KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Žygimantas Šilanskas

Servisų bendravimas pagrįstas trečių šalių komponentais

Magistro darbas

Darbo vadovas
doc. dr. T. Blažauskas
2010-05-27

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Žygimantas Šilanskas

Servisų bendravimas pagrįstas trečių šalių komponentais

Magistro darbas

Recenzentas
doc. Vytautas Pilkauskas
2010-05

Vadovas
doc. dr. T. Blažauskas
2010-05-27

Atliko
IFM-4/2 gr. stud.
Žygimantas Šilanskas
2010-05-27

Kaunas, 2010

TURINYS

1 ĮVADAS

1.1 Dokumento paskirtis

Šio dokumento paskirtis yra aprašyti paskirstytų sistemų privalumus ir trūkumus, pateikti ir aprašyti trečių šalių komponentus, skirtus bendravimui tarp paskirstytų sistemų dalių. Pasirinkus komponentus, kurie labiausiai tinka magistrinio darbo metu sukurtai sistemai, pateikti šių komponentų tyrimą. Pateikti bei įvertinti rezultatus; pateikti rekomendacijas ir išvadas apie komponentų patogumą naudotis vartotojui.

Šis dokumentas gali būti naudojamas suteikiant daugiau informacijos apie paskirstytas sistemas bei kaip pagrindas, pasirenkant komponentus, skirtus bendravimui tarp paskirstytų sistemų dalių. Tačiau toks pasirinkimas pasiteisins, kada nagrinėjama sistema sutampa su magistrinio darbo metu sukurta arba jų veiklos kontekstai yra panašūs.

Dokumentas gali būti skirtas paskirstytų sistemų vartotojams, kūrėjams, analitikams, projektuotojams, testuotojams ir t.t.

1.2 Santrauka

Šiuolaikinio verslo sektorius yra labai sudėtingas, kadangi daugybė programinės ir kompiuterinės įrangos yra naudojama skirtingose platformose: internetinių sistemų bendravimui, verslo sistemų integravimui ir t. t. Dauguma šiuolaikinės programinės įrangos priklauso paskirstytų sistemų tipui. Ją sudaro kliento-serverio architektūra. Nagrinėjant konkrečią paskirstytą sistemą, joje klientų ir serverių kiekis gali būti gausus. Tokio tipo sistemos turi labai daug privalumų, tačiau turi ir trūkumų. Paprastai paskirstytas sistemas sudaro ne viena duomenų bazė, veikia daug servisų, kurie ne visada gali būti pasiekiami, visa tai yra galimų klaidų šaltinis. Kliento programinė įranga bendrauja tarp įvairių programinės įrangos komponentų (pavyzdžiui servisų), kaip ir jie tarpusavyje, naudodama nuotolinių procedūrų kvietimo technologijas ir žinutes.

Naudojant žinučių technologiją, bendravimas tarp servisų gali būti daug patikimesnis, kadangi ši technologija užtikrina žinutės pristatymą, net ir tada, kai gavėjas yra už ryšio ribų. Šiuolaikinėse servisais paremtose sistemose, bendravimas tarp komponentų yra paremtas žinučių, naudojančių eiles, metodu.

Šiame darbe bus nagrinėjama bendravimo tarp komponentų technologija, kuri paremta žinutėmis. Taip pat bus apžvelgiamos ir nagrinėjamos trečių šalių sukurtos,

nemokamos technologijos, kurios skirtos bendravimui tarp paskirstytų sistemų komponentų žinučių metodu. Kadangi magistrinio darbo sistema buvo sukurta naudojant Microsoft technologijas, bus nagrinėjami tik tie komponentai, kurie naudos Microsoft firmos sukurta žinučių bendravimo protokolą (MSMQ). Darbe bus pasirinktos bendravimo žinutėmis technologijos, kurios labiausiai tinka magistrinio darbo sistemos kontekste. Bus atliekamas šių technologijų tyrimas pagal tam tikrus kriterijus, kurie suformuluoti iš problemų, kilusių naudojant magistrinio darbo sistemą. Kriterijai apims ir siuntimo metodus, ir komponentų panaudojimo paprastumą, dokumentacijos pasiūlą. Galiausiai bus suformuotas tyrimo modelis, kuris įgyvendins pateiktus tyrimo metodus. Naudojant šį modelį bus atliekamas ir pats tyrimas, ir surinkti tyrimui skirti duomenys. Pagal gautus rezultatus bus pateiktos komponentų pasirinkimo rekomendacijos vartotojui.

2 BENDRAVIMAS TARP PASKIRSTYTOS SISTEMOS DALIŲ

2.1 Paskirstytos sistemos

Paskirstytą sistemą sudaro daug nepriklausomų kompiuterių, kurie bendrauja tarpusavyje naudodami kompiuterių tinklą [1]. Taip pat šią sistemą sudaro ir programinė įranga, kuri veikia per daugelį kompiuterių ir padeda vartotojui pasiekti norimą tikslą [1].

Paskirstytos sistemos - tai šiuolaikiškos sistemos. Verslo programinė įranga kuriama naudojant tokio tipo sistemas. Manoma, jog ateities operacinės sistemos taip pat turės paskirstytų sistemų bruožų [1]. Kadangi tokios sistemos komponentai išsibarstę po daugelį kompiuterių, iš tokio tipo sistemų reikalaujama, kad vartotojai nejaustų jog dirba tinkle. Pagrindiniai tokių sistemų privalumai [2]:

- Centralizuota architektūra, lengvesni komponentų pakeitimai, atnaujinimai.
- Palaiko daug klientų, kurie bendrauja per centrinius servisus.
- Užtikrina lygiagretų bendravimą tarp serverio dalies ir klientų.
- Didelė panaudojimo vertė – tam tikrus komponentus galima panaudoti ir kitose projektuose (pvz. servisus).
- Didesnis patikimumas.

Trūkumai [2]:

- Tokias sistemas sunku kurti, įdiegti ir palaikyti.
- Didelė priklausomybė nuo tinklo patikimumo ir ryšio kokybės.
- Daugiau komponentų, sudėtingesnė sistema, todėl didesnė tikimybė atsirasti defektams.

2.2 Windows operacinės sistemos komunikacijų pamatas (WCF)

Šis pamatas yra Microsoft .NET Framework dalis.

2.2.1 Microsoft .NET Framework ir WCF

.NET Framework yra viena pažangiausių Microsoft technologijų skirtų šiuolaikinės programinės įrangos, servisų bei komunikacijos priemonių kūrimui. .NET nuo pat pradžių buvo vystoma kaip pažangi programavimo technologija, orientuota į šiuolaikinio verslo poreikius ir tų poreikių greitą augimą, pokyčius. Ši technologija yra sukurta kaip labai lanksti strategija, apimanti įvairius ir plačius poreikius, kurie gali keistis, augti ir vystytis [3]. Ši technologija yra nepriklausoma nuo programavimo kalbos, todėl ja paremtus sprendimus, atsižvelgiant į jų specifiką, galima kurti naudojant įvairias programavimo kalbas: ASP.NET, C#, C++, VB. Microsoft .NET užtikrina sprendimų, sukurtų skirtingomis programinėmis priemonėmis .NET terpėje, suderinamumą tarpusavyje ir tai padeda sutaupyti laiko ir lėšų atnaujinant informacines sistemas bei suderinant jas su senesnėmis sistemomis. Pagrindiniai Microsoft .NET technologijų privalumai [3]:

- Pilna, greita ir patogi integracija su visais Microsoft produktais, taip pat geras suderinamumas tarp Microsoft produktų.
- Saugumas. Šios technologijos saugumo politika paremti sprendimai yra gerokai saugesni vien dėl to, kad jie naudoja bendrą integruotą saugumo užtikrinimą.
- .NET technologija yra ir bus vystoma bei palaikoma Microsoft kompanijos, todėl remiantis ja bus galima kurti naujus sprendimus integruojant juos į veikiančias, seniau sukurtas sistemas.
- Naujų sistemų kūrimo paprastumas. To paties funkcionalumo sprendimui realizuoti .NET (ASP.NET, C#) reikia mažiau kodo lyginant su PHP ar JAVA. Žymiai mažiau kodo – atitinkamai mažesnė tikimybė įsivelti programinėms klaidoms, o tai tiesiogiai susiję su programinio sprendimo veikimo patikimumu.

- Microsoft .NET technologijose realizuota daug sprendimų, kurie susiję su internetinėmis technologijomis (ASP.NET), saugumu (assemblies, CAS (Code Access Security)).
- Sukurta bendra .NET klasių biblioteka, kurioje realizuota daug pagrindinių funkcijų, tiesioginis darbas su duomenų bazės objektais (ADO.NET), komponentinis programavimas (COM/DCOM) ir kita.

Windows operacinės sistemos komunikacijų pamatas (WCF) tai - Microsoft firmos programų kūrimo sąsaja (API), naudojanti nuotolinių procedūrų kvietimo technologiją. Ši sąsaja yra naudojama kurti paskirstytas sistemas. WCF yra .NET Framework dalis, tikslas - visus komunikacinius modelius, pristatytus ankstesnėse .NET Framework versijose, sudėti į vieną grupę, kuri vadinama Servisais paremtu kūrimu [4]. Ši technologija, naudojant tam tikrus formatus, įgalina bendravimą netik tarp to paties tipo procesų, bet ir tarp procesų, kurie kurti naudojant kitas technologijas (kitas nei .NET Framework) [4]. Taigi kuriami komponentai gali bendrauti su kitokio tipo sistemomis.

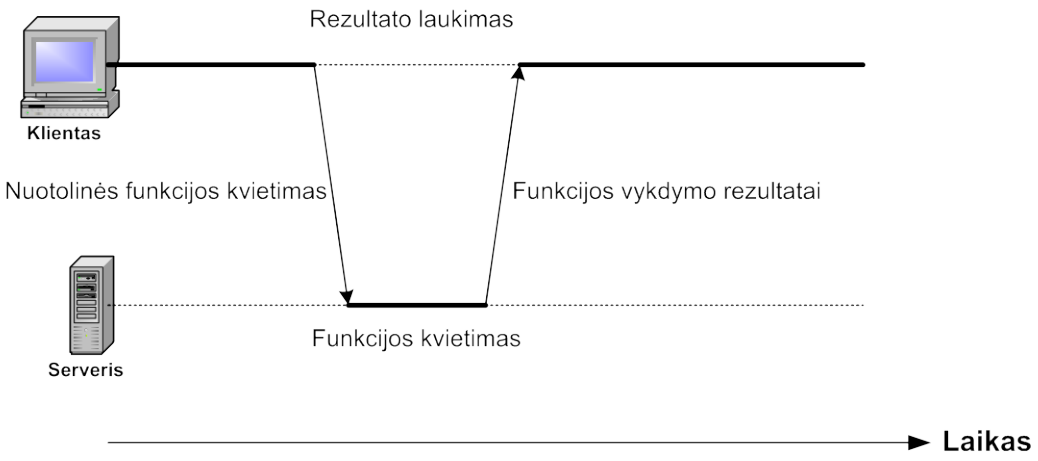
Naudojantis šiomis technologijomis, paskirstytos sistemos yra greičiau kuriamos ir atitinka visas servisais paremtos architektūros savybes [4].

2.3 Nuotolinių procedūrų kvietimo technologija

Kadangi nagrinėsime WCF pagrindu kurtas sistemas, tai išskirsime tokius bendravimo tarp paskirstytos sistemos komponentų tipus:

- Nuotolinių procedūrų kvietimo technologijos.
- Žinučių, naudojančių eiles, technologijos (bus aptarta skyriuje 2.7 Žinučių siuntimo technologija naudojanti eilių duomenų struktūrą).

Nuotolinių procedūrų kvietimo technologija – tai aukšto lygio komunikavimo tarp paskirstytos sistemos komponentų modelis [5]. Jis leidžia kurti programinę įrangą, kuri yra skirta darbui tinkle [5]. Ši technologija remiasi kliento serverio architektūra ir įgalina bendravimą tarp procesų, kurie yra skirtinguose kompiuteriuose [5].



1 pav. Tipinis nuotolinės funkcijos kvietimas ir vykdymas.

1 pav. matome, jog sistemą sudaro klientas, serveris ir laiko skalė. Klientas sinchroniniu metodu kviečia serverio procedūrą ir laukia kol serveris grąžins rezultatą. Nuotolinės funkcijos kvietimas vyksta tokiu būdu:

1) Klientas siunčia užklausą serveriui. Užklausoje yra informacija apie nuotolinę procedūrą ir jai skirti parametrai.
2) Serveris suranda reikalingą procedūrą ir ją įvykdo.
3) Įvