sistemos; gauti jam persiunčiamą autoriaus darbą; nusiųsti savo recenziją.

Patirtis dalykinėje srityje: srities specialistas.

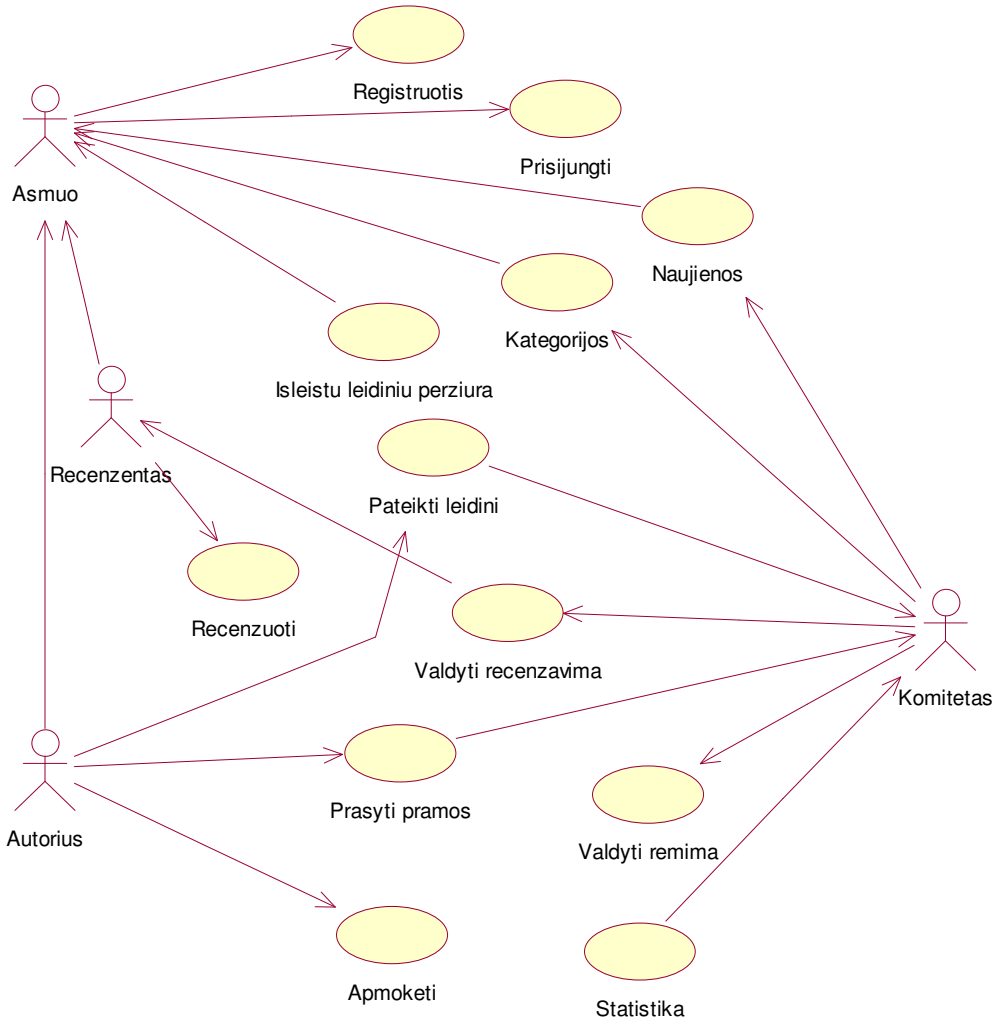
Patirtis informacinėse technologijose: naujokas.

Papildomos vartotojo charakteristikos:

- atsakingas požiūris į darbą;
- bendri lietuvių kalbos įgūdžiai;

Vartotojų prioritetai – pirmaeiliai.

4.2. Reikalavimai leidybos paslaugų sistemai



16 pav. Panaudos atvejų diagrama

Pastaba: kadangi sistema yra kuriama pagal konsultanto nustatytą metodą, reikalavimų specifikacija šiek tiek skiriasi nuo Volere šablono – prieš sąlygos, sužadinimo sąlygos ir po sąlygos yra nustatomos ne tik panaudojimo atvejui, bet ir kiekvienam panaudojimo atvejo žingsniui. Pirmo panaudojimo atvejo žingsnio prieš sąlygos, sužadinimo sąlygos ir po sąlygos sutampa su panaudojimo atvejo sąlygomis. Jei panaudojimo atvejis apima tik vieną įvykį, įvykio ir panaudojimo atvejo specifikacija sutampa.

1 Panaudojimo atvejis „Registruotis“

Vartotojas / Aktorius: Autorius, Recenzentas

Prieš sąlyga: Nėra

Aprašai:

1. Asmuo pateikia savo duomenis (vardas, pavardė, laipsnis, el. paštas, slaptažodis) pageidaudamas užsiregistruoti sistemoje.

   Prieš sąlyga: Sistemoje nėra registruoto AsmuoInfo, kurio el. pašto adresas sutaptų su pateikiamu.

   Sužadinimo sąlyga: Asmuo nori registruotis

   Po sąlyga: Sistemoje yra užregistruotas naujas AsmuoInfo su pateiktais duomenimis. Autoriui siunčiamas atsakymas apie registraciją sistemoje.

2. Asmuo kreipiasi į sistemą norėdamas pakoreguoti savo duomenis

   Prieš sąlyga: Sistemoje yra registruotas AsmuoInfo.

   Sužadinimo sąlyga: Asmuo nori redaguoti savo duomenis

   Po sąlyga: Sistemoje yra atnaujinta AsmuoInfo su pateiktais duomenimis. Asmeniui nusiustas atsakymas apie AsmuoInfo pakeitimus sistemoje.

3. Asmuo kreipiasi į sistemą norėdamas išsiregistruoti.

   Prieš sąlyga: Sistemoje yra registruotas AsmuoInfo.

   Sužadinimo sąlyga: Asmuo nori ištrinti savo duomenis

   Po sąlyga: Sistemoje nėra informacijos apie AsmuoInfo. Asmeniui nusiustas atsakymas apie AsmuoInfo ištrynimą sistemoje

Po sąlyga: AsmuoInfo duomenys yra pakeisti (įvesti, redaguoti, ištrinti). Asmeniui

2 Panaudojimo atvejis „Prisijungti“

Vartotojas / Aktorius: Autorius

Prieš sąlyga: Asmuo su pateiktais duomenimis yra registruotas sistemoje ir nėra prisijungęs

Aprašai:

1. Asmuo pateikia savo duomenis (el.pašto adresą ir slaptažodį) norėdamas prisijungti prie sistemos.

   Prieš sąlyga: Asmuo su pateiktais duomenimis yra registruotas sistemoje ir jis yra atsijungęs

   Sužadinimo sąlyga: Asmuo nori prisijungti

   Po sąlyga: Asmuo yra prisijungęs; Asmeniui pateiktas prisijungimo patvirtinimas

2. Asmuo pateikia savo duomenis norėdamas atsijungti nuo sistemos.

   Prieš sąlyga: Asmuo su pateiktais duomenimis yra registruotas sistemoje ir jis yra prisijungęs

   Sužadinimo sąlyga: Asmuo nori atsijungti

   Po sąlyga: Asmuo yra atsijungęs; Asmeniui pateiktas atsijungimo patvirtinimas

3 Panaudojimo atvejis „Naujienos“

Vartotojas / Aktorius: Recenzentas, Autorius, Komitetas

Prieš sąlyga: Komitetas yra prisijungęs.

Aprašai:

1. Asmuo gali peržiūrėti naujienas.
    a. **Prieš sąlyga:**
    b. **Sužadinimo sąlyga:** Asmuo nori peržiūrėti naujienas.
    c. **Po sąlyga:**
2. Komitetas gali kurti naujienas.
    a. **Prieš sąlyga:**
    b. **Sužadinimo sąlyga:** Komitetas kurti naujienas.
    c. **Po sąlyga:** Naujiena sukurta .
3. Komitetas gali trinti naujienas.
    a. **Prieš sąlyga:** Trinama naujiena egzistuoja.
    b. **Sužadinimo sąlyga:** Komitetas trinti naujienas.
    c. **Po sąlyga:** Naujiena istrinta iš DB .

Po sąlyga: Asmeniui pateikiamos atnaujintos naujienos.

4 Panaudojimo atvejis „Kategorijos“

Vartotojas / Aktorius: Autorius /Komitetas

Prieš sąlyga: Komitetas yra prisijungęs.

Aprašai:

1. Asmuo gali peržiūrėti kategorijas.
    a. **Prieš sąlyga:**
    b. **Sužadinimo sąlyga:** Asmuo nori peržiūrėti kategorijas.
    c. **Po sąlyga:**
2. Komitetas gali kurti kategorijas.
    a. **Prieš sąlyga:** Nera norimos kategorijos.
    b. **Sužadinimo sąlyga:** Komitetas nori kurti kategorijas.
    c. **Po sąlyga:** Kategorija sukurta.
3. Komitetas gali trinti kategorijas.
    a. **Prieš sąlyga:** Trinama kategorija egzistuoja.
    b. **Sužadinimo sąlyga:** Komitetas trinti kategorijas.
    c. **Po sąlyga:** Kategorija istrinta iš DB .
4. Asmuo gali siulyti naujas kategorijas.
    a. **Prieš sąlyga:** Kategorija neegzistuoja.
    b. **Sužadinimo sąlyga:** Asmuo, pateikdamas leidinį ir nerasdamas jam tinkamos kategorijos, nori siulyti naują.
    c. **Po sąlyga:** Kategorija irašoma į DB ir laukia komiteto patvirtinimo.

Po salyga: Asmeniui pateikiamos atnaujintos kategorijos.

5 Panaudojimo atvejis „Isleistu leidiniu perziura“

Vartotojas / Aktorius: Asmuo

Prieš sąlyga: Egzistuoja išleisti leidiniai.

Aprašas: Jei leidinys išleistas, ji gali pamatyti neregistruotas asmuo.

Sužadinimo sąlyga: Asmuo nori peržiūrėti išleistus leidinius.

Po sąlyga: Išleisti leidiniai pateikiami asmeniui.

6 Panaudojimo atvejis „Pateikti leidinį“

Vartotojas / Aktorius: Autorius

Prieš sąlyga: Autorius yra prisijungęs ir Leidinys tokiu pavadinimu sistemoje neegzistuoja

Sužadinimo sąlyga: Autorius nori pateikti leidinį

Aprašas:

1. Autorius įkelia informaciją apie naują Leidinį - jos Autorių, pavadinimą, anotaciją.

   Įvykio prieš sąlyga: Autorius yra prisijungęs ir Leidinys tokiu pavadinimu sistemoje neegzistuoja

   Įvykio Sužadinimo sąlyga: Autorius nori pateikti leidinį

   Įvykio po sąlyga: Sistemoje yra sukurtas naujas Leidinys nurodytu pavadinimu ir šis leidinys yra būsenoje „Įkeltas“. Autoriui siunčiamas atsako pranešimas apie naujo Leidinio sukūrimą.

2. Sistema pateikia Leidinį svarstymui – siunčia Komitetui pranešimą pradėti Leidinio svarstymą, Autoriui siunčia Leidinio pateikimo patvirtinimą.

   Įvykio Prieš sąlyga: Pateikiamas Leidinys yra būsenoje „Įkeltas“.

   Įvykio Sužadinimo sąlyga: nėra (ankstesnis žingsnis)

   Įvykio Po sąlyga: Leidinys yra būsenoje „pateiktas“. Komitetui nusiųstas pranešimas pradėti Leidinio svarstymą. Autoriui nusiųstas Leidinio pateikimo patvirtinimas.

3. Sistema atsiunčia atsakymą apie leidinio svarstymo rezultatus

   Įvykio Prieš sąlyga: Leidinys yra „Priimtas“ arba „Atmestas“. Leidinio Įvertinimas yra sukurtas.

   Įvykio Sužadinimo sąlyga: atsakymo apie leidinio svarstymo rezultatus gavimas sistemoje.

   Įvykio Po sąlyga: Autoriui pateiktas atsakymas į Leidinio pateikimą:

7 Panaudojimo atvejis "Recenzuoti"

Vartotojas / Aktorius: Recenzentas

Prieš sąlyga: Recenzentas yra prisijungęs.

Aprašai:

1. Recenzentas pildo Leidinio Recenziją.

   Prieš sąlyga: Recenzentas yra Prisijungęs; Leidinys yra Svarstomas; Recenzija priskirta Recenzentui.

   Sužadinimo sąlyga: Recenzento kreipimasis į sistemą dėl recenzavimo

   Po sąlyga: Recenzija yra būsenoje „Pradėta".

2. Recenzentas pateikia baigtą pildyti Leidinio Recenziją .

   Prieš sąlyga: Leidinys yra svarstomas. Recenzentas yra prisijungęs. Recenzija gali būti „Pradėta" arba neegzistuoti.

   Sužadinimo sąlyga: Recenzento kreipimasis į sistemą dėl recenzavimo

   Po sąlyga: Recenzija yra „Baigta". Recenzentui pateiktas baigtos Recenzijos gavimo patvirtinimas.

Po sąlyga: Recenzija yra baigta. Recenzentui pateiktas baigtos Recenzijos gavimo patvirtinimas.

8 Panaudojimo atvejis "Valdyti recenzavimą"

Vartotojas / Aktorius: Komitetas

Sužadinimo sąlyga: Komitetas nori pradėti Leidinio svarstymą

Prieš sąlyga: Leidinys yra pateiktas svarstymui (būsenoje „Pateiktas")

Aprašas:

1. Komitetas paskiria Leidiniui Recenzentus

   Įvykio Prieš sąlyga: Leidinys yra pateiktas svarstymui (būsenoje „Pateiktas"); sistemoje yra registruotų Recenzentų

   Įvykio Sužadinimo sąlyga: Komitetas nori pradėti Leidinio svarstymą

   Įvykio Po sąlyga: Leidiniui priskirti Recenzentai, kurių skaičius nemažesnis už minimalų recenzijų skaičių; Leidinys yra būsenoje „Svarstomas"; Recenzentams nusiųsti pranešimai apie paskirtą Leidinį

2. Komitetas peržiūri Leidinio Recenzijas ir suformuoja Leidinio Įvertinimą. Recenzijose pateikti įvertinimai susumuojami, skaičiuojamas reitingas.

   Įvykio Prieš sąlyga: Leidinys yra „Svarstomas"; Visos Leidinio Recenzijos yra „Baigtos".

   Įvykio Sužadinimo sąlyga: Komitetas nori peržiūrėti Leidinio Recenzijas.

   Įvykio Po sąlyga: Leidinio Įvertinimas yra „Sukurtas"

3. Komitetas patvirtina Leidinio Įvertinimą – priima galutinį sprendimą, ar Leidinys „Priimtas".

   Įvykio Prieš sąlyga: Leidinys yra „Svarstomas", Įvertinimas „Sukurtas".

   Įvykio Sužadinimo sąlyga: Komitetas nori patvirtinti Leidinio Įvertinimą

   Įvykio Po sąlyga: Leidinio Įvertinimas yra „Patvirtintas", Leidinys yra „Priimtas" arba „Atmestas". Autoriui nusiųstas Leidinio Įvertinimas.

Po sąlyga: Leidinys turi naują Įvertinimą, kurio būsena „Patvirtintas". Leidinys yra

9 Panaudojimo atvejis „Prašyti paramos“

Vartotojas / Aktorius: Autorius

Prieš sąlyga: Autorius yra prisijungęs. Leidinys yra „Priimtas“.

Sužadinimo sąlyga: Autorius nori prašyti Paramos

Aprašai:

1. Autorius pateikia prašymą dėl paramos Leidiniui skyrimo su aprašymu.

   Įvykio Prieš sąlyga: Autorius yra „Prisijungęs“. Leidinys yra „Priimtas“.

   Įvykio Sužadinimo sąlyga: Autorius nori prašyti Paramos

   Įvykio Po sąlyga: Leidiniui sukuriama nauja klasė Parama, kuri yra būsenoje „Pateiktas prašymas“. Komitetui siunčiamas pranešimas pradėti Paramos skyrimo svarstymą

2. Komitetas atsiunčia atsakymą apie Paramos skyrimo svarstymo rezultatus.

   Įvykio Prieš sąlyga: -

   Įvykio Sužadinimo sąlyga: -

   Įvykio Po sąlyga: Po sąlyga: gautas atsakymas dėl Paramos; Parama yra „Skirta“ arba „Neskirta“.

10 Panaudojimo atvejis „Valdyti rėmimą“

Vartotojas / Aktorius: Komitetas

Prieš sąlyga: Leidinio Paramos Prašymas yra Pateiktas; Autorius yra gavęs ne daugiau dviejų Paramų

Sužadinimo sąlyga: Komitetas nori apsvarstyti Paramos prašymą

Aprašai:

1. Komitetas priima sprendimą dėl paramos skyrimo.

Po sąlyga: Paramos Prašymas yra „Patvirtintas“; „Parama yra „Suteikta“ arba „Nesuteikta“; Autoriui nusiustas pranešimas dėl Paramos skyrimo arba atmetimo.

11 Panaudojimo atvejis “Apmokėti“

Vartotojas / Aktorius: Autorius

Prieš sąlyga: Autorius yra prisijungęs. Leidinys yra „Priimtas“ ir Parama jam nėra „Suteikta“.

Aprašas: Autorius pateikia sistemai informaciją apie apmokėtą Leidinio išleidimą.

Sužadinimo sąlyga: Autorius kreipiasi į sistemą dėl apmokėjimo.

Po sąlyga: Leidinys yra „Apmokėtas“; Autoriui nusiųstas patvirtinimas apie Leidinio apmokėjimo gavimą.

12 Panaudojimo atvejis "Statistika"

Vartotojas / Aktorius: Komitetas

Prieš sąlyga: Komitetas yra prisijungęs.

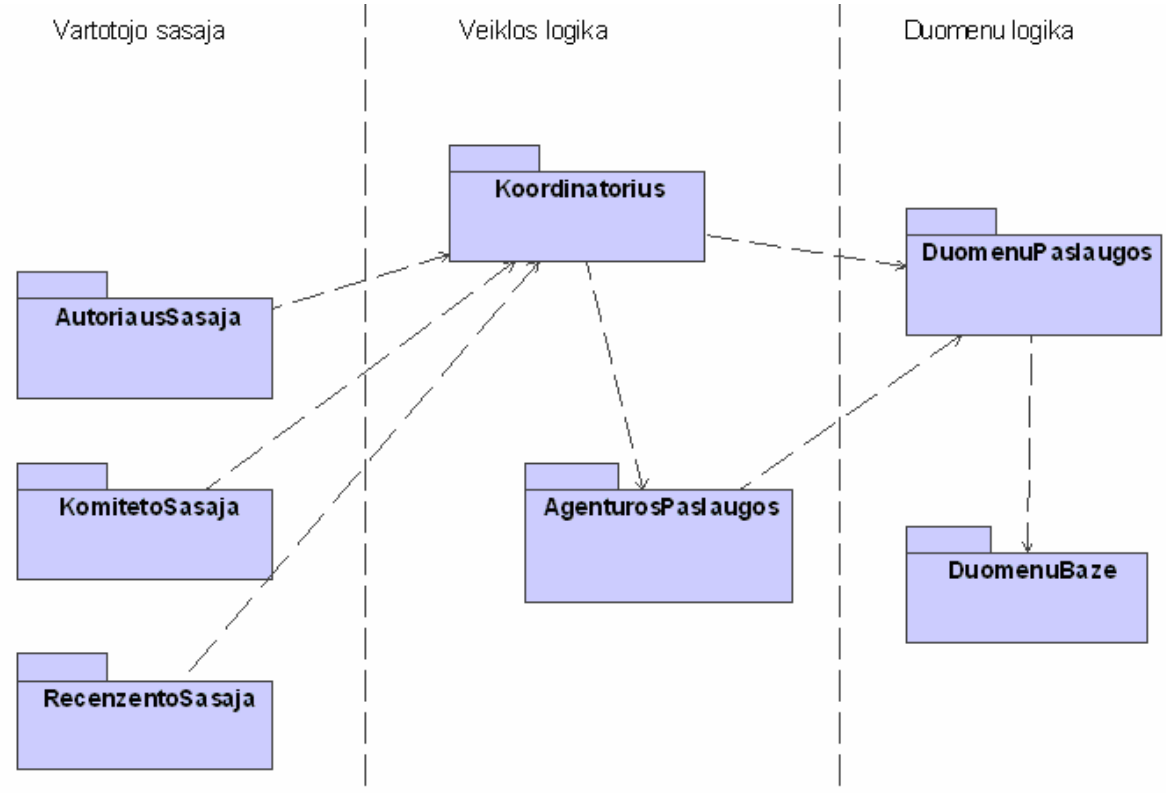
Aprašas: Komietui pateikiama statistika apie sistemą.

Sužadinimo sąlyga: Komitetas nori peržiūrėti statistiką.

Po sąlyga: Statistika pateikiama komitetui.

4.3. Leidybos paslaugų sistemos architektūra

Sistemos statinis vaizdas

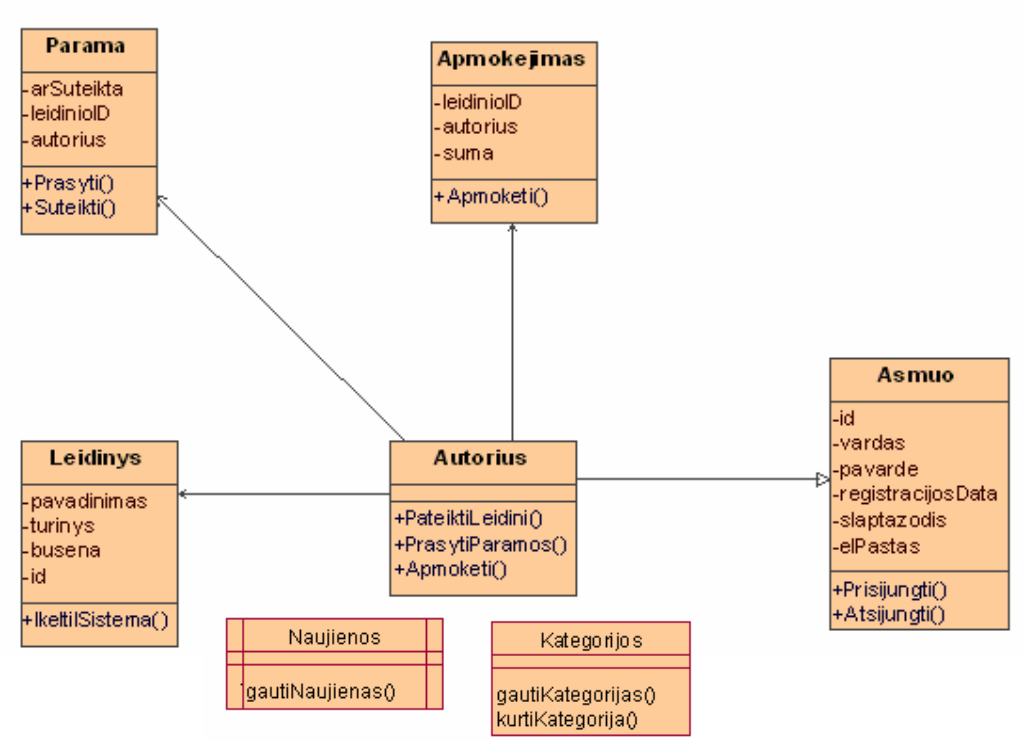


17 pav. Paketų diagrama

Paketų detalizavimas

Paketas „Autoriaus sąsaja"

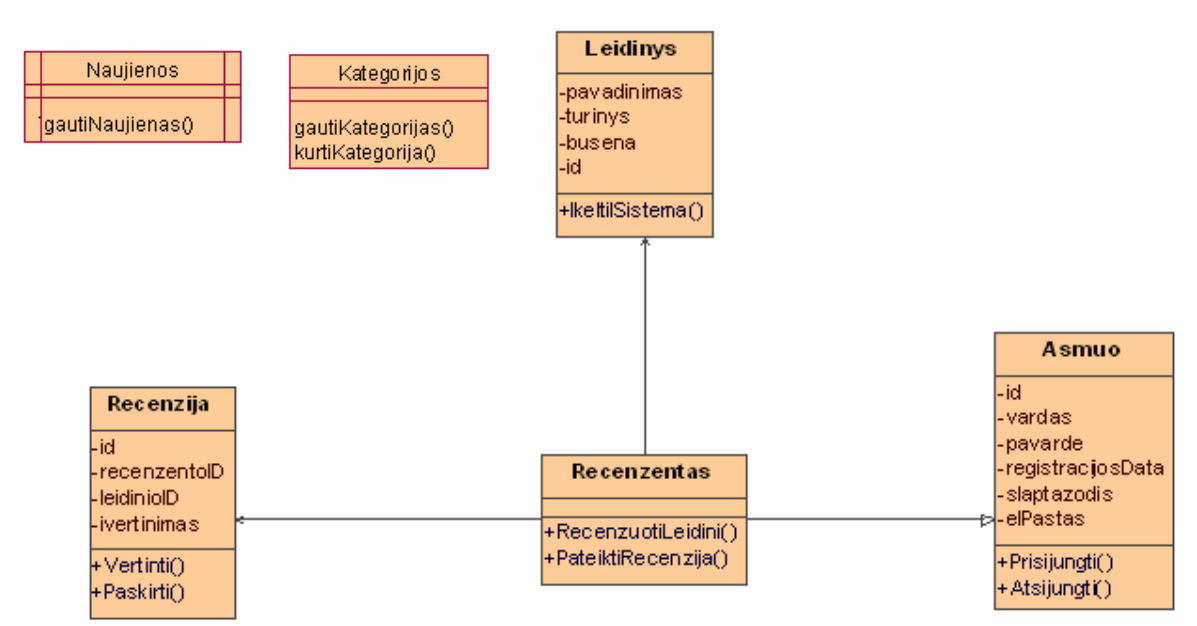
Šiame pakete pateiktos klasės realizuojančios autoriaus sąsają.



18 pav. Paketas "AutoriausSasaja"

Paketas „RecenzentoSasaja"

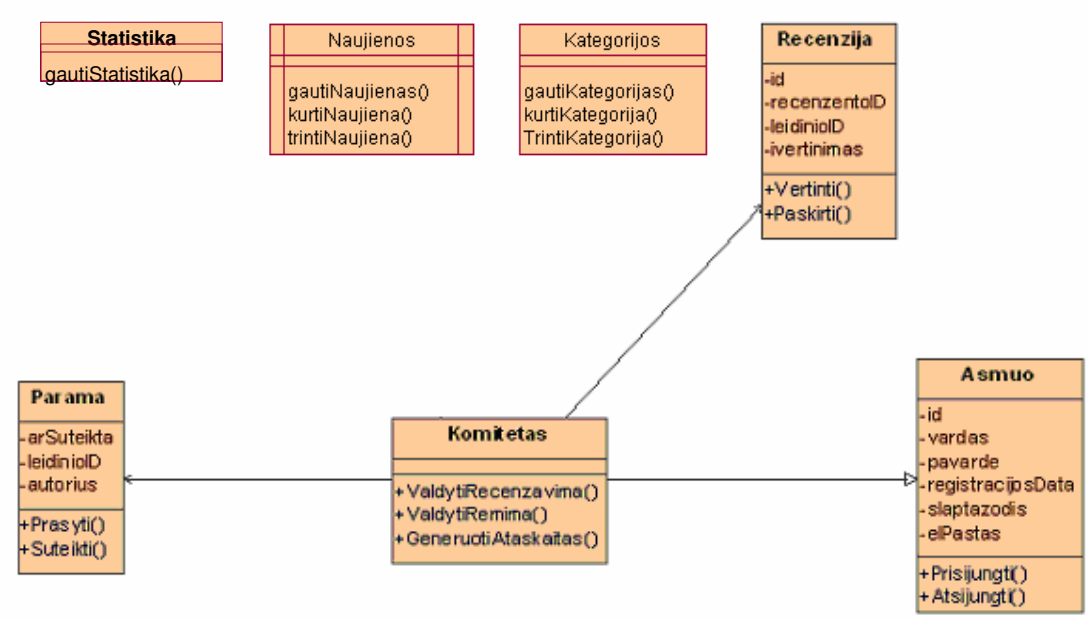
Šiame pakete pateiktos klasės realizuojančios recenzento sąsają.



19 pav. Paketas "RecenzentoSasaja"

Paketas „KomitetoSasaja"

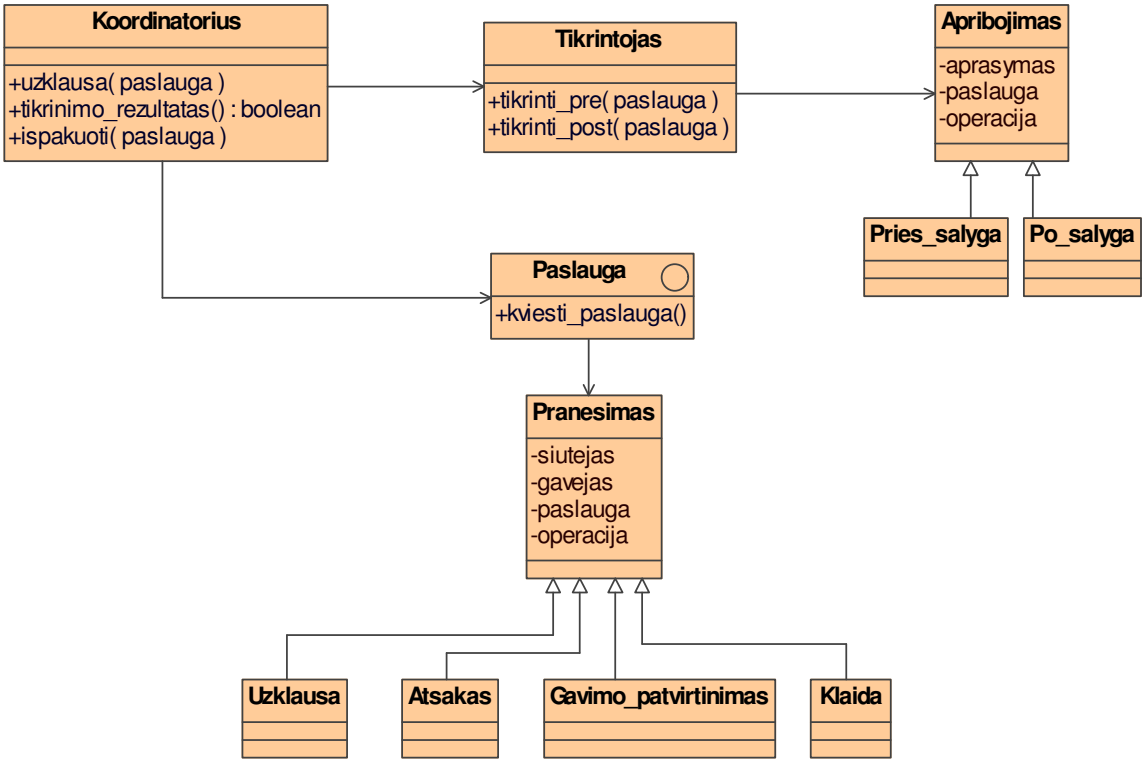
Šiame pakete pateiktos klasės realizuojančios komiteto sąsają.



20 pav. Paketas "KomitetoSasaja"

Paketas „Koordinatorius“

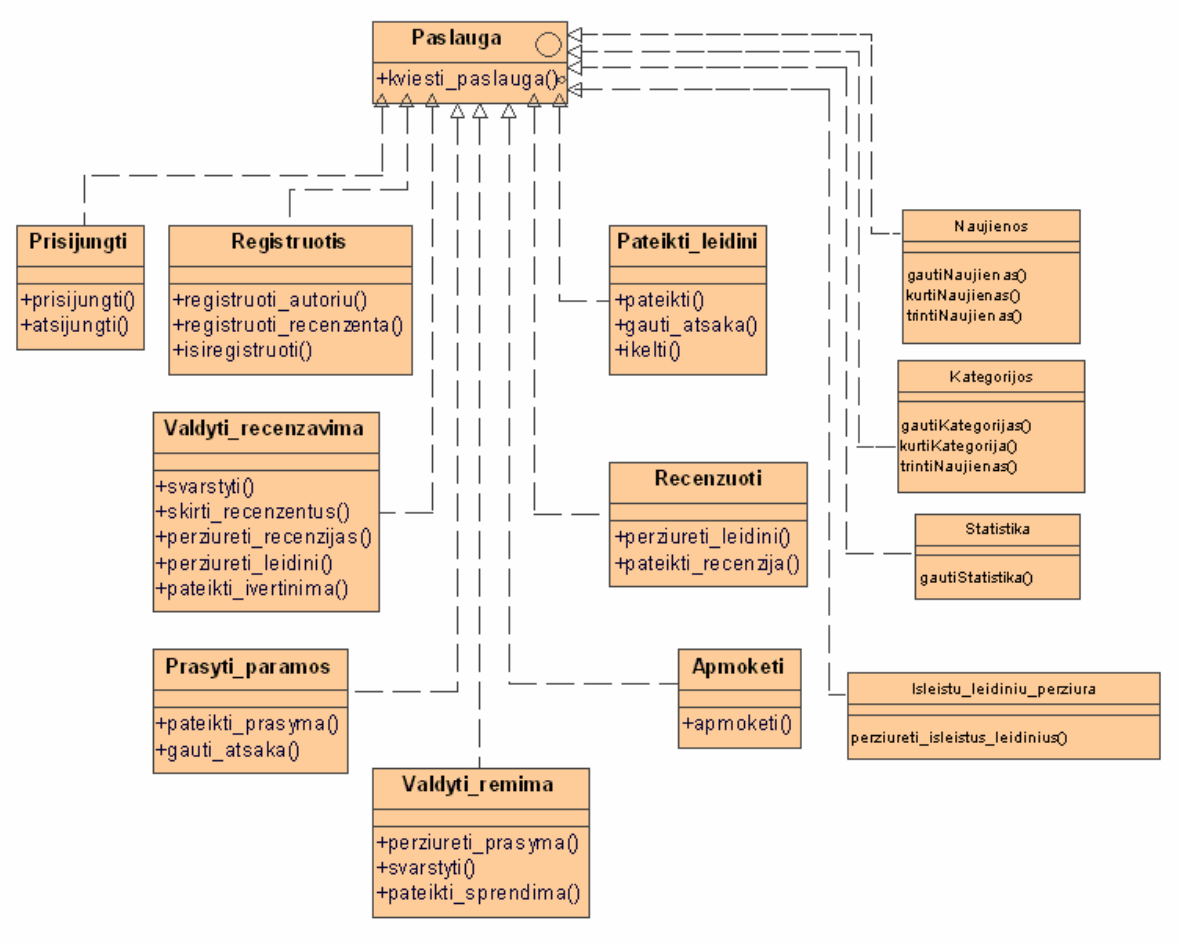
„Koordinatorius“ saugomos klasės, kurios priima iš vartotojo sąsajos ateinančias užklausas, nukreipia jas į tikrintoją, kuris tikrina, ar užklausą galima vykdyti. Jei tikrintojas patvirtina, kad užklausą galima vykdyti, koordinatorius kviečia atitinkamą paslaugą. „Paslauga“, „Apribojimas“ ir „Pranešimas“ yra šabloninės klasės, vietoje jų įstatomos konkrečios klasės, kurios aprašytos Leidinių agentūros pakete.



21 pav. Paketas “Koordinatorius”

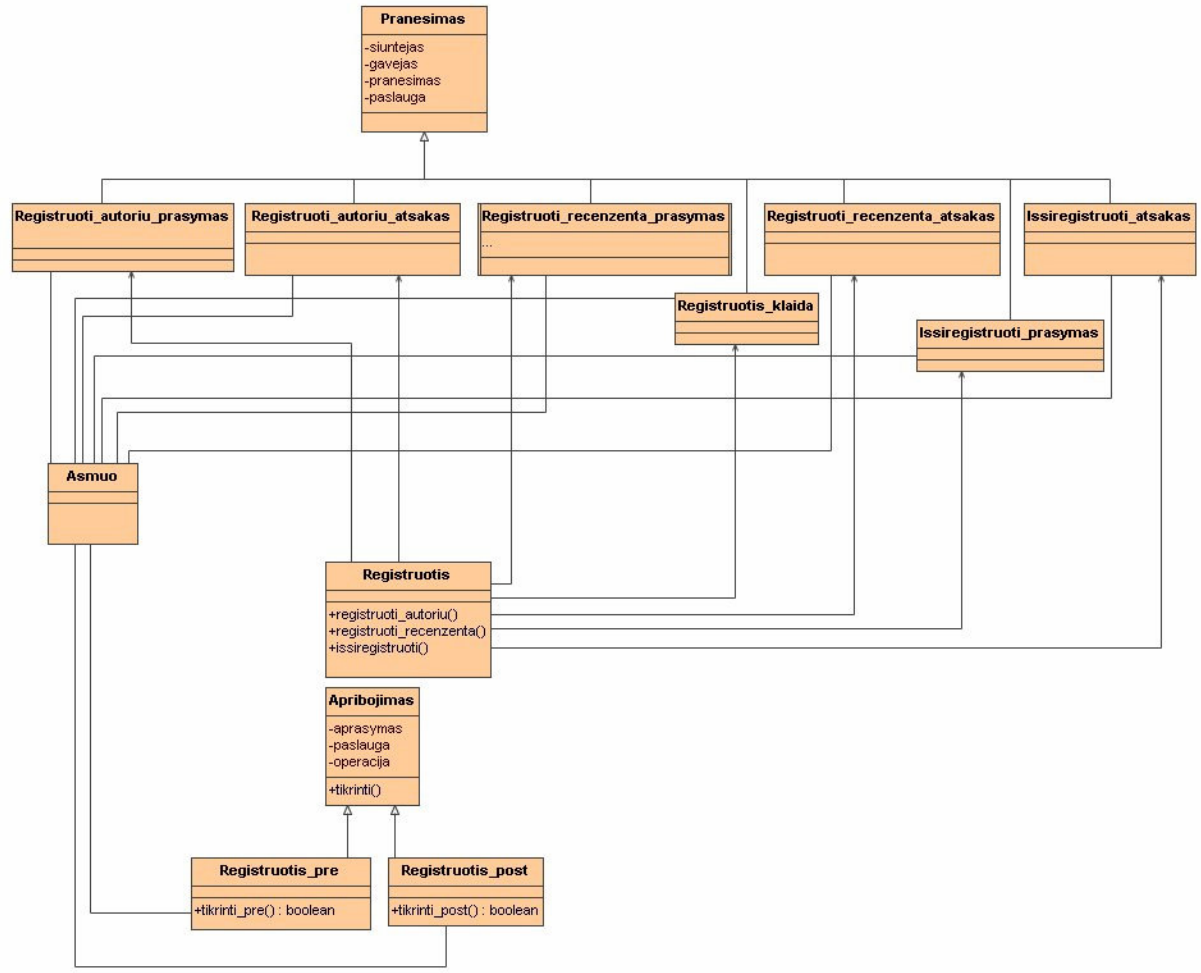
Paketas „AgenturosPaslaugos“

Šiame pakete saugomos klasės, kurios realizuoja nepriklausomas paslaugas. Visos paslaugos realizuoja bendrą interfeisą, kuris iškviečia paslaugą pagal jos vardą. Kiekviena paslauga aprašyta atskirame pakete, kadangi ji turi savo pranešimus ir tikrinimo operacijas.



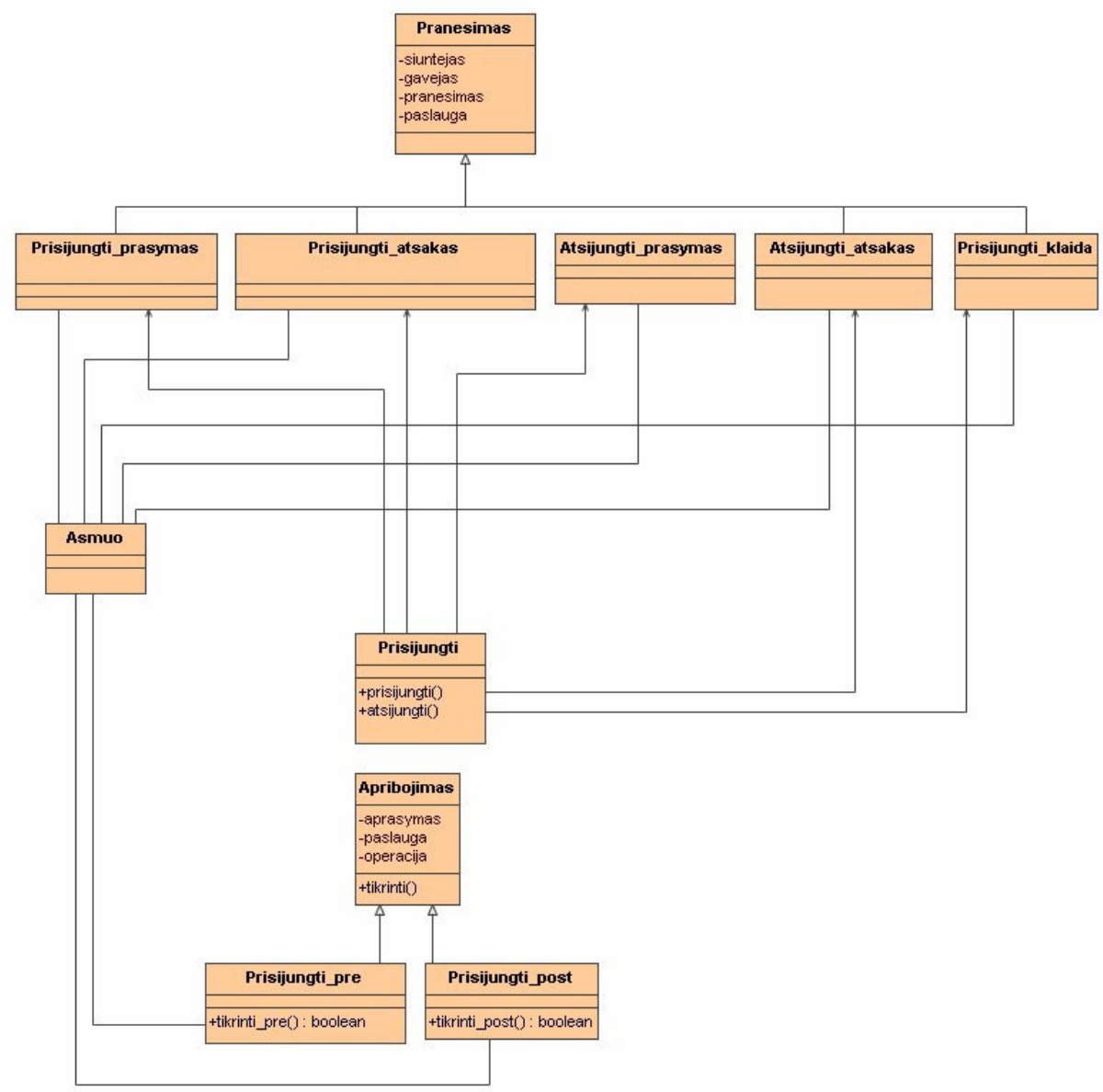
22 pav. Paketas “AgenturosPaslaugos”

Paslaugos „Registruotis“ paketas:



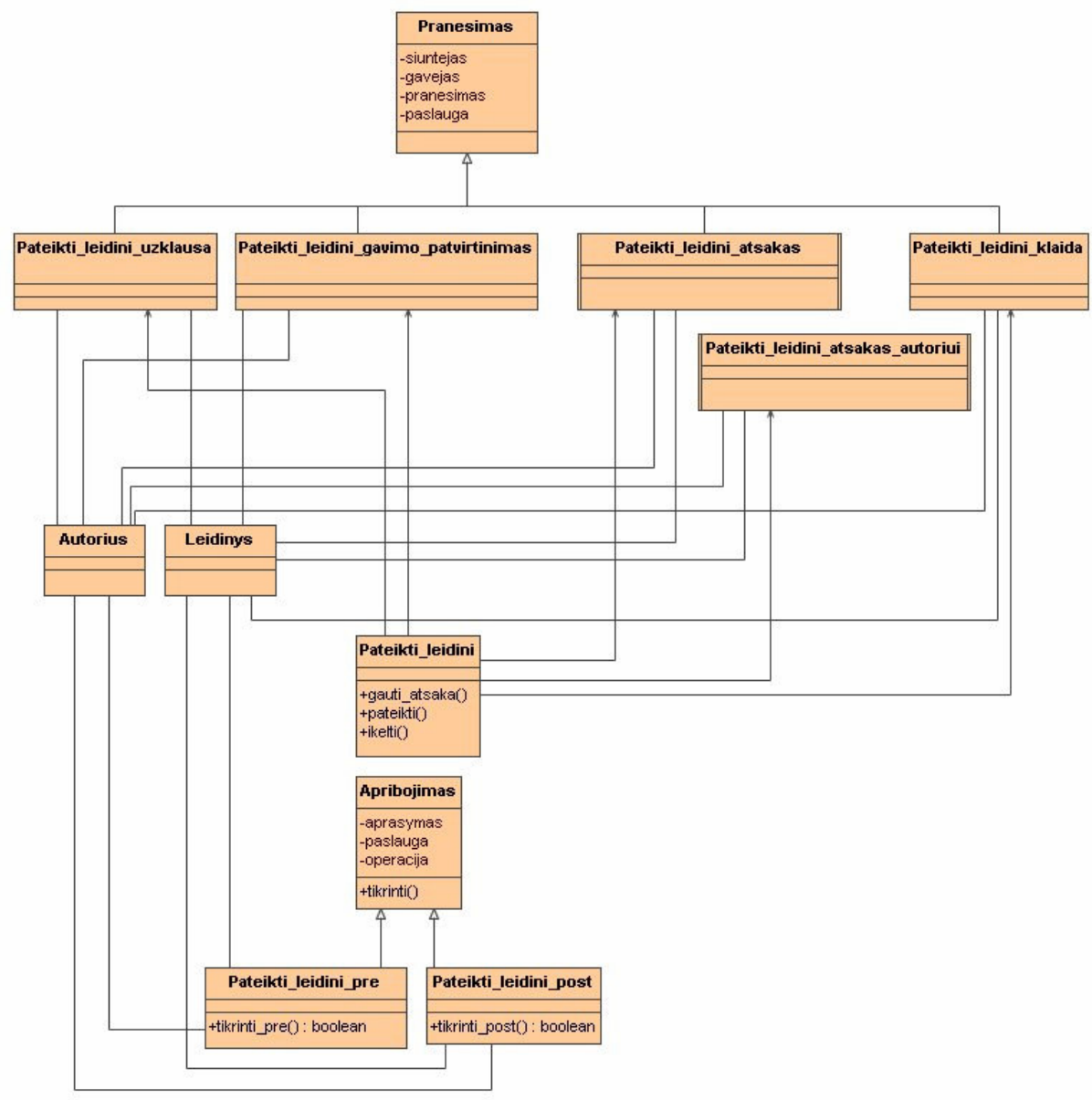
23 pav. Paketas “Registruotis”

Paslaugos „Prisijungti“ paketas:



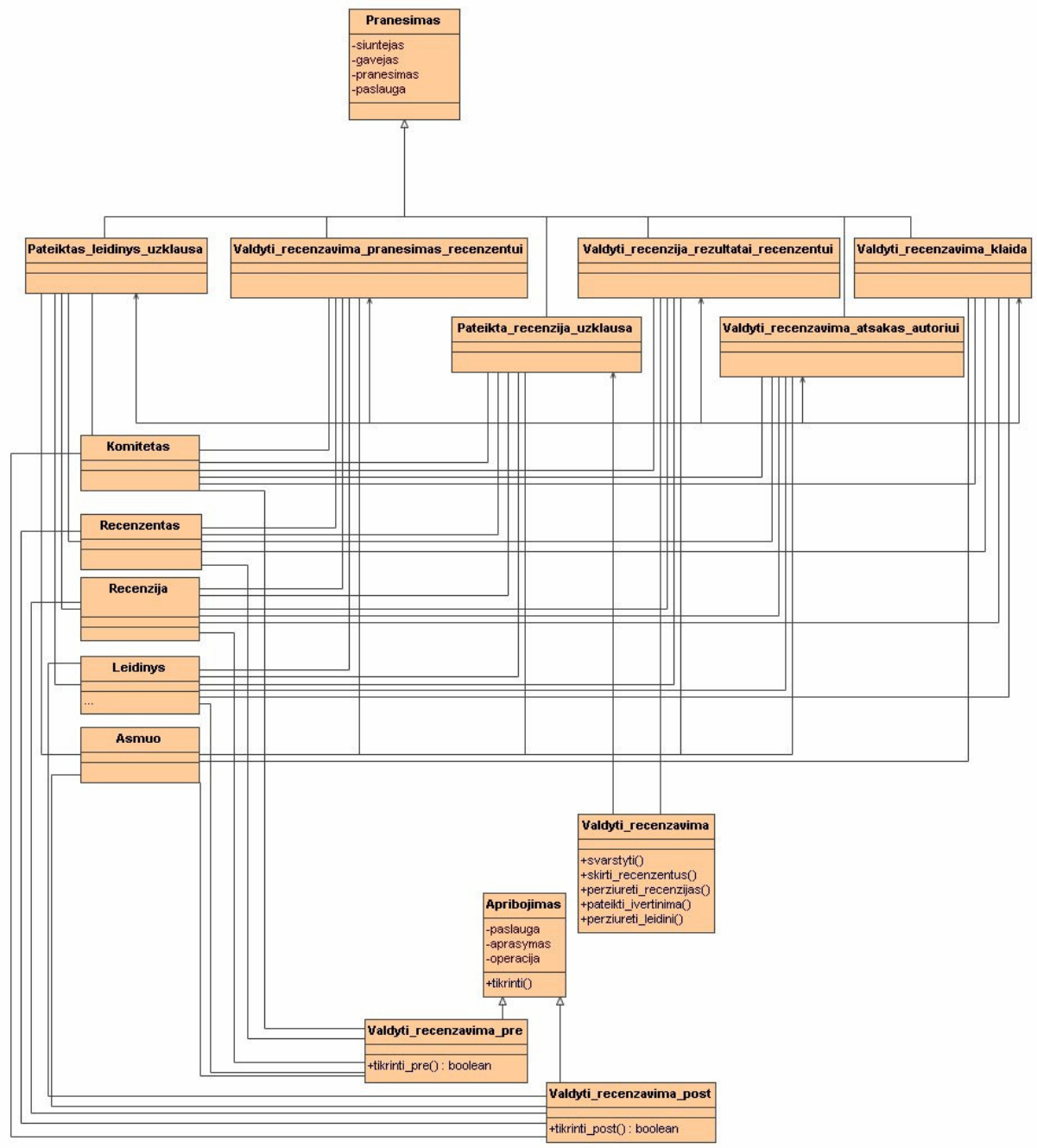
24 pav. Paketas “Registruotis”

Paslaugos „Pateikti_leidini“ paketas:



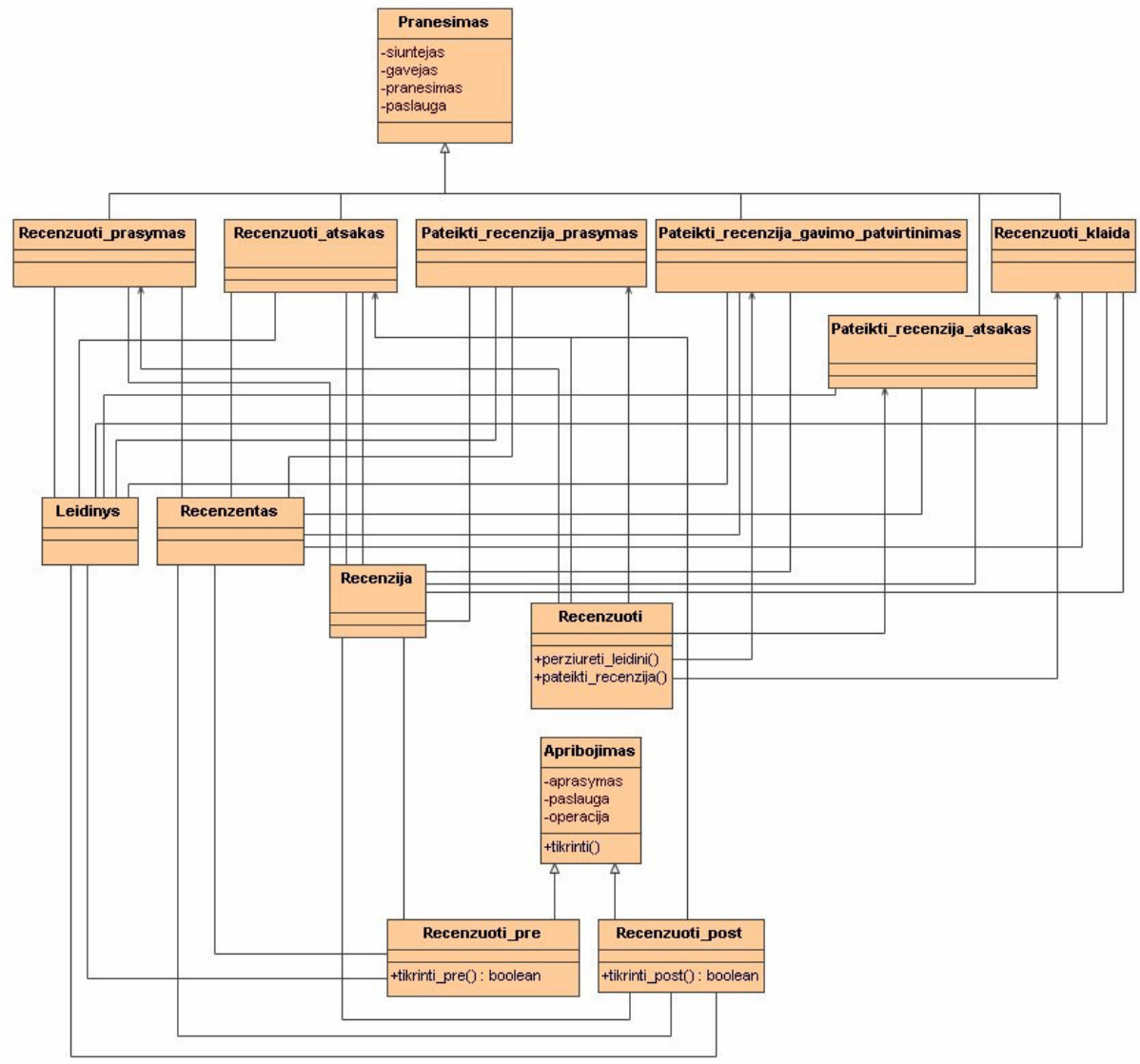
25 pav. Paketas “pateikti_leidini”

Paslaugos „Valdyti_recenzavima“ paketas:



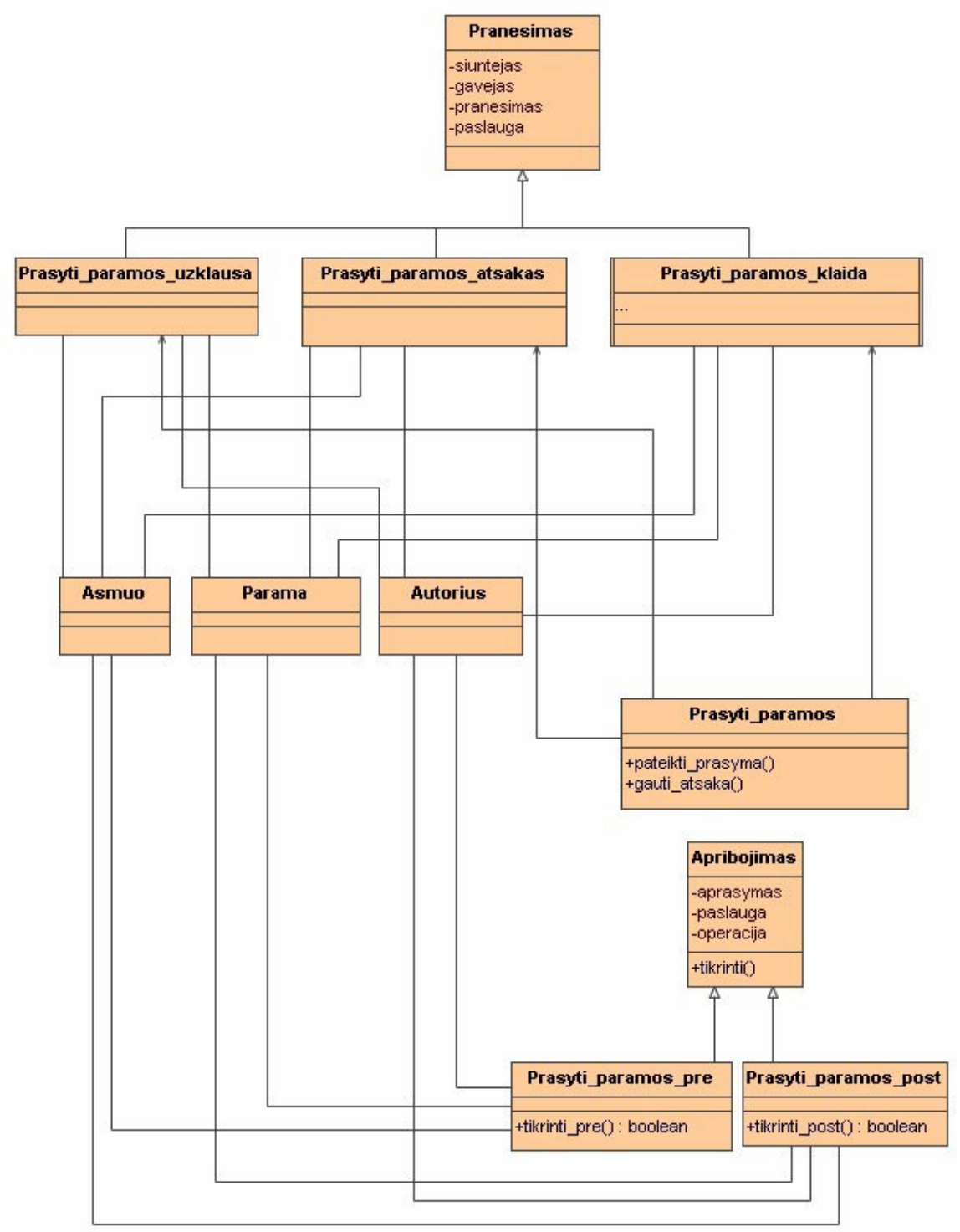
26 pav. Paketas “Valdyti_recenzavima”

Paslaugos „Recenzuoti“ paketas:



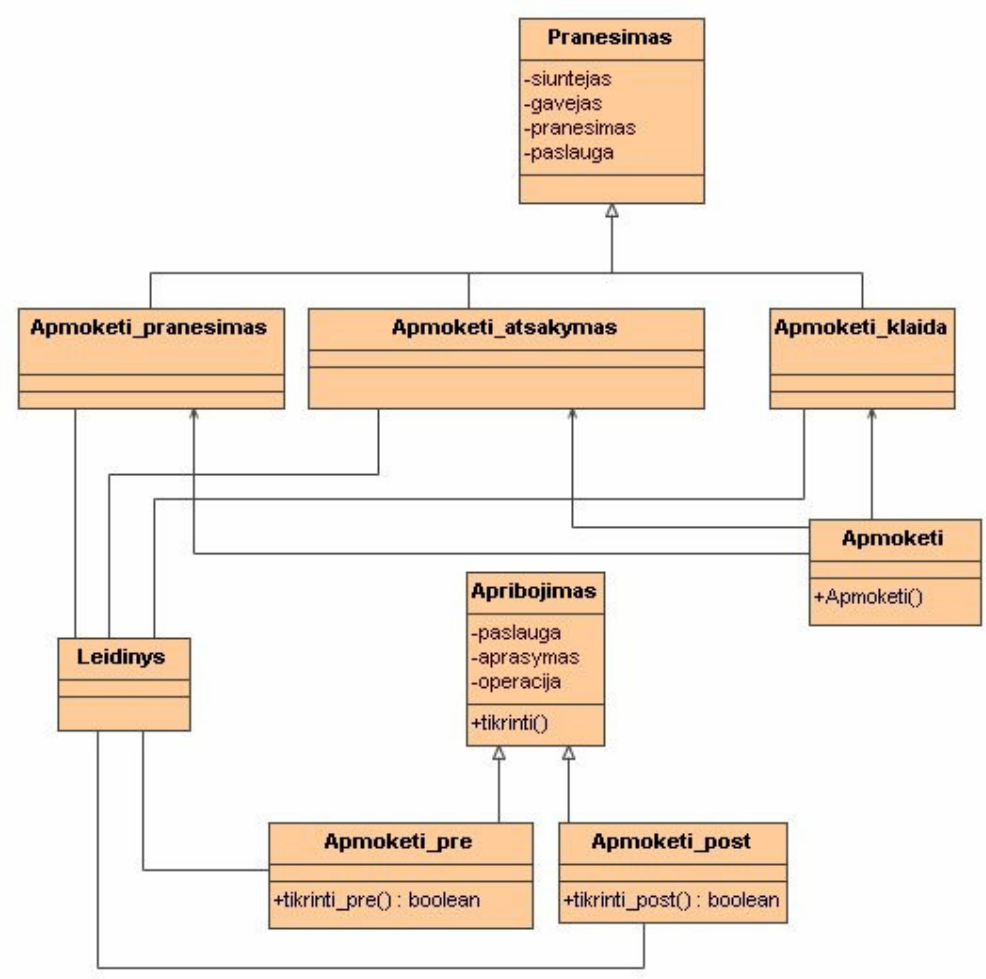
27 pav. Paketas “Recenzuoti”

Paslaugos „Prasyti_paramos“ paketas:



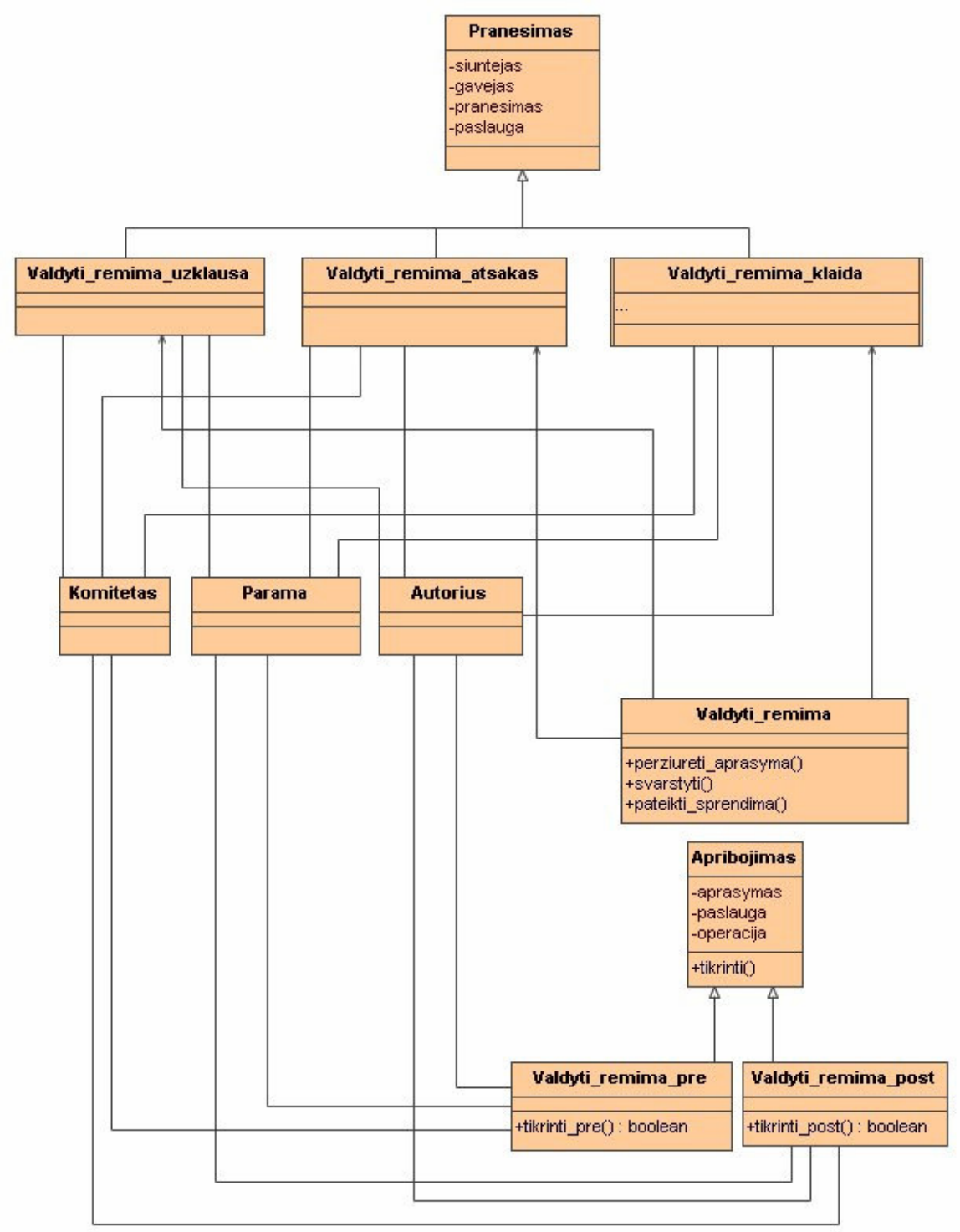
28 pav. Paketas “Prasyti_paramos”

Paslaugos „Apmoketi“ paketas:



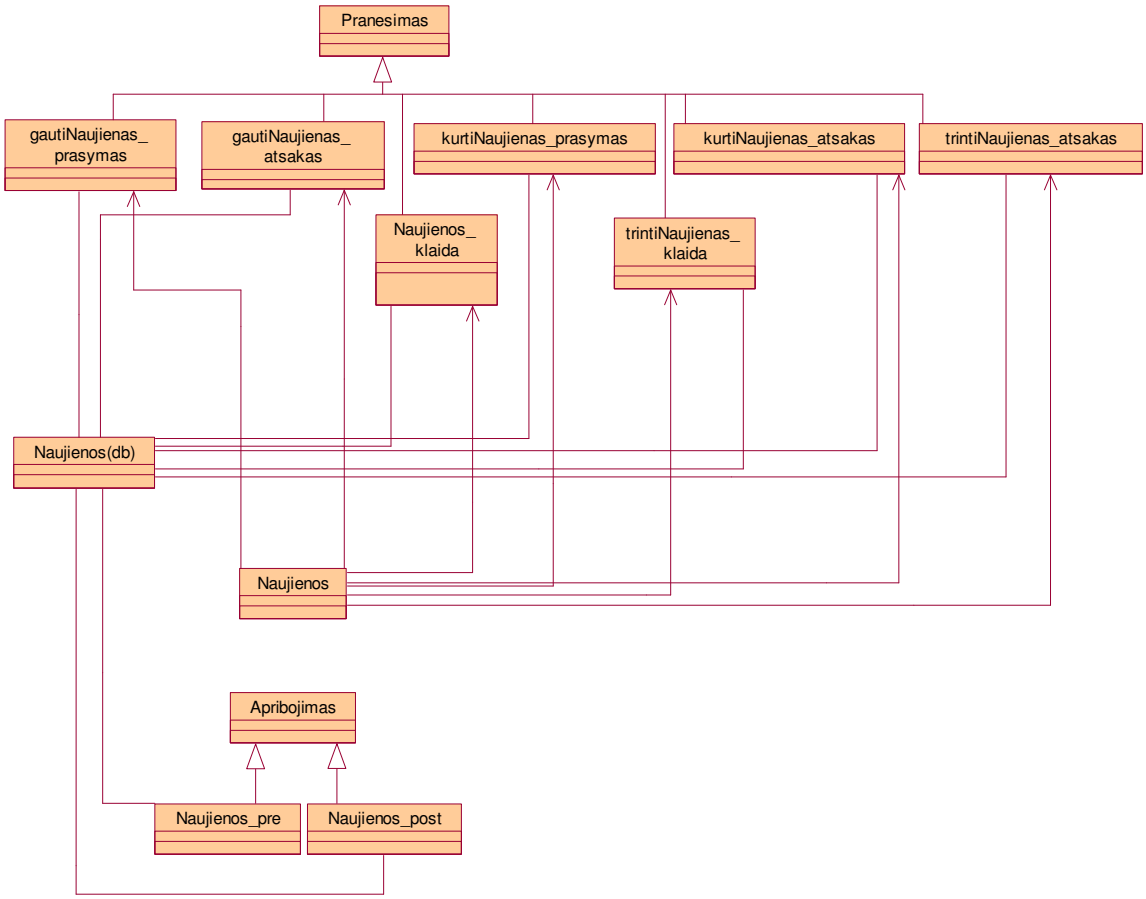
29 pav. Paketas “Apmoketi”

Paslaugos „Valdyti_remima“ paketas:



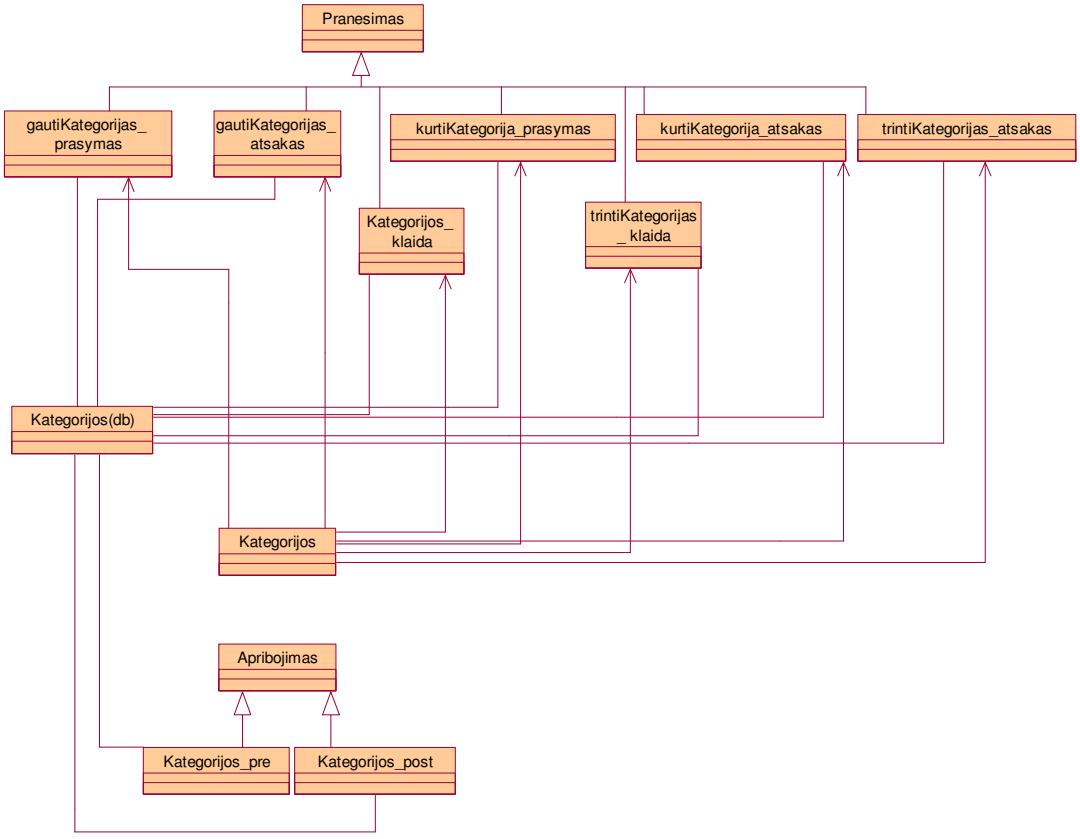
30 pav. Paketas “Valdyti_remima”

Paslaugos “Naujienos” paketas:



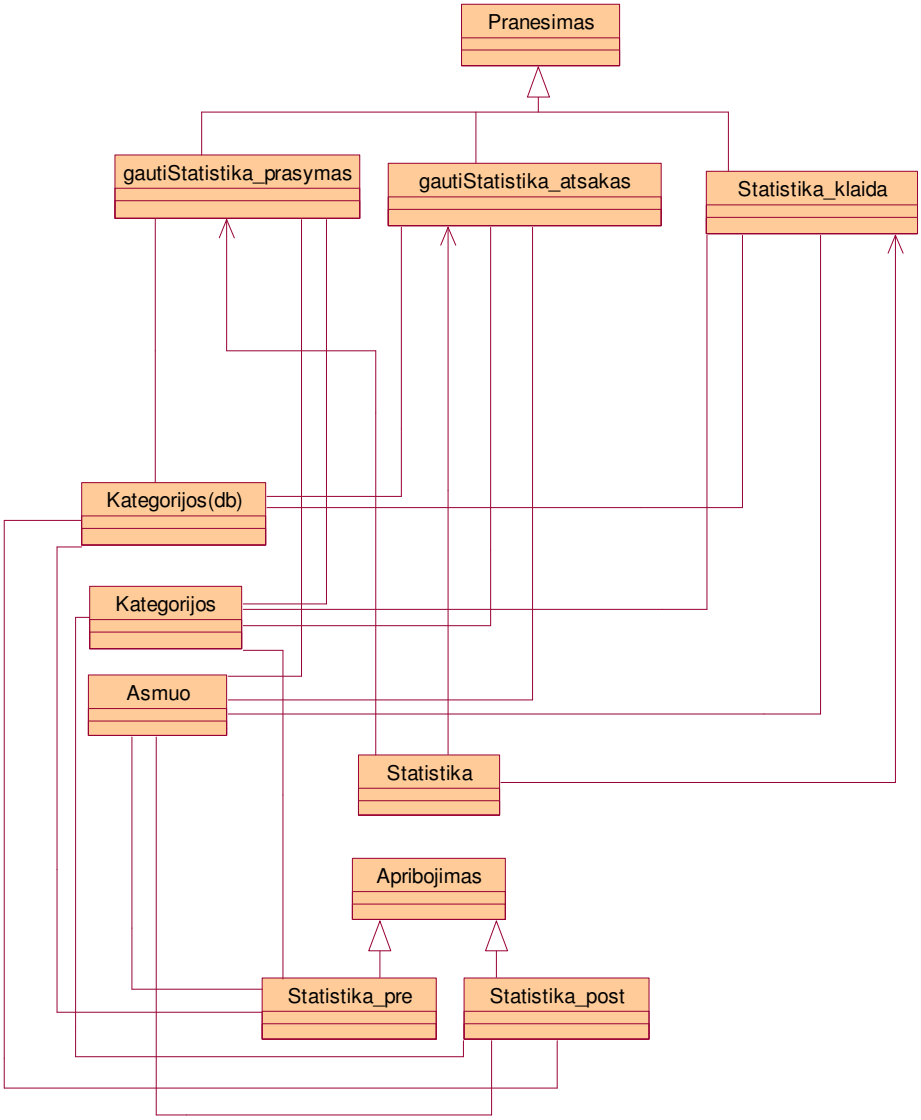
31 pav. Paketas “Naujienos”

Paslaugos “Kategorijos” paketas:



32 pav. Paketas “Kategorijos”

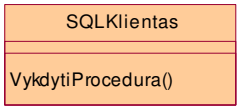
Paslaugos “Statistika” paketas:



33 pav. Paketas “Statistika”

Paketas „DuomenuPaslaugos“

Paketas skirtas duomenų bazės abstrakcijos klasėms. Tai klasės, skirtos darbui su duomenų baze. Jos naudojamos daugumoje paketų.

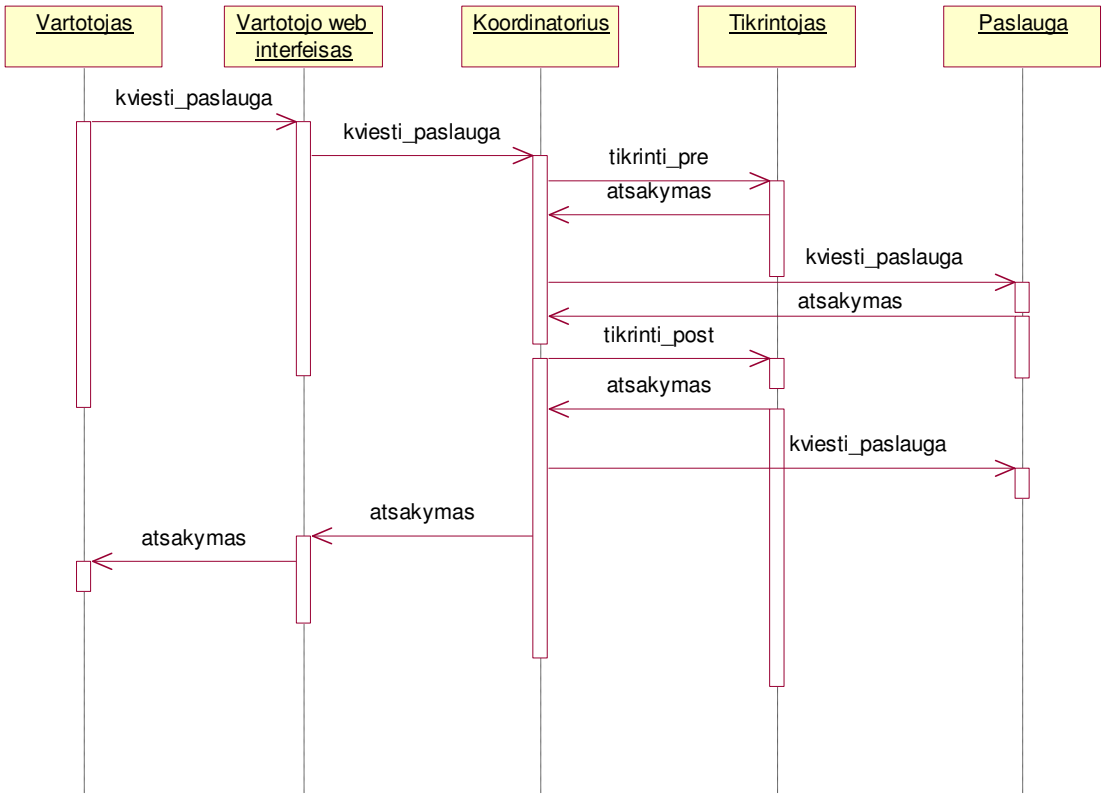


34 pav. Paketas “DuomenuPaslaugos”

4.4. Koordinatoriaus sąveikos su elementariomis paslaugomis

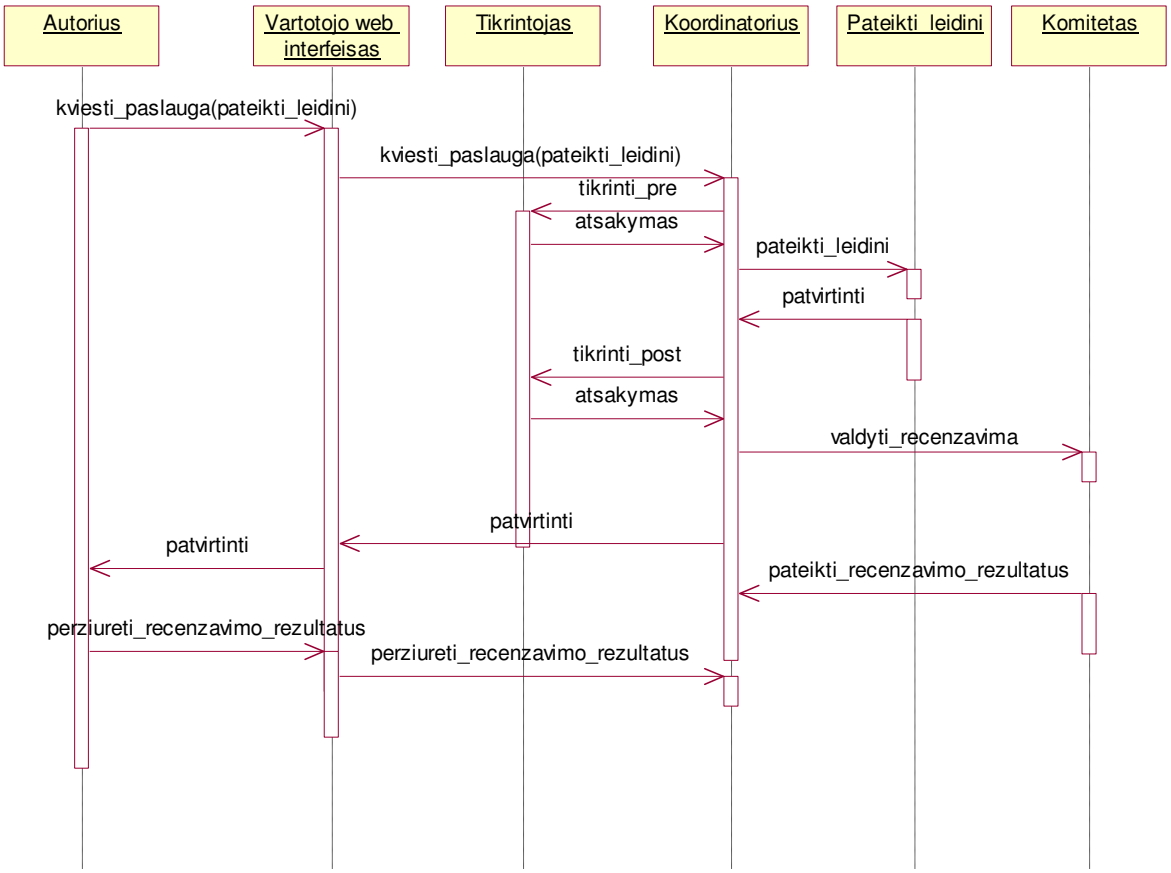
Sekų diagrama

Koordinatorius, gavęs pranešimą, visais atvejais elgiasi vienodai: tikrina paslaugos prieš-sąlygą, ir, jei ji tenkinama, kviečia paslaugą. Gavęs atsaką, koordinatorius tikrina paslaugos po sąlygą ir siunčia atsaką vartotojui. Apibendrinta sekų diagrama:



35 pav. Apibendrinta sekų diagrama

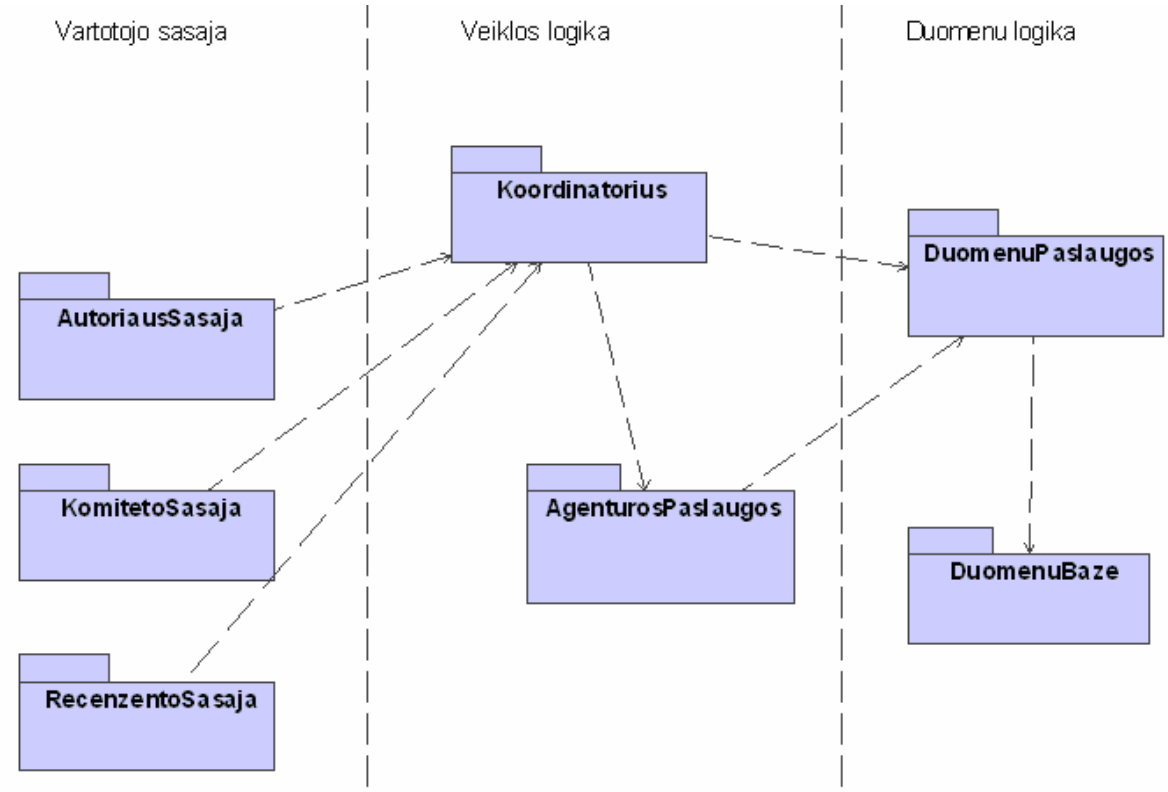
Kitų paslaugų sekų diagramos yra beveik analogiškos, todėl žemiau pateikiamas vienas atvejis „pateikti leidini“:



36 pav. Panaudojimo atvejo „pateikti leidini“ sekų diagrama

4.5. Sistemos komponentų aprašymas

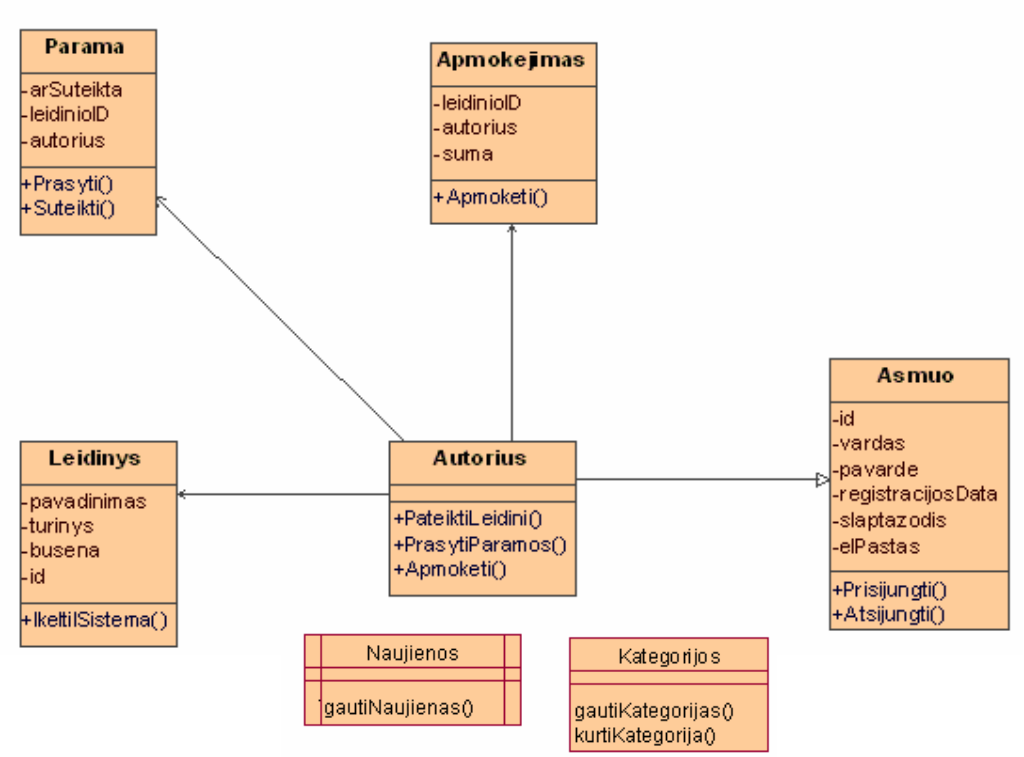
Sistema suskaidyta į 7 paketus aukščiausiame lygyje.



37 pav. Sistemos išskaidymas į paketus

Paketas „Autoriaus sąsaja“ atsakingas už leidinio pateikimą, autoriaus paramos prašymo valdymą, apmokėjimą. Paketas „Komiteto sąsaja“ atsakingas už ataskaitų generavimą, recenzijų vertinimą, recenzentų paskyrimą, paramos valdymą, recenzavimo valdymą. Paketas “Recenzento sąsaja“ atsakingas už leidinio recenzavimą bei recenzijos pateikimą. Paketas „Koordinatorius“ priima iš vartotojo sąsajos ateinančias užklausas ir nukreipia jas į tikrintoją, kuris tikrina, ar užklausą galima vykdyti. Jei tikrintojas patvirtina, kad užklausą galima vykdyti, koordinatorius kviečia atitinkamą paslaugą. Paketas „Koordinatorius“ atsakingas už užklausos priėmimą, pre ir post sąlygų patikrinimą, atitinkamos paslaugos iškvietimą, pranešimo siuntimą pradinės užklausos siuntėjui ir pan. Paketas „Agentūros paslaugos“ realizuoja nepriklausomas paslaugas. Paketas „Duomenų paslaugos“ atsakingas už darbą su duomenų baze.

4.5.1. Autoriaus sąsaja

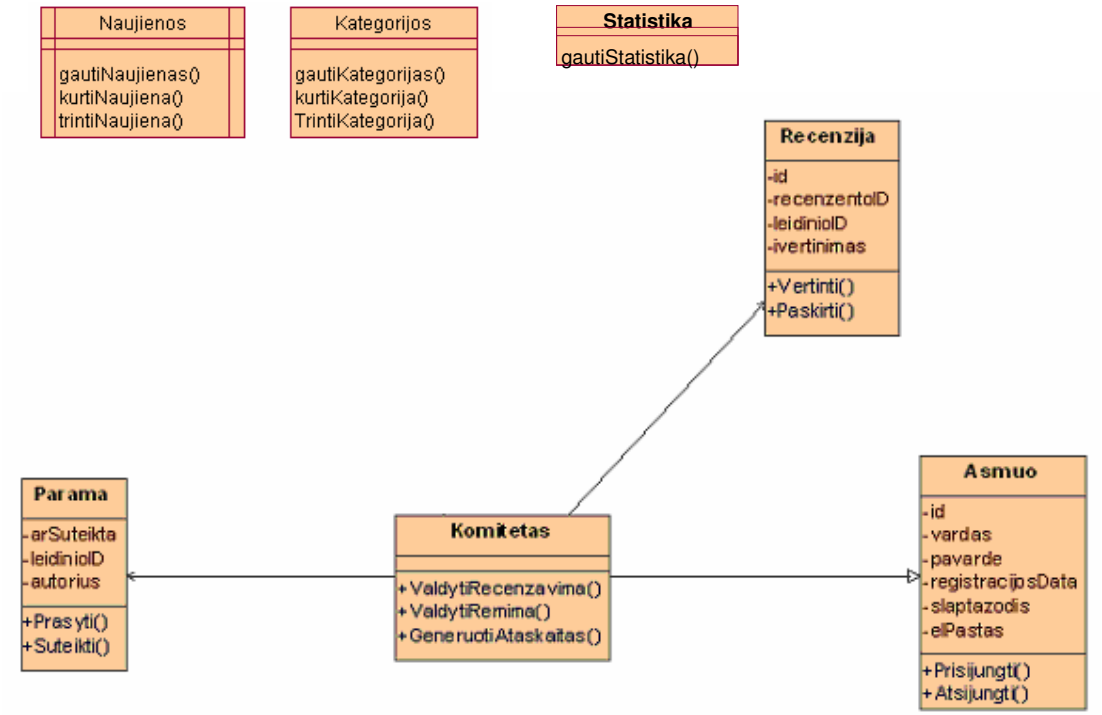


38 pav. Autoriaus sąsaja

3 pav. Autoriaus sąsajos posistemio klasių diagrama

Klasifikacija:	Posistemis
Apibrėžimas:	Šiame pakete pateiktos klasės realizuojančios autoriaus sąsają
Atsakomybės:	Leidinio pateikimas, paramos prašymas, apmokėjimas.
Apribojimai:	
Struktūra:	Posistemį sudaro klasės, pavaizduotos 3 pav.
Sąveikavimas:	Sąveikauja su posistemiu „Koordinatorius“
Resursai:	Posistemis naudoja leidinių agentūros paslaugas.
Sąsaja/eksportas:	Posistemio sąsają sudaro visų jo klasių sąsajų visuma.

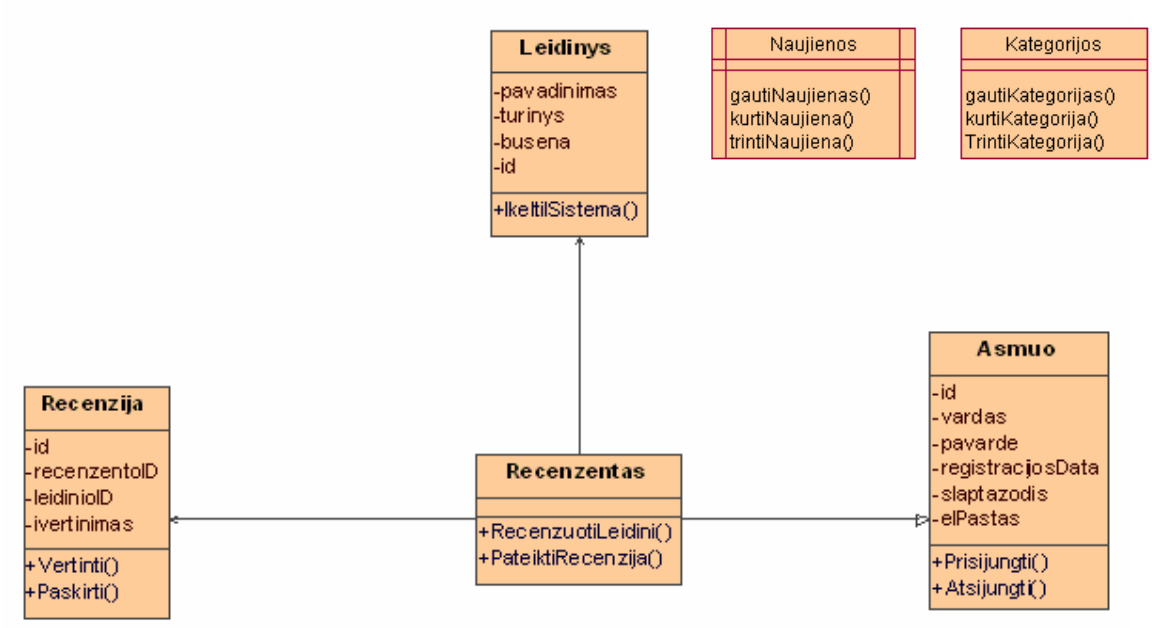
4.5.2. Komiteto sąsaja



39 pav. Komiteto sąsajos posistemio klasių diagrama

Klasifikacija:	Posistemis
Apibrėžimas:	Šiame pakete pateiktos klasės realizuojančios recenzento sąsają
Atsakomybės:	Ataskaitų generavimas, recenzijų vertinimas, recenzentų paskyrimas, paramos valdymas, recenzavimo valdymas
Apribojimai:	
Struktūra:	Posistemį sudaro klasės, pavaizduotos 4 pav.
Sąveikavimas:	Sąveikauja su posistemiu „Koordinatorius“
Resursai:	Posistemis naudoja duomenų bazės lenteles Asmuo, Parama, Recenzija
Sąsaja/eksportas:	Posistemio sąsają sudaro visų jo klasių sąsajų visuma.

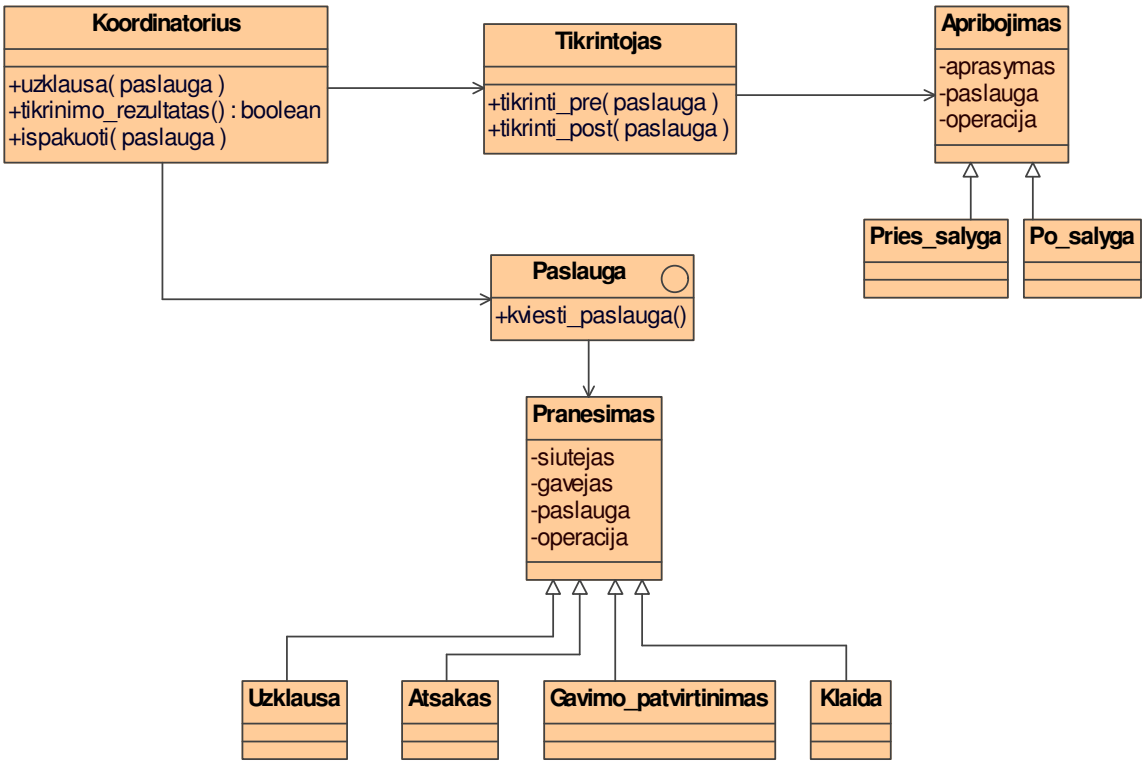
4.5.3. Recenzento sąsaja



40 pav. Recenzento sąsajos posistemio klasių diagrama

Klasifikacija:	Posistemis
Apibrėžimas:	Šiame pakete pateiktos klasės realizuojančios komiteto sąsają
Atsakomybės:	Leidinio recenzavimas, recenzijos pateikimas
Apribojimai:	
Struktūra:	Posistemį sudaro klasės, pavaizduotos 5 pav.
Sąveikavimas:	Sąveikauja su posistemiu „Koordinatorius“
Resursai:	Posistemis naudoja leidinių agentūros paslaugas.
Sąsaja/eksportas:	Posistemio sąsają sudaro visų jo klasių sąsajų visuma.

4.5.4. Koordinatorius

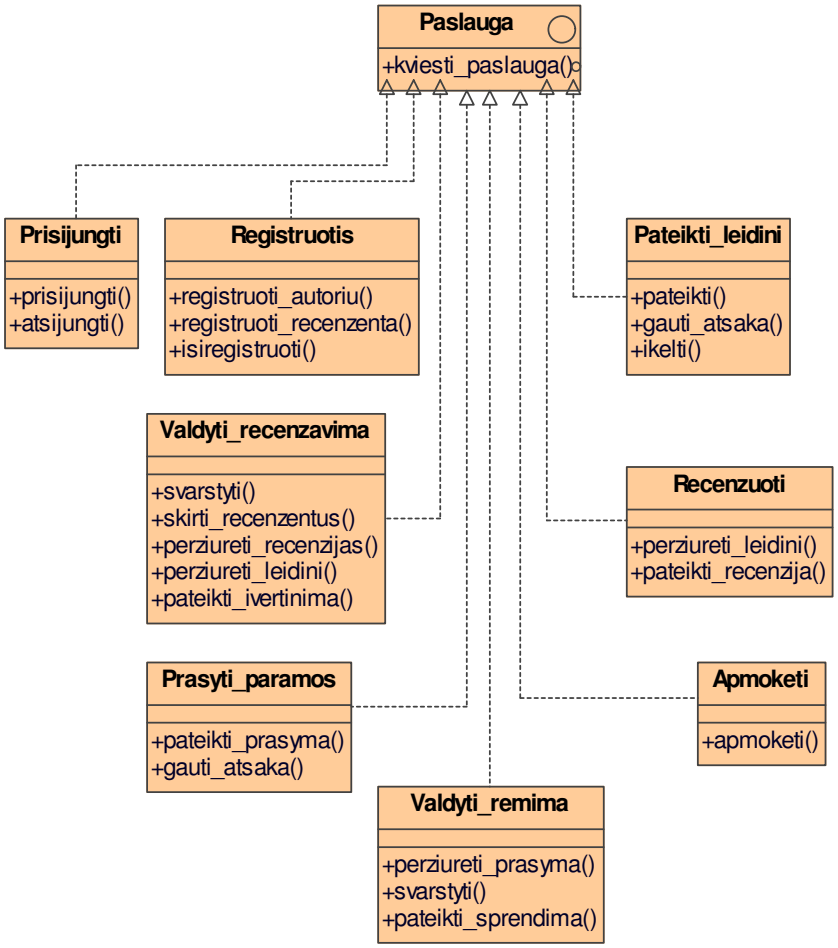


41 pav. Koordinatoriaus posistemio klasių diagrama

Klasifikacija:	Posistemis
Apibrėžimas:	„Koordinatorius“ saugomos klasės, kurios priima iš vartotojo sąsajos ateinančias užklausas, nukreipia jas į tikrintoją, kuris tikrina, ar užklausą galima vykdyti. Jei tikrintojas patvirtina, kad užklausą galima vykdyti, koordinatorius kviečia atitinkamą paslaugą. „Paslauga“, „Apribojimas“ ir „Pranešimas“ yra šabloninės klasės, vietoje jų įstatomos konkrečios klasės, kurios aprašytos Leidinių agentūros pakete
Atsakomybės:	Užklausos priėmimas, pre ir post sąlygų patikrinimas, atitinkamos paslaugos iškvietimas, pranešimo siuntimas pradinės užklausos siuntėjui.
Apribojimai:	

Struktūra:	Posistemį sudaro klasės, pavaizduotos 6 pav.
Sąveikavimas:	Sąveikauja su posistemiais: Prisijungimas, AutoriausSasaja, KomitetoSasaja, RecenzentoSasaja. Gautas užklausas persiunčia/sąveikauja su posistemiu AgentūrosPaslaugos. Tikrinant pre ir post sąlygas – sąveikauja su posistemiu DuomenuPaslaugos
Resursai:	Posistemis (Tikrintojas klasė) pagal poreikį gali naudoti duomenų bazės lenteles: Asmuo, Leidinys, Recenzija, Parama, Apmokejimas, Ivertinimas, Konstantos.
Sąsaja/eksportas:	Posistemio sąsają sudaro visų jo klasių sąsajų visuma.

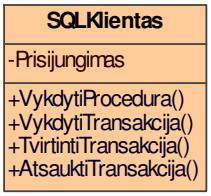
4.5.5. Agentūros paslaugos



42 pav. Agentūros paslaugų posistemio klasių diagrama

Klasifikacija:	Posistemis
Apibrėžimas:	Šiame pakete saugomos klasės, kurios realizuoja nepriklausomas paslaugas. Visos paslaugos realizuoja bendrą interfeisą, kuris iškviečia paslaugą pagal jos vardą. Kiekviena paslauga aprašyta atskirame pakete, kadangi ji turi savo pranešimus ir tikrinimo operacijas
Atsakomybės:	Prisijungimas, registravimas, leidinio pateikimas, recenzavimas, recenzavimo valdymas, paramos prašymas, rėmimo valdymas, apmokėjimas.
Apribojimai:	
Struktūra:	Posistemį sudaro klasės, pavaizduotos 7 pav.
Sąveikavimas:	Sąveikauja su posistemiais Koordinatorius ir DuomenuPaslaugos
Resursai:	Posistemis pagal poreikį gali naudoti duomenų bazės lenteles: Asmuo, Leidinys, Recenzija, Parama, Apmokejimas, Ivertinimas, Konstantos.
Sąsaja/eksportas:	Posistemio sąsają sudaro visų jo klasių sąsajų visuma.

4.5.6. Duomenų paslaugos



43 pav. Duomenų paslaugų posistemio klasių diagrama

Klasifikacija:	Posistemis

Apibrėžimas:	Pakete pateikiamos klasės, skirtos darbui SQL Server duomenų bazėmis.
Atsakomybės:	Darbas su duomenų baze.
Apribojimai:	Darbui su duomenų baze naudojama ADO.NET technologija
Struktūra:	Posistemį sudaro klasės, pavaizduotos 8 pav.
Sąveikavimas:	Šio paketo klasės naudojamos pakete Koordinatorius bei AgenturosPaslaugos. Klaidos šiame pakete privers klaidingai veikti visą sistemą.
Resursai:	Paketas naudoja domenų bazę (MS SQL Server) duomenims saugoti.
Sąsaja/eksportas:	Duomenų abstrakcijos sąsaja pateikiama kaip klasė SqlKlientas su viešais metodais duomenims duomenų bazėje įrašyti, atnaujinti ar trinti.

4.6. Detalūs paslaugų sistemos klasių aprašymai

4.6.1. Parama

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo autoriaus paramos prašymo ir jos suteikimo valdymą.
Atsakomybės:	Paramos prašymas ir jos suteikimas.
Apribojimai:	Paramos prašyti gali tik autorius, kurio darbas teigiamai įvertintas recenzentų
Struktūra:	Klasės struktūra pateikta 3 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per klasę „Autorius“
Resursai:	Naudojamos duomenų bazės lentelės: Parama
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.2. Apmokėjimas

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo autoriaus apmokėjimo už leidinio išleidimą valdymą.
Atsakomybės:	Apmokėjimo valdymas.
Apribojimai:	Leidinio išleidimą apmoka autorius, kuris neprašė paramos arba kurio prašymas dėl paramos buvo atmestas
Struktūra:	Klasės struktūra pateikta 3 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per klasę „Autorius“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.3. Leidinys

Klasifikacija:	Klasė
Apibrėžimas:	Klasė atsakinga už informacijos apie leidinį bei pačio leidinio sistemoje valdymą.
Atsakomybės:	Leidinio įkėlimas į sistemą.
Apribojimai:	Leidinį įkelti gali tik prisiregistravęs ir prisijungęs autorius.
Struktūra:	Klasės struktūra pateikta 3 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per klasę „Autorius“
Resursai:	Naudojamos duomenų bazės lentelės: Leidinys
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.4. Autorius

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo autoriaus sąsajos valdymą.

Atsakomybės:	Paramos prašymas, leidinio pateikimas, apmokėjimas, recenzavimo rezultatų peržiūra.
Apribojimai:	Minėtus veiksmus gali atlikti tik sistemoje užsiregistravęs ir prie jos prisijungęs autorius
Struktūra:	Klasės struktūra pateikta 3 pav.
Sąveikavimas:	Komponentas iškviečia kitas posistemio klases – „Parama“, „Apmokėjimas“, „Leidinys“, „Asmuo“.
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.5. Asmuo

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo prisijungimą ir atsijungimą.
Atsakomybės:	Asmenų prisijungimas ir atsijungimas.
Apribojimai:	
Struktūra:	Klasės struktūra pateikta 2 pav.
Sąveikavimas:	
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.6. Komitetas

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo komiteto sąsajos valdymą.
Atsakomybės:	Recenzavimo ir rėmimo valdymas, ataskaitų generavimas.
Apribojimai:	Minėtus veiksmus gali atlikti tik sistemoje užsiregistravęs ir prie jos prisijungęs komiteto narys

Struktūra:	Klasės struktūra pateikta 4 pav.
Sąveikavimas:	Komponentas iškviečia kitas posistemio klases – „Parama“, „Ataskaita“, „Recenzija“, „Asmuo“.
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Recenzija, Parama
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.7. Recenzija

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo recenzijų ir recenzentų valdymą.
Atsakomybės:	Recenzentų paskyrimas, leidinių vertinimas pagal recenzijas.
Apribojimai:	Minėtus veiksmus gali atlikti tik sistemoje užsiregistravęs ir prie jos prisijungęs komiteto narys. Leidinys, kuriam paskiriamas recenzentas, turi būti įkeltas į sistemą. Leidinys gali būti vertinamas tik tada, kai į sistemą įkeltos visos jį apibūdinančios recenzijos.
Struktūra:	Klasės struktūra pateikta 4 pav.
Sąveikavimas:	Komponentą iškviečia kita posistemio klasė – „Komitetas“.
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Recenzija, Vertinimas
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.8. Naujienos

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo naujienų atvaizdavimą, koregavimą
Atsakomybės:	Naujienų atvaizdavimas, koregavimas
Apribojimai:	Peržiūrėti naujienas gali visi asmenys. Koreguoti – tik prisijungę komiteto nariai.

Struktūra:	Klasės struktūra pateikta 4 pav.
Sąveikavimas:	Komponentą iškviečia kita posistemio klasė – „Komitetas“.
Resursai:	Naudojamos duomenų bazės lentelės: Naujienos
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.9. Kategorijos

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo kategorijų atvaizdavimą, koregavimą
Atsakomybės:	Kategorijų atvaizdavimas, koregavimas
Apribojimai:	Peržiūrėti kategorijas gali visi asmenys. Trinti ar kurti – tik prisijungę komiteto nariai. Autorius gali kurti naują kategoriją, be tokiu atveju ji nebus matoma kol komitetas jos nepatvirtins.
Struktūra:	Klasės struktūra pateikta 4 pav.
Sąveikavimas:	Komponentą iškviečia kita posistemio klasė – „Komitetas“.
Resursai:	Naudojamos duomenų bazės lentelės: Kategorijos
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.10. Statistika

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo statistikos atvaizdavimą.
Atsakomybės:	Statistikos atvaizdavimas
Apribojimai:	Minėtus veiksmus gali atlikti tik sistemoje užsiregistravęs ir prie jos prisijungęs komiteto narys.
Struktūra:	Klasės struktūra pateikta 4 pav.

Sąveikavimas:	Komponentą iškviečia kita posistemio klasė – „Komitetas“.
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys, Kategorijos
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.11. Recenzentas

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo recenzento sąsajos valdymą.
Atsakomybės:	Recenzuojamo kūrinio pateikimas recenzentui, recenzijos įkėlimas į sistemą.
Apribojimai:	Minėtus veiksmus gali atlikti tik sistemoje užsiregistravęs ir prie jos prisijungęs recenzentas. Recenzuojamas leidinys turi būti autoriaus įkeltas į sistemą, o recenzentas turi būti paskirtas komiteto šį leidinį recenzuoti.
Struktūra:	Klasės struktūra pateikta 5 pav.
Sąveikavimas:	Komponentas iškviečia kitas posistemio klases – „Recenzija“, „Leidinys“, „Asmuo“.
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys, Recenzija
Sąsaja/eksportas:	Klasės metodai aprašyti žemiau

4.6.12. Registruotis

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo autorių, recenzentų registraciją.
Atsakomybės:	Asmenų registracija

Apribojimai:	Kiekvienas vartotojas turi būti unikalus. Recenzentais gali registruotis asmenys, gavę administratoriaus kvietimą, todėl sistema tikrina recenzentų duomenis ir tik tada patvirtina jų registraciją
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per kitą komponentą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.13 Prisijungti

Klasifikacija:	Klasė
Apibrėžimas:	Vykdo autorių, recenzentų prisijungimą ir atsijungimą.
Atsakomybės:	Asmenų prisijungimas ir atsijungimas.
Apribojimai:	
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per kitą komponentą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.14. Pateikti_leidini

Klasifikacija:	Klasė
Apibrėžimas:	Klasė, apjungianti funkcijas, reikalingas autoriui pateikti leidinį.
Atsakomybės:	Leidinio pateikimas.

Apribojimai:	Autorius turi būti registruotas
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per interfeisą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys, Ivertinimas
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.15. Valdyti_recenzavima

Klasifikacija:	Klasė
Apibrėžimas:	Klasė, apjungianti funkcijas, reikalingas recenzavimo valdymui.
Atsakomybės:	Recenzentų skyrimas, recenzijų peržiūrėjimas, leidinių peržiūrėjimas, leidinio įvertinimas
Apribojimai:	Turi būti pateiktas leidinys
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per kitą komponentą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys, Recenzija, Ivertinimas, Konstantos.
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.16. Recenzuoti

Klasifikacija:	Klasė
Apibrėžimas:	Klasė, apjungianti funkcijas, reikalingas recenzuoti leidinį.
Atsakomybės:	Leidinio peržiūrėjimas, recenzijos pateikimas
Apribojimai:	Recenzentas turi parašyti recenziją per nustatytą laiką

Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per interfeisą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Leidinys, Recenzija
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.17. Prasyti_paramos

Klasifikacija:	Klasė
Apibrėžimas:	Klasė, apjungianti funkcijas, reikalingas paramos prašymui
Atsakomybės:	Paramos prašymas, atsakymo gavimas.
Apribojimai:	Autorius prašyti paramos, jei jo leidinys yra priimtas ir jei jis nėra gavęs daugiau nustatyto skaičiaus paramų
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per kitą komponentą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys, Parama
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.18. Valdyti_remima

Klasifikacija:	Klasė
Apibrėžimas:	Klasė, apjungianti funkcijas, reikalingas rėmimo valdymui.
Atsakomybės:	Rėmimo prašymo peržiūrėjimas, rėmimo svarstymas, rėmimo sprendimo pateikimas.
Apribojimai:	Komitetas turi priimti sprendimą per nustatytą laiką
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per kitą komponentą „Paslauga“

Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys, Ivertinimas, Konstantos.
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.19. Apmoketi

Klasifikacija:	Klasė
Apibrėžimas:	Klasė, susijusi su apmokėjimu.
Atsakomybės:	Pažymėjimas, kad leidinys apmokėtas.
Apribojimai:	Pažymėti apmokėjimą gali tik komitetas
Struktūra:	Klasės struktūra pateikta 7 pav.
Sąveikavimas:	Komponentas yra iškviečiamas per kitą komponentą „Paslauga“
Resursai:	Naudojamos duomenų bazės lentelės: Asmuo, Leidinys
Sąsaja/eksportas:	Metodai aprašyti žemiau

4.6.20. SQL Klientas

Klasifikacija:	Klasė
Apibrėžimas:	Klasė skirta darbui su SQL (MS SQL Server) duomenų baze.
Atsakomybės:	Darbas su SQL duomenų baze. Sukurtų procedūrų vykdymas, transakcijų vykdymas, patvirtinimas, bei atšaukimas
Apribojimai:	
Struktūra:	Klasės struktūra pateikta klasių diagramoje
Sąveikavimas:	Klasė naudojama pakete Koordinatorius bei AgenturosPaslaugos.
Resursai:	Klasė naudoja SQL duomenų bazė, ADO.NET bibliotekos.

Sąsaja/eksportas:	Metodai VykdytiProcedura

5. EKSPERIMENTINĖ PASLAUGŲ SISTEMOS REALIZACIJA IR TESTAVIMAS

Ši programinė įranga skirta leidykloms ir leidybos agentūroms. Ji padeda koordinuoti šias užduotis, susijusias su publikuojamais darbais bei kasdieniu leidyklos darbu:

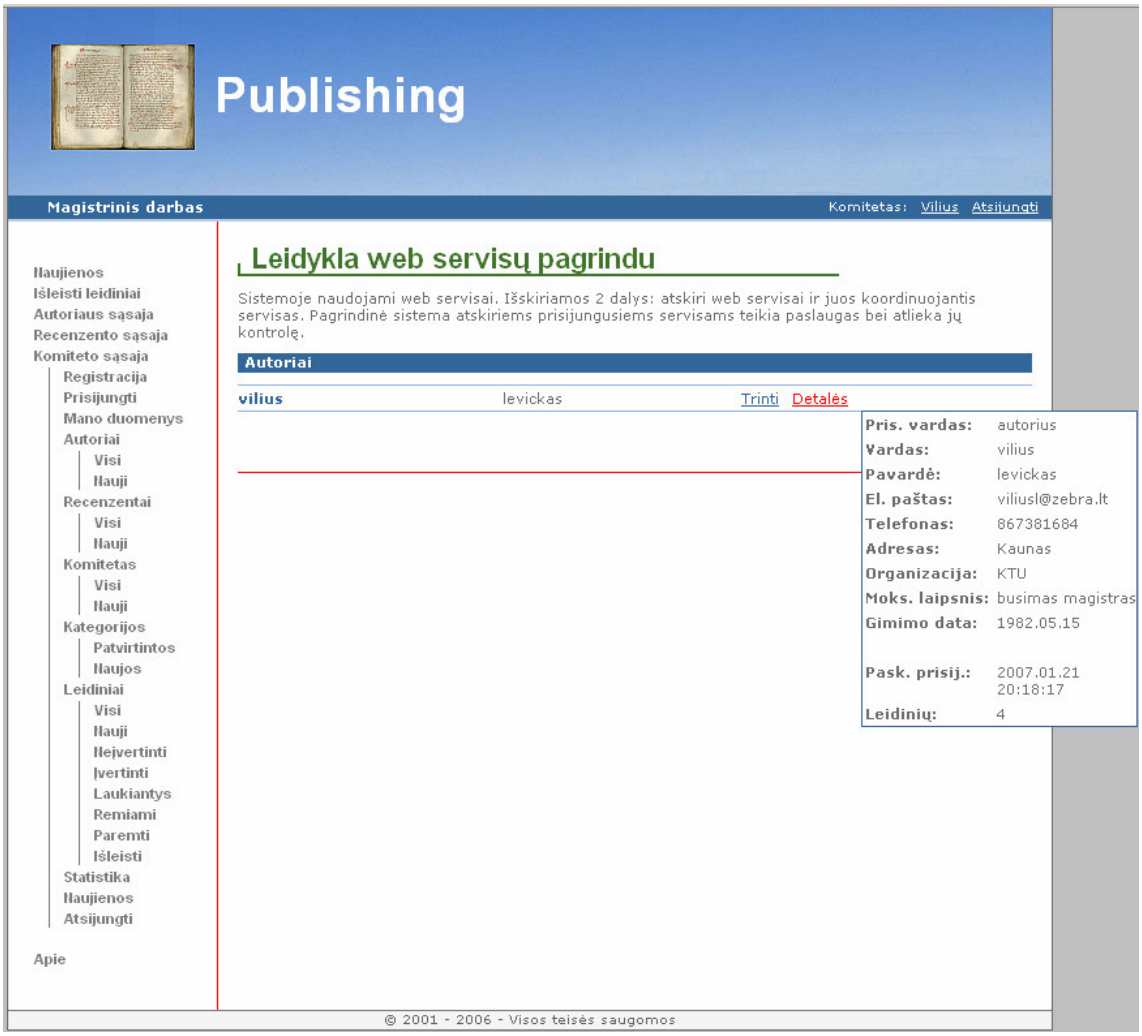
- Parašytų darbų atsiuntimą į leidyklos informacinę sistemą
- Publikuoti ruošiamų darbų recenzavimą
- Finansavimo skyrimą kai kuriems darbams išleisti
- Sprendimą publikuoti konkretų darbą
- Informacijos apie ruošiamus publikuoti darbus ir jų statusą pateikimą realiu laiku
- Ataskaitų formavimą ir spausdinimą

Informacine sistema gali naudotis kelių tipų vartotojai:

1. **Autoriai.** Jie gali registruotis prie sistemos ir siųsti savo publikuoti paruoštus darbus į bendrą duomenų bazę.
2. **Recenzentai.** Jie skiriami recenzuoti autorių darbus. Recenzentus skiria komitetas.
3. **Komiteto nariai.** Komitetas atsakingas už recenzentų skyrimą bei sprendimus dėl autorių darbų publikavimo bei finansavimo. Komitetas leidybos procese turi galutinį sprendžiamąjį balsą.

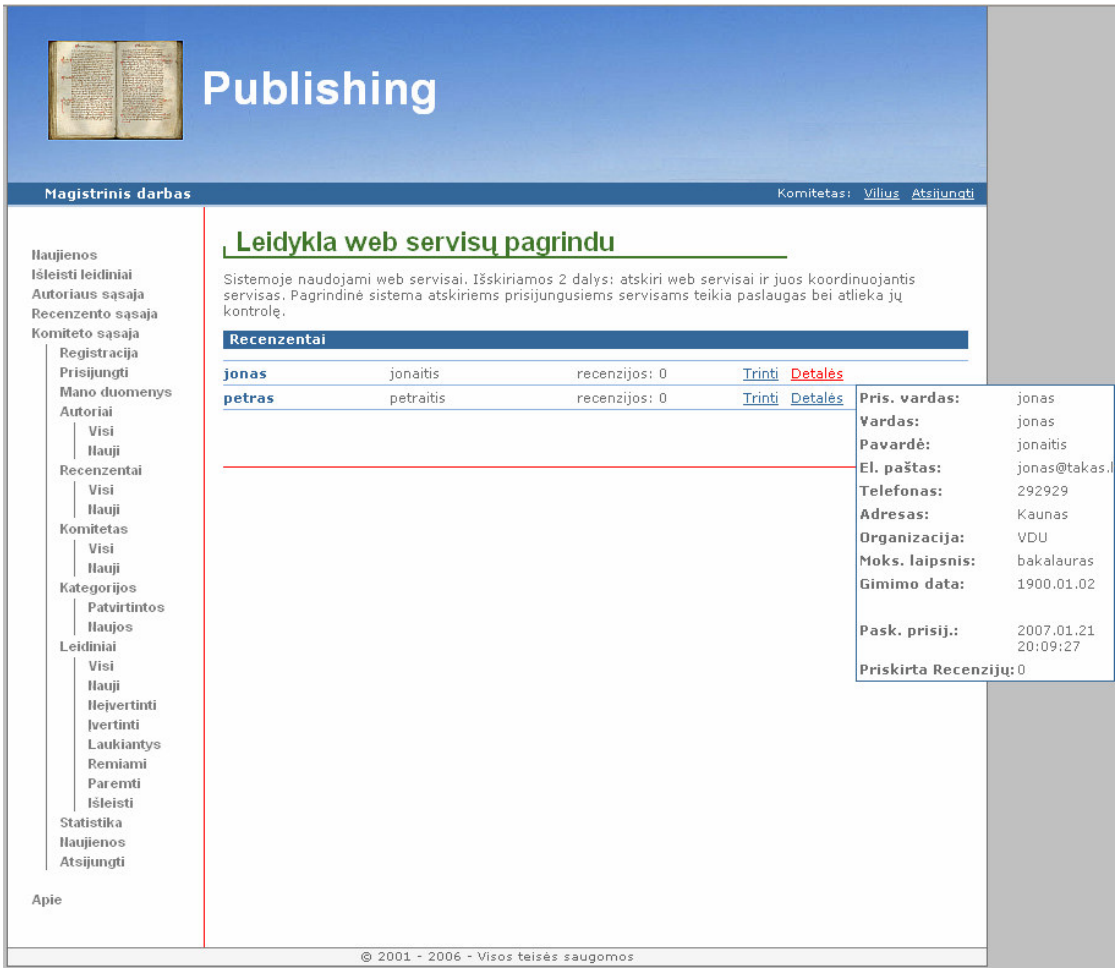
Komiteto funkcijos:

1. Peržiūrėti sistemoje registruotus autorius (komitetas gali peržiūrėti sistemoje registruotus autorius ir juos valdyti (registruoti, redaguoti, šalinti). Pasiekimas – „Autoriai“.



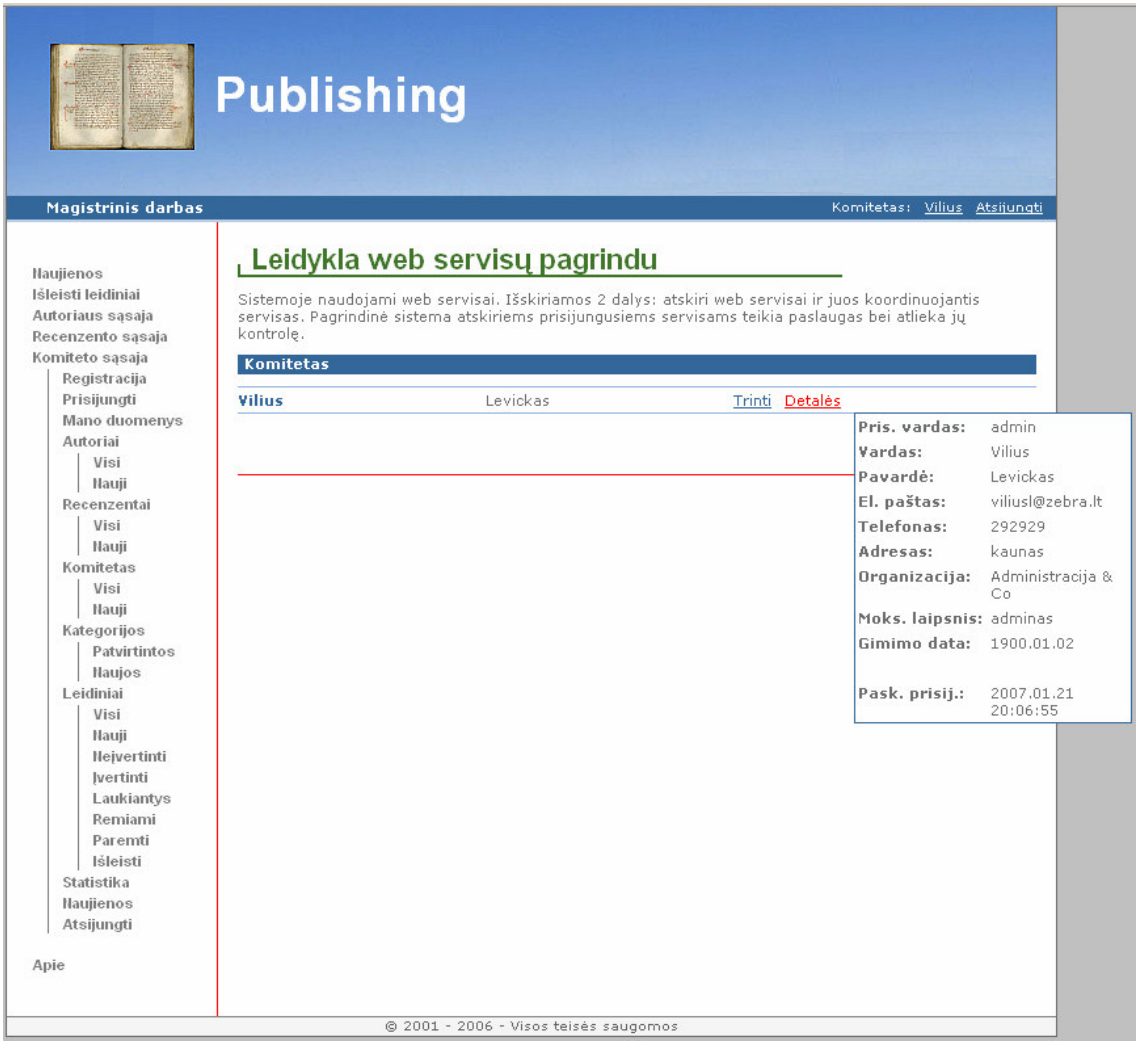
44 pav. Autoriai

2. Peržiūrėti sistemoje registruotus recenzentus (komitetas gali peržiūrėti sistemoje registruotus recenzentus ir juos valdyti (registruoti, redaguoti, šalinti). Pasiekimas – „Recenzentai“.



45 pav. Recenzentai

3. Peržiūrėti komiteto narius (komitetas gali peržiūrėti sistemoje registruotus komiteto narius ir juos valdyti (registruoti, redaguoti, šalinti). Pasiekimas – „Komitetas“.

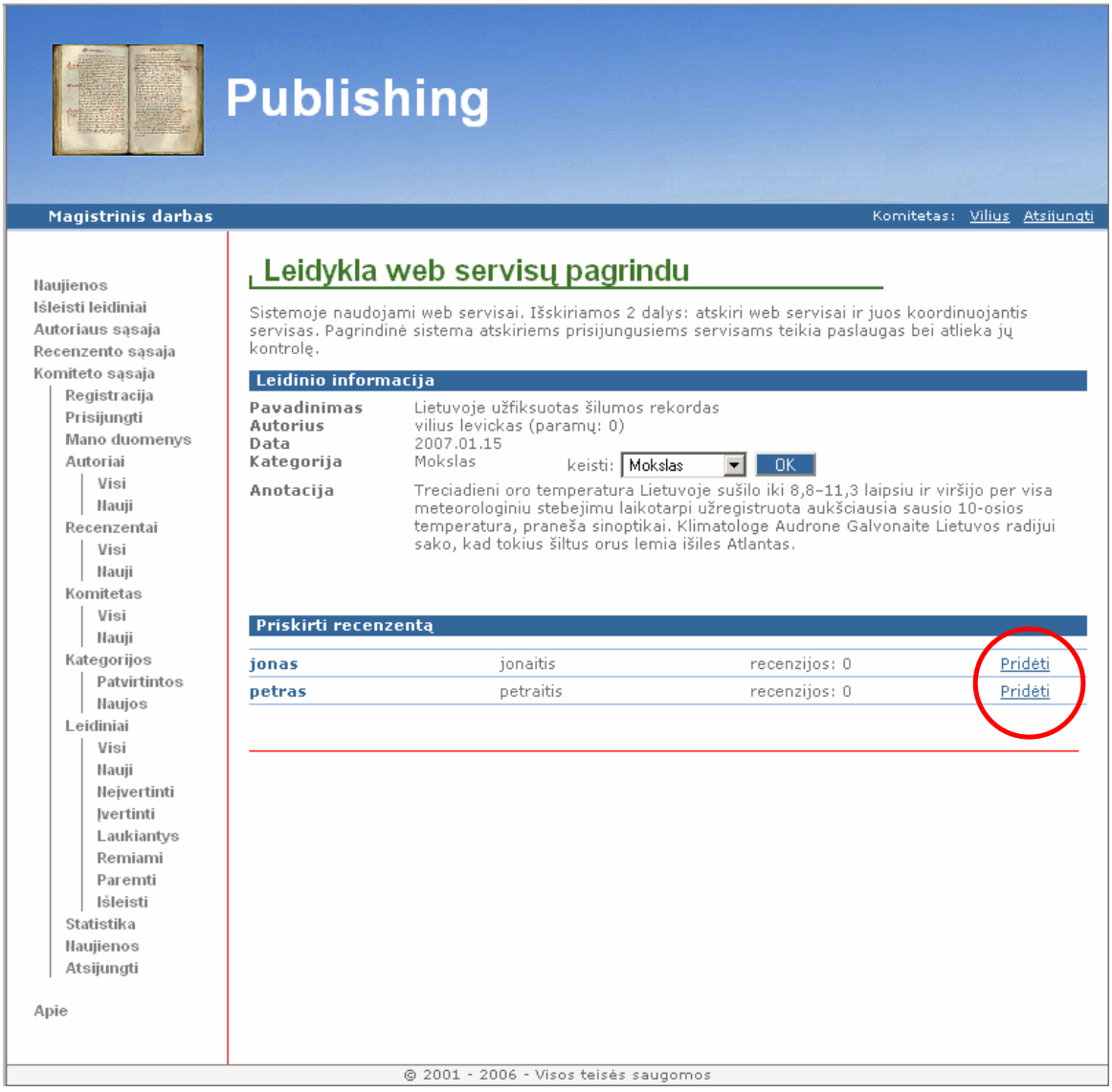


46 pav. Komiteto nariai

4. Peržiūrėti sistemoje registruotus leidinius (komitetas gali peržiūrėti sistemoje registruotus leidinius, jų autorius ir juos valdyti (registruoti, redaguoti, šalinti). Pasiekimas – „Leidiniai“.

47 pav. Leidinių peržiūra

5. Skirti recenzentus (komitetas kiekvienam autoriaus darbui skiria po vieną ar kelis recenzentus). Pasiekimas – „Leidiniai“ -> pasirinkti leidini -> „pridėti“



48 pav. Recenzentų skyrimas

6. Suteikti paramą (komitetas gali suteikti finansinę paramą kūrinio išleidimui, jei jos prašo autorius). Pasiekimas – „Leidiniai“-> „Laukiantys“ -> pasirinkti leidinį -> pateikti atsakymą.

49 pav. Paramos teikimas

7. Valdyti kategorijas (peržiūrėti, patvirtinti autorių pasiūlytas, kurti naujas). Pasiekimas – „Kategorijos“.

50 pav. Kategorijų valdymas

8. Peržiūrėti sistemos statistiką (komitetas gali peržiūrėti bendrą sistemos statistiką). Pasiekimas – „Statistika“.

51 pav. Statistika

Testavimas

Sistema buvo testuojama išsamiai ir keliais lygiais. Iš pradžių testuota vienetų (modulių) lygyje (buvo naudojamas „baltos dėžės“ testavimo modelis, kuriuo atskirai testuotos kiekvieno modulio funkcijos ir procedūros, iš jų sudaryti objektai).

Vėliau sistema buvo testuojama integracijos lygyje. Šio testavimo metu buvo tikrinamas bendradarbiavimas tarp sistemos komponentų (modulių), kurių kiekvienas atskirai jau buvo patikrintas vienetų lygyje. Integracijos lygio testavimas leidžia patikrinti sąveiką tarp visų sistemos modulių bei užtikrinti jų darnų veikimą tarpusavyje.

Atlikus integracinį testavimą, programinė įranga testuota sistemos lygyje. Šio testavimo tikslas – patikrinti bei užtikrinti pilną visos sistemos veikimą bei surasti galimas klaidas. Sistemos testavimo lygmenyje buvo tikrinama programa kaip visuma, jos sąsaja su galimais išoriniais įrenginiais, su jais susijusiomis programomis bei galimomis pagalbinėmis priemonėmis.

Dar vienas programinės įrangos testavimo etapas – patikrinimas, ar ji atitinka reikalavimų specifikaciją. Šio etapo metu buvo testuojama, ar programa atlieka tai, kam ji ir buvo kuriama, t.y. ar ji užtikrina vykdymą funkcijų, kurių iš jos ir buvo reikalaujama.

Taigi, sistema buvo ištestuota naudojantis šiomis strategijomis:

- Vienetų testavimo
- Integracinio testavimo
- Priėmimo testavimo
- Aukšto lygio testavimo

6. Koordinatoriaus šablono įvertinimas

Pasirenkant koordinatoriaus šabloną paslaugų sistemos kūrimui, buvo iškelta hipotezė, kad mažų ir vidutinių sistemų kūrimui labiau tinka koordinatoriaus šablonas nei chaotiškas kūrimo metodas ar kūrimas naudojant BPEL technologiją. Jai patikrinti buvo atlikta apklausa. Trijų programų inžinerijos studentų buvo prašoma susipažinti su šiame darbe taikytu koordinatoriaus modeliu, BPEL technologija ir išanalizuoti naujos paslaugos diegimą trim atvejais:

1. Chaotiškai projektuojant sistemą;
    - kuriama nesilaikant jokios metodologijos, griežtos struktūros,
    - Tokios sistemos procesai neapibrėžti, nėra valdymo kontrolės,
    - sistemos pasisekimas iš esmės priklauso nuo atskirų darbuotojų patirties,
    - Konkretus chaotiškos sistemos procesas gali būti toks sudėtingas, kad niekas nežino kaip ir kodėl jis veikia,
    - Procesai nėra dokumentuojami, taigi yra sunkiai ar visai nesuprantami.
2. Projektuojant pagal BPEL technologiją;
3. Naudojant koordinatorių.

Be to, kiekvieną atvejį išnagrinėti ir įvertinti kai kuriama:

1. Maža sistema;
2. Vidutinio dydžio sistema;
3. Didelė sistema.

Vertinimai buvo atliekami vertinimo skalėje nuo 1 iki 10, kur 10 reiškia aukščiausią įvertinimą, 0 – žemiausią.

Vertintojai:

1. Almanė Grinkevičiūtė
2. Ona Kostinaitė
3. Mindaugas Zlatarinskas

Faktoriai:

1. Suprantamumas
2. Sudėtingumas
3. Sistemos įdiegimas
4. Naujo modulio įtraukimas
5. Patikimumas
6. Testuojamumas
7. Lankstumas

Rezultatai:

Almanė Grinkevičiūtė:

2. lent. Almanės Grinkevičiūtės vertinimai

Nr.	**Faktorius**	**Vertinimai ***								
		Chaotiška sistema			BPEL			Mūsų sistema		
		V_m	V_v	V_d	V_m	V_v	V_d	V_m	V_v	V_d
1	Suprantamumas	6	5	3	4	8	10	9	8	7
2	Sudėtingumas	8	5	3	4	5	8	9	8	7
3	Sistemos įdiegimas	7	6	5	4	4	4	7	7	7
4	Naujo modulio įtraukimas	7	5	1	9	9	10	7	7	8
5	Patikimumas	5	5	1	9	9	9	8	8	8
6	Testuojamumas	5	4	1	8	8	9	9	9	8
7	Lankstumas	6	5	4	8	9	9	9	9	9

* kur:
V_m - vertinimas mažos sistemos atveju,
V_v - vertinimas vidutinės sistemos atveju,
V_d - vertinimas didelės sistemos atveju.

Ona Kostinaitė:

3. lent. Onos Kostaitės vertinimai

Nr.	**Faktorius**	**Vertinimai**								
		Chaotiška sistema			BPEL			Mūsų sistema		
		V_m	V_v	V_d	V_m	V_v	V_d	V_m	V_v	V_d
1	Suprantamumas	6	4	4	5	9	9	8	8	8
2	Sudėtingumas	6	4	4	5	8	9	8	8	8

3	Sistemos įdiegimas	8	7	6	5	5	5	7	7	7
4	Naujo modulio įtraukimas	5	5	3	9	9	9	8	8	8
5	Patikimumas	6	4	2	9	9	9	8	8	7
6	Testuojamumas	4	3	2	9	9	9	9	9	9
7	Lankstumas	6	4	4	9	9	9	9	8	8

Mindaugas Zlatarinskas

4. lent. Mindaugo Zlatarinsko vertinimai

Nr.	**Faktorius**	**Vertinimai**								
		Chaotiška sistema			BPEL			Mūsų sistema		
		V_m	V_v	V_d	V_m	V_v	V_d	V_m	V_v	V_d
1	Suprantamumas	6	4	3	3	5	10	8	8	7
2	Sudėtingumas	7	4	3	3	5	10	8	8	7
3	Sistemos įdiegimas	7	6	5	5	5	5	6	7	7
4	Naujo modulio įtraukimas	5	5	5	10	10	10	8	8	8
5	Patikimumas	6	5	3	9	9	9	8	8	8
6	Testuojamumas	4	4	2	9	9	9	9	9	9
7	Lankstumas	5	5	3	8	9	9	8	8	7

Faktorių įvertinimas apskaičiavus vidurkius

5. lent. Sistemų vertinimai, apskaičiavus vidurkius

Nr.	**Faktorius**	**Vertinimai**								
		Chaotiška sistema			BPEL			Mūsų sistema		
		V_m	V_v	V_d	V_m	V_v	V_d	V_m	V_v	V_d
1	Suprantamumas	6,0	4,3	3,3	4,0	7,3	9,7	8,3	8,0	7,3
2	Sudėtingumas	7,0	4,3	3,3	4,0	6,0	9,0	8,3	8,0	7,3
3	Sistemos įdiegimas	7,3	6,3	5,3	4,7	4,7	4,7	6,7	7,0	7,0
4	Naujo modulio įtraukimas	5,7	5,0	3,0	9,3	9,3	9,7	7,7	7,7	8,0
5	Patikimumas	5,7	4,7	2,0	9,0	9,0	9,0	8,0	8,0	7,7
6	Testuojamumas	4,3	3,7	1,7	8,7	8,7	9,0	9,0	9,0	8,7

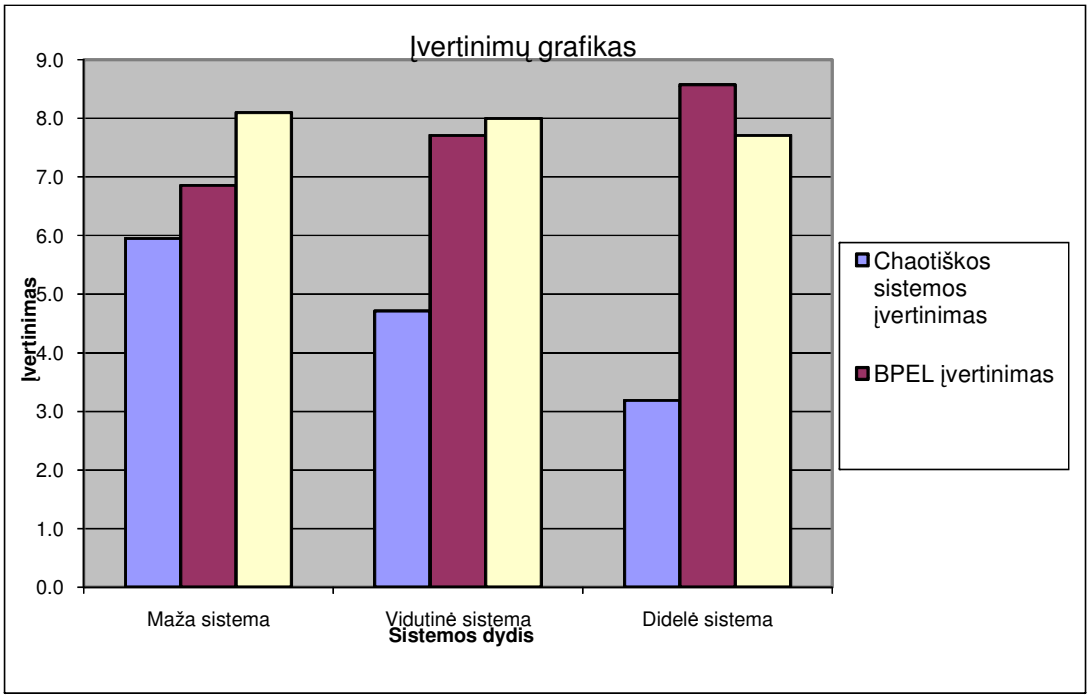
7	Lankstumas	5,7	4,7	3,7	8,3	9,0	9,0	8,7	8,3	8,0

Galutiniai įvertinimai:

Sistemos įvertinimai apskaičiuojami imant kiekvieno faktoriaus įvertinimo vidurkį.

6. lent. Galutiniai įvertinimai

	Chaotiškos sistemos įvertinimas	BPEL įvertinimas	Mūsų koordinatoriaus modelio įvertinimas
Maža sistema	6.0	6.9	8.1
Vidutinė sistema	4.7	7.7	8.0
Didelė sistema	3.2	8.6	7.7



52 pav. Įvertinimų grafikas

Kūrimo, naudojant koordinatoriu, palyginimas su chaotišku procesu bei BPEL

Realizuoti chaotišką sistemą yra lengviau, nei koordinatoriaus modelį. Tačiau aplinka kinta, reikalavimai taip pat, todėl dažnai tenka koreguoti sistemą ar papildyti naujomis funkcijomis, moduliais.
Koreguoti ar pridėti naujus modelius, o taip pat ir testuoti koordinatoriaus modelį(taip pat ir BPEL) yra žymiai lengviau ir greičiau nei chaotišką.

Sistemų palyginimo apibendrinta lentelė:

7. lent. Sistemų apibendrinimas

Savybė	**Chaotiška sistema**	**BPEL**	**Koordinatoriaus modelis**
Tinkamumas mažoms sistemoms	+	-	+
Tinkamumas vidutinėms sistemoms	?	+	+
Tinkamumas didelėms sistemoms	-	+	+
Dokumentacija	-	+	+
Nepriklausymas nuo platformos	?	+	-
Perprantamumas, sudėtingumas	sudėtingas	sudėtingas	lengvas
Instaliavimo sudėtingumas	lengvas	sudėtingas	lengvas
Naujo komponento pridėjimas	sudėtingas	lengvas	vidutinis
Pakartotinis panaudojamumas	-	+	+

Vertinimo išvados:

Iškelta hipotezė, kad mažų ir vidutinių sistemų kūrimui labiau tinka koordinatoriaus šablonas nei chaotiškas kūrimo metodas ar kūrimas naudojant BPEL technologiją, buvo vertinama trijų ekspertų, taikant minėtus metodus didelėms, mažoms ir vidutinio dydžio paslaugų sistemoms. BPEL gerai tinka tik didelėms sistemoms, nes reikia parengti daug dokumentacijos, tačiau tai yra patvirtintas standartas, tinka bet kokiems paslaugų jungimo atvejams, lengva įdiegti naują modulį. Mažoms ir vidutinėms sistemoms labiau tinka koordinatoriaus modelis, nes BPEL naudojimo atveju reikia įsisavinti BPEL kalbą bei įdiegti BPEL variklį, o tai sudėtinga ir brangu, reikalauja didelės darbuotojų kvalifikacijos.

Taigi remiantis apklausos rezultatais ir atlikta analize, galima teigti, kad hipotezė pasitvirtino: mažų ir vidutinių sistemų kūrimui labiau tinka koordinatoriaus šablonas nei chaotiškas kūrimo metodas ar kūrimas naudojant BPEL technologiją.

7. Išvados

1. IS kūrimo praktikos analizė rodo, kad paslaugų sistemos dažnai kuriamos chaotiškai, todėl jos būna trumpalaikės, sunkiai modifikuojamos
2. Paslaugų kūrimo metodikai pagerinti pasirinktas koordinatoriaus šablonas, kadangi praktikoje paplitusi BPEL reikalauja daug įdiegimo ir įsisavinimo pastangų
3. Web servisais paremto koordinatoriaus modelio įgyvendinimas leidybos agentūrų sistemai ir jo tyrimas parodė, kad koordinatoriaus modelį verta taikyti nedidelių ir vidutinių paslaugų sistemų kūrimui
4. Koordinatoriaus modelis užtikrina griežtą sistemos struktūrą, patikimumą, testuojamumą, valdymo galimybes, lengviau pakeisti jau esamas ar pridėti naujas sistemos funkcijas.
5. Atliktas koordinatoriaus modelio taikymas ir vertinimas patvirtino hipotezę, kad mažų ir vidutinių sistemų kūrimui labiau tinka šiame darbe nagrinėtas koordinatoriaus šablonas nei chaotiškas kūrimo metodas ar kūrimas naudojant BPEL technologiją
6. Sukurta koordinatoriaus programinė įranga yra universalus produktas, pritaikomas daugeliui panašių sistemų.
7. Darbo tematika buvo pristatytas straipsnis „Koordinatoriaus modelis programinės įrangos paslaugų sistemos kūrimui“ 12-ojoj tarpuniversitetinėje doktorantų ir magistrantų konferencijoje „Informacinė visuomenė ir universitetinės studijos ivus‘07“[38].

Literatūra

1. Web Services Architecture. Prieiga per internetą: <http://www.w3.org/TR/ws-arch/>
2. XML Web Services Basics. Prieiga per internetą: <http://msdn2.microsoft.com/en-us/library/ms996507.aspx>
3. XML Web Services Basics. Prieiga per internetą: <http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=/rzakl/rzaklintroadvantages.htm>
4. XML Web Services Infrastructure. Prieiga per internetą: <http://msdn2.microsoft.com/en-us/library/sd5s0c6d(VS.71).aspx>
5. What are the advantages of XML. Prieiga per internetą: <http://www.ornl.gov/~rwd/VH/PRESENTATIONS/SOT2001/tsld010.htm>
6. Bilgin, Ahmet Soydan. **Semantic Web Services Query and Manipulation Language for Quality Attributes of Web Services.** 2003, Prieiga per internetą: <http://www.lib.ncsu.edu/theses/available/etd-08182003-234629/, žiūrėta 2005 11 05>
7. D. Buttler. **Building Blocks for Composable Web Services.** 2003, Prieiga per internetą: <http://etd.gatech.edu/theses/available/etd-04082004-180046/unrestricted/buttler%5Fdavid%5F200312%5Fphd.pdf, žiūrėta 2005 11 12>
8. Cooper, W. James. **Introduction to Design Patterns in C#.** Addison Wesley, 2002.
9. Day, J. Clark. **A Framework for Autonomic Web Service Selection.** 2005. Prieiga per internetą: <http://library.usask.ca/theses/available/etd-08222005-190246/, žiūrėta 2005 11 14>.
10. Ekelund, Marcus; Larsson, Andreas. **Web Services: A Comparative Study.** Växjö University, School of Mathematics and Systems Engineering, Prieiga per internetą: <http://www.diva-portal.org/vxu/undergraduate/abstract.xsql?dbid=23>, žiūrėta 2005 11 03).
11. Gamma – Helm – Johnson – Vlissides. **Design Patterns: Elements of Reusable Object Oriented Software.** 1994.
12. Hasan, Jeffrey. **Expert Service-Oriented Architecture in C#: Using the Web Services Enhancements 2.0.** Apress, 2004.
13. Lai, Ray. **J2EE Platform Web Services.** Prentice Hall PTR, 2003.
14. Laurent, St. Simon; Johnston, Joe; Dumbill, Edd. **Programming Web Services with XML-RPC.** O'Reilly Publishing, 2001.
15. Maximilien, Eugene Michael. **Toward Autonomic Web Services Trust and Selection.** 2004, 230 p. Prieiga per internetą:

<http://www.lib.ncsu.edu/theses/available/etd-09092004-134545/>. žiūrėta 2005 11 03).

16. Nagarajan, Karthik. **Integration of Business Events and Rules Management with the Web Services Model.** 2003, 74 p. Prieiga per internetą: <http://etd.fcla.edu/UF/UFE0000753/nagarajan_k.pdf, žiūrėta 2005 11 14>
17. Ouzzani, Mourad. **Efficient Delivery of Web Services.** 2004. žiūrėta 2005 11 12. Prieiga per internetą: <http://scholar.lib.vt.edu/theses/available/etd-04272004-124255/>
18. Newcomer, Eric; L. Greg. **Understanding SOA with Web Services.** Addison Wesley Professional, December 14, 2004, 0-321-18086-0, 480 p.
19. Seth, Hitesh. Microsoft® .NET Kick Start. Sams Publishing, 2003 [žiūrėta 2005-11-13].
20. Y. Shohoud. **Real World XML Web Services.** Pearson Education, Inc., 2004 [žiūrėta 2005-11-13].
21. J. Snell. **Programming Web Services with SOAP.** O'Reilly, 2001.
22. Web service. [Žiūrėta 2005 11 13], Prieiga per internetą: <http://en.wikipedia.org/wiki/Web_services>
23. [Ceponiene, 2005] Čeponienė L., Nemuraitė L., Ambrazevičius E. Using state coordinator pattern for transition from design independent to platform independent model //Informacinės technologijos ir valdymas. ISSN 1392-124X. Kaunas 2005, T. 34, nr. 3, p. 263-268.
24. E. Newcomer, G. Lomow. Understanding SOA with Web Services. Publisher : Addison Wesley Professional, Pub Date : December 14, 2004, 480 p.
25. Lietuvos leidėjų katalogas. [Žiūrėta 2007 04 16], Prieiga per internetą: <http://www.lnb.lt/leidejai/index1.html>
26. A Note on Distributed Computing, S. C. Kendall, J. Waldo, A. Wollrath, G. Wyant, November 1994, Prieiga per internetą: <prieiga internete http://research.sun.com/techrep/1994/abstract-29.html>
27. BEEP protokolas, Prieiga per internetą: <http://www.rfc-editor.org/rfc/rfc3080.txt>.
28. P. Denning. Annotated List of Web Services Specs. Prieiga per internetą: <http://lists.w3.org/Archives/Public/www-ws-arch/2004Feb/0022.html>
29. Web services protocol stack, Prieiga per internetą: <http://en.wikipedia.org/wiki/Web_Services_Protocol_Stack>
30. SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation 24 June 2003. Prieiga per internetą: <http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>
31. XML Binary Characterization, W3C Working Group Note 31 March 2005. Prieiga per internetą: <http://www.w3.org/TR/xbc-characterization/>

32. C. Schittko. Web Service Orchestration with BPEL. 2003. Prieiga per internetą: <http://www.idealliance.org/papers/dx_xml03/papers/04-06-01/04-06-01.html>

33. Wikipedia. Programming in the large. Prieiga per internetą: <http://en.wikipedia.org/wiki/Programming_in_the_large>

34. Open Source ESB (and ESB-like) announcements: ServiceMix and Apache Synapse. 2005. Prieiga per internetą: <http://elementallinks.typepad.com/bmichelson/2005/09/view_bpel_proce.html>

35. Guide to BPEL. http://www.radikalfx.com/bpel/

36. Tuomas Piispanen "BPEL & SOA Presentation", 2006.

37. Orchestration / Engines. http://www.radikalfx.com/bpel/orchestration.html.

38. V. Levickas, S. Grinkevičius. „Koordinatoriaus modelis programinės įrangos paslaugų sistemos kūrimui". 12-oji tarpuniversitetinėje doktorantų ir magistrantų konferencijoja „Informacinė visuomenė ir universitetinės studijos ivus'07". Kaunas, 2007.

Terminų ir santrumpų žodynas

API – Application programming interfaze

BPEL – Business Process Execution Language

COM – Component Object Model

CORBA – Common Object Request Broker Architecture

FTP – File Transfer Protocol

HTTP – Hypertext Transfer Protocol

RPC – Remote procedure call

SOA – Service-oriented architecture

SOAP – Simple Object Access Protocol

SSL – Secure Sockets Layer

TLS – Transport Layer Security

UDDI – Universal Description Discovery and Integration

UML – Unified Modeling Language

VPN – Virtual Private Network

W3C – World Wide Web Consortium

WSD – Web Service Description

WSDL – Web Services Description Language

XML – Extensible Markup Language

Priedai

Straipsnis konferencijai “Informacinė visuomenė ir universitetinės studijos” (IVUS‘07)

Koordinatoriaus modelis programinės įrangos paslaugų sistemos kūrimui

Simas Grinkevičius

Vilius Levickas

Kauno technologijos universitetas, Studentų g. 50, LT-3031 Kaunas

Straipsnyje pateikiamas koordinatoriaus modelis[3], skirtas interneto paslaugų sistemų kūrimui, ir šio modelio tinkamumo patvirtinimui sukurta interneto paslaugų sistema. Koordinatoriaus paslaugos kūrimas pareikalavo didesnių pastangų, tačiau koordinatorius palengvina naujų paslaugų įtraukimą ar esamų modifikavimą ir gali būti naudojamas, kuriant kitas sistemas. Remiantis įgyta patirtimi, sudaryta paslaugų informacinių sistemų kūrimo, naudojant paslaugų koordinatorių, metodika.

Įvadas

Šis projektas skirtas sukurti bei praktikoje įgyvendinti universalų koordinatoriaus modelį, skirtą programinės įrangos paslaugų sistemos kūrimui. Daugelio versle vykdomų veiklos procesų taisyklės nuolat kinta ir reikalauja dažnų programinės įrangos pakeitimų[4]. Pagrindinė problema iškyla tada, kai reikia restruktūrizuoti esamą programų sistema, pridėti naujas funkcijas ar modifikuoti jau esančias. Tokiu atveju paprastai reikia daug darbo sąnaudų. Įgyvendintas koordinatoriaus modelis gerokai palengvina tokių pakeitimų atlikimą. Be to, šis modelis gerokai palengvina bei supaprastina ir visiškai naujų programinės įrangos paslaugų sistemų kūrimą.

Darbo tikslas yra sukurti Web servisų koordinatoriaus modelį, ištirti jo veikimą ir pasiūlyti koordinatoriaus šabloną, kurį galėtų naudoti daugelis projektuotojų, siekiantys kurti informacines sistemas, komponuodami jas iš elementarių, būsenų nesaugančių Web servisų. Svarbu yra ne tik sukurti gerą programinį produktą, bet ir išbandyti Informacijos sistemų katedroje konsultanto pasiūlytą Web servisų projektavimo metodą – jį realizuoti ir atlikti šio modelio tyrimą. Sukurtas koordinatoriaus servisas turėtų būti universalus produktas,

[3] Sistema remiasi doktorantės Linos Čeponienės sudarytu metodu (veiklos paslaugų sistemos projektavimas, pradedant nuo reikalavimų apibrėžimo, kuris transformuojamas į projektą remiantis architektūriniu šablonu).

[4] [3], 215p.

pritaikomas daugeliui panašių sistemų. Koordinatoriaus šablonas ir sukurtas servisas galėtų būti alternatyva Web servisų varikliams (pavyzdžiui, BPEL[5] [4]), juos galėtų naudoti įvairių informacinių sistemų projektuotojai.

Taigi, šio projekto tyrimo sritis – Web servisais paremto koordinatoriaus modelio sudarymo ir jo įgyvendinimo programine įranga metodika. Tyrimo objektas – koordinatoriaus modelio pritaikymas leidybos agentūrų sistemai, kurioje vyksta iš daugelio etapų susidedantys ir didelių organizacinių pastangų reikalaujantys procesai bei galimas koordinatoriaus modelio pritaikymas naujų sistemų kūrime.

Sistemos kūrimo metu buvo išnagrinėta daug literatūros šaltinių bei išanalizuotas problemos sprendimas pasaulyje. Taip pat buvo įvertinta situacija Lietuvoje. Tuomet buvo pradėta projektuoti pati programų sistema. Buvo numatytos ir suprojektuotos programų sistemos funkcijos, vartotojo charakteristikos, aprašytos problemos, tikslai bei bendri apribojimai. Atlikti projekto įgyvendinimo planai ir kokybės vertinimo analizė. Buvo aprašyti reikalavimų specifikavimas, architektūros specifikacija bei detalios architektūros specifikacija.

Sistema buvo įgyvendinta pagal parengtus dokumentus ir specifikacijas, naudojant Microsoft .NET technologiją. Užbaigus pagrindinius programavimo darbus buvo atliktas visapusiškas sistemos testavimas panaudojant juodos ir baltos dėžės metodus. Taip pat buvo sukurta detali sistemos vartotojo dokumentacija bei atliktas sistemos kokybės vertinimas.

Buvo sukurta anksčiau minėtu metodu besiremianti programinė įranga – nedidelė eksperimentinė leidyklų agentūros sistema, kurioje naudojami Web servisai. Taigi, buvo sukurtas Web servisų sistemos modelis, ištirtas jo veikimas ir pasiūlytas šablonas, kurį galėtų naudoti daugelis projektuotojų, siekiantys kurti informacines sistemas, komponuodami jas iš elementarių, būsenų nesaugančių Web servisų. Sukurtas produktas buvo eksperimentiškai išbandytas realių vartotojų.

Sukurta sistema skirta naudojimui leidyklose bei leidybos agentūrose. Dėka sukurto koordinatoriaus modelio, sistema nesunkiai pritaikoma skirtingose darbo aplinkose, t.y. skirtingos leidybos agentūros bei leidyklos gali lengvai ją pritaikyti savo reikmėms. Negana to, sistema su nedideliais pakeitimais gali būti naudojama ir kitose srityse, kuriose reikalinga centrinė vartotojų pateikiamos informacijos koordinacija ir valdymas.

Tokia informacinė sistema turi nemažų perspektyvų konkuruoti tiek Lietuvoje, tiek ir pasaulyje, nes ypač Lietuvoje analogiškų sistemų nėra labai daug. Pagal Lietuvos leidėjų katalogą[6], vien Lietuvoje šiuo metu įregistruoti 1597 leidėjai. Daugiau nei 10 skirtingų pavadinimų knygų per metus išleidžia net 67 Lietuvos leidėjai. Be abejonės, užsienyje leidėjų yra gerokai daugiau. Taigi, kuriamam produktui egzistuoja plati rinka, tereikia sugebėti ją išnaudoti.

Plėtojant verslą, bus galima taikyti lanksčią kainodarą, informacinės sistemos kainą derinant su kliento dydžiu. Pavyzdžiui, kuo daugiau autorių ir recenzentų galės naudotis sistema, tuo ji bus brangesnė. Galimi ir kiti rinkodaros variantai – iš autorių, leidėjų ir

---

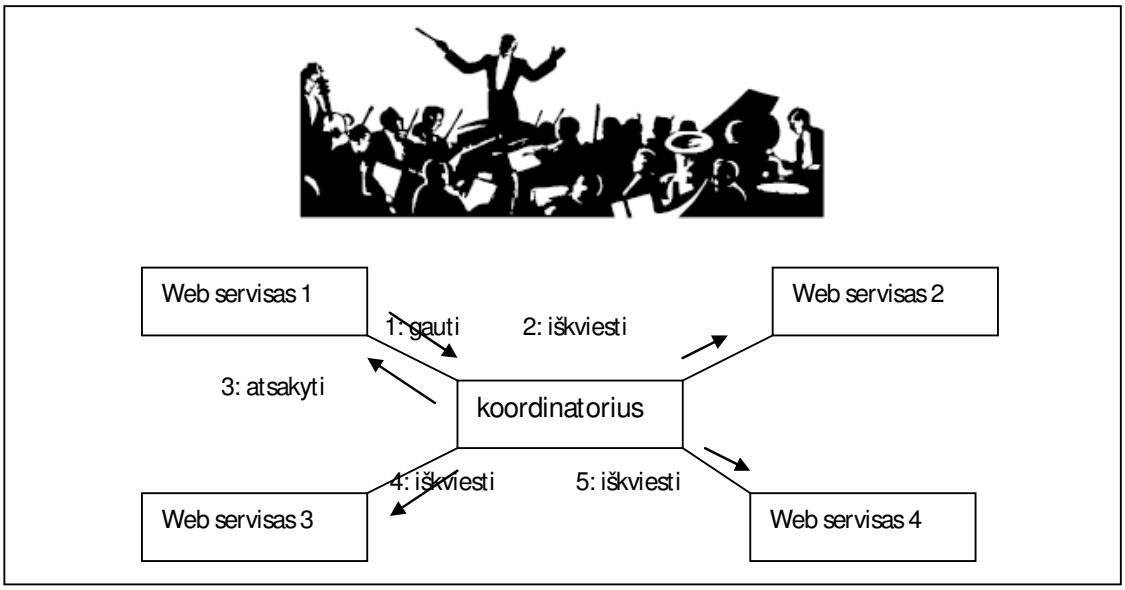
[5] Business Process Execution Language – verslo procesų vykdymo kalba, pagrįsta XML technologija ir sukurta tam, kad standartizuotų integracijos logiką ir procesų automatizavimą tarp Web servisų.

[6] [1], žiūrėta 2006 04 16

recenzentų galima imti mėnesinį registracijos mokestį už galimybę naudotis sistema, pačius web servisus leidžiant savo tarnybinėse stotyse. Taip pat galima parduoti sistemą leidybų agentūrai ir vėliau apmokestinti jos palaikymą bei priežiūrą. Žinoma, uždirbti galima ir toliau plėtojant sistemą bei parduodant jos naujas versijas. Be to, Lietuvoje ir kitose šalyse, atsižvelgiant į rinkos padėtį, būtų galima taikyti skirtingą kainodarą, atsižvelgiant į šalies specifiką, tačiau tai jo nėra šio darbo tikslas.

Koordinatoriaus modelis

Koordinatorius – tai pagrindinis servisas, koordinuojantis kitus, šalutinius, servisus. Jis atsakingas už užklausų priėmimą, *pre* ir *post* sąlygų patikrinimą, atitinkamos paslaugos iškvietimą, pranešimo siuntimą pradinės užklausos siuntėjui ir pan. Koordinatoriaus veikimas atrodo taip:



1 pav. Būsenų koordinatoriaus veikimas

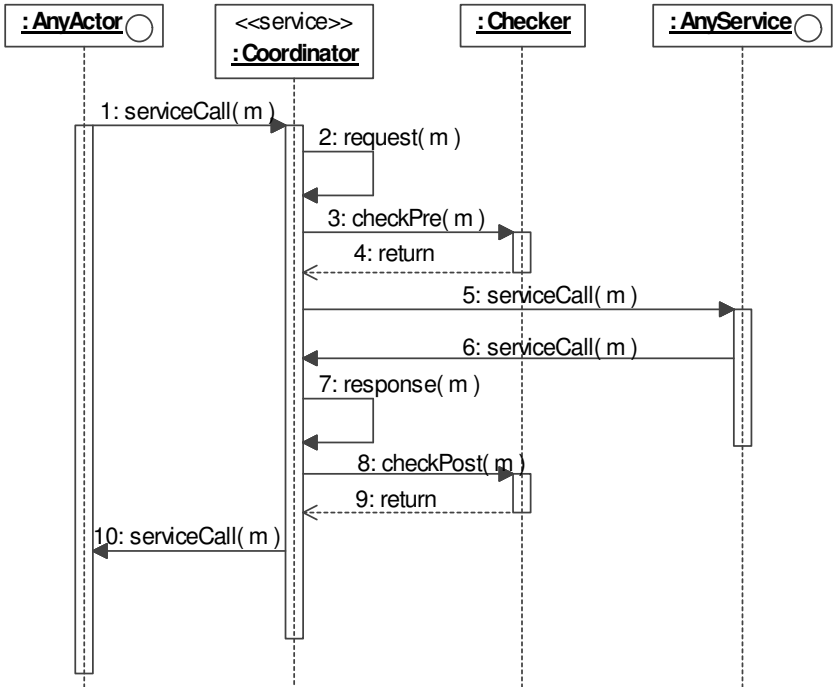
Koordinatorius gauna ir reaguoja į dviejų tipų pranešimus – užklausas (request) ir atsakus (response). Užklausos yra gaunamos iš aktorių ar paslaugų, kurie siekia iškviesti konkrečios paslaugos operaciją. Atsakas gaunamas iš paslaugos, kuri tą operaciją atliko. Kadangi koordinatorius yra paslauga, bendraujanti pranešimais paremtu bendravimo modeliu, į jį kreipiamasi iškviečiant operaciją serviceCall, kurios argumentas – pranešimas (MessageEntity), kur nurodyta pageidaujama iškviesti operacija, jos argumentai ir jų reikšmės, kas ją nori iškviesti ir t.t.

Gavęs užklausos pranešimą, koordinatorius pirmiausia iškviečia tikrintojo klasę (Checker), kad būtų patikrinta reikalaujamos operacijos prieš sąlyga. Operacijų prieš ir sąlygos saugomos apribojimų bazėje (arba realizuojamos kaip atskiros operacijos), apribojimai aprašomi naudojant dalykinės srities esybių atributus ir ryšius. Jei tikrintojo klasė praneša, kad operacijos prieš sąlyga negalioja, užklausos siuntėjui grąžinamas atmetimo

(exception) pranešimas. Jei sąlyga galioja, koordinatorius persiunčia užklausos pranešimą konkrečiai paslaugai.

Kai koordinatorius gauna atsakymą iš paslaugos apie operacijos įvykdymo rezultatus, jis vėl kviečia tikrinimo klasę, kuri iš apribojimų bazės (ar iškvietęs konkrečią tikrinimo operaciją) turi pateikti operacijos po sąlygoje aprašytus siunčiamus pranešimus. Tai dažniausiai pranešimo siuntimas vienam paslaugos interfeisui, bet gali būti ir pranešimų sekos siuntimo aprašas, ar reikalavimas siųsti keletą lygiagrečių pranešimų skirtingiems gavėjams ir pan. Visi šie pranešimų siuntimo būdai aprašomi panaudojant XML[7]. Jei reikia siųsti atsako ar priėmimo patvirtinimo (acceptance) pranešimą, koordinatorius persiunčia tokį pranešimą aktoriaus paslaugai. Jei reikia siųsti užklausos pranešimą, koordinatorius vėl elgiasi kaip gavęs užklausą operacijai atlikti.

Koordinatoriaus reakcijos į gaunamus pranešimus gali būti tokios:



2 pav. Koordinatoriaus reakcija į gaunamus pranešimus

Kaip pavyzdys buvo paimtas panaudojimo atvejis „Pateikti leidini“, bei, kad būtų aiškiau supaprastintas iki vieno žingsnio:

Panaudojimo atvejis Pateikti leidinį

1. Autorius nurodo sistemai pateikti Leidinį svarstymui

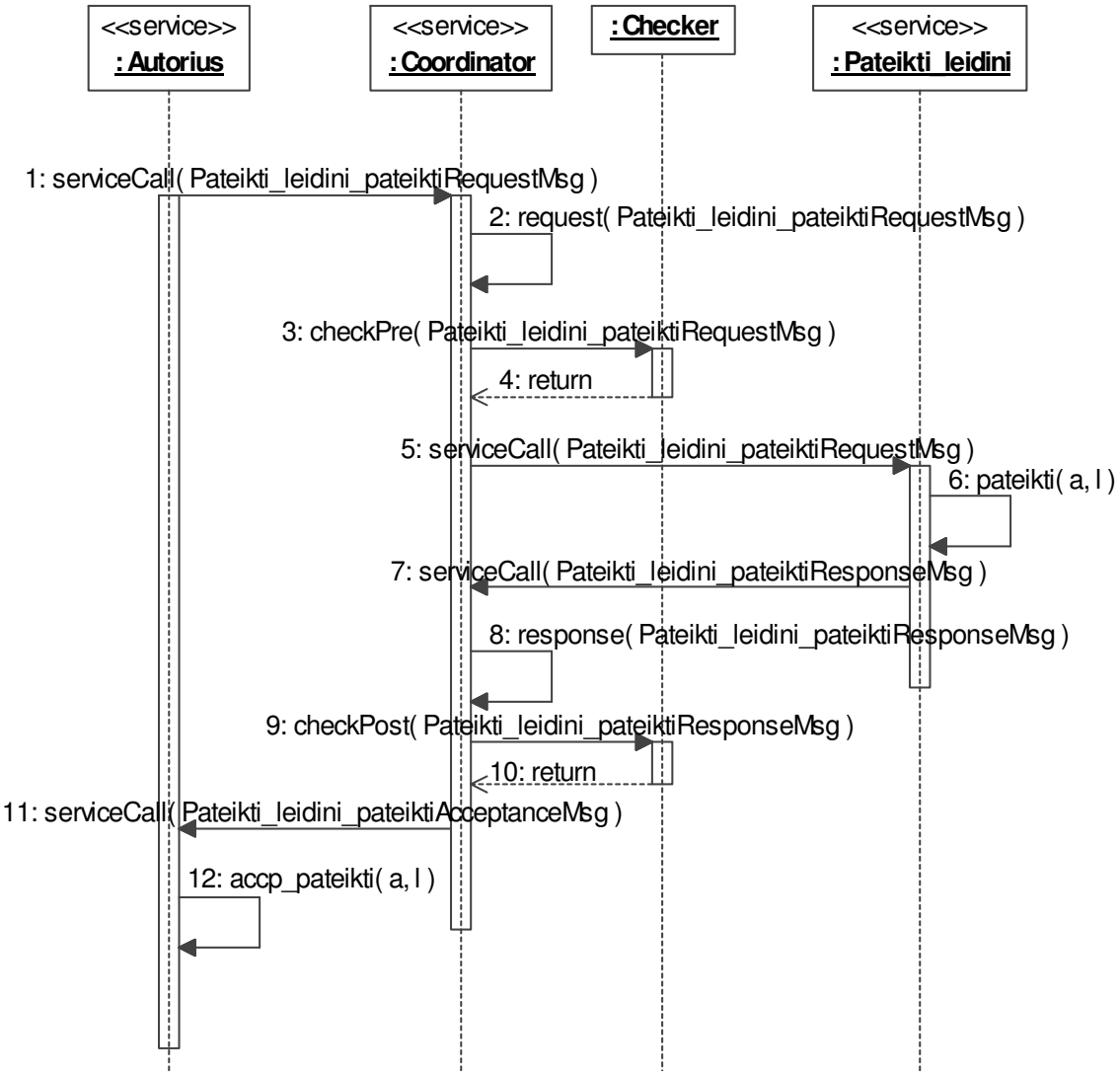
   Įvykio Prieš sąlyga: Pateikiamas Leidinys yra būsenoje „Įkeltas“.

   Įvykio Sužadinimo sąlyga: Autorius nori pateikti Leidinį svarstymui

   Įvykio Po sąlyga: Leidinys yra būsenoje „pateiktas“. Autoriui nusiųstas Leidinio pateikimo patvirtinimas.

[7] [2], žiūrėta 2007 04 16

Ši sekos diagrama su sąsajomis, pritaikius koordinatoriaus šabloną turėtų atrodyti taip:



3 pav. Koordinatoriaus veikimas naudojant sąsajas

Išvados

Taigi, koordinatoriaus modelis užtikrina griežtą sistemos struktūrą bei jos patikimumą, testuojamumą bei valdymo galimybes. Be to, naudojant koordinatorių kur kas lengviau pakeisti jau esamas ar pridėti naujas sistemos funkcijas ar net sukurti visiškai naują programinės įrangos paslaugų sistemą. Todėl galima teigti, jog sukurtas koordinatoriaus servisas yra universalus produktas, pritaikomas daugeliui panašių sistemų.

Aptarta Web servisais paremto koordinatoriaus modelio sudarymo ir jo įgyvendinimo programine įranga metodika leido pritaikyti ją leidybos agentūrų sistemos, kurioje vyksta iš daugelio etapų susidedantys ir didelių organizacinių pastangų reikalaujantys procesai, kūrimui. Ši sistema buvo sukurta ir sėkmingai išbandyta realioje aplinkoje.

Literatūros sąrašas

[1] Lietuvos leidėjų katalogas. [Žiūrėta 2007 04 16], prieiga internete <http://www.lnb.lt/leidejai/index1.html>

[2] **S. Yasser.** Real World XML Web Services. Pearson Education, Inc., 2004.

[3] **E. Newcomer, G. Lomow.** Understanding SOA with Web Services. Publisher : Addison Wesley Professional, Pub Date : December 14, 2004, 480 p.

[4] The BPEL language. [Žiūrėta 2007 04 20], prieiga internete <http://www.radikalfx.com/bpel/language.html>

[5] **S. Laurent, J. Johnston, E. Dumbill.** Programming Web Services with XML-RPC. O'Reilly Publishing, 2001, 230 p.

Coordinator model for software services system creation

In this article the coordinator model, designed for Internet services systems creation is presented. Also in this article is presented Internet services system, which is designed to prove the appropriateness of the aforementioned coordinator model. The creation of coordinator service required some effort, but this service greatly relieves the struggle of including new services or modification of existing and can be used when creating new systems. Based on the acquired experience, the technique of creating information systems services using service coordinator was created.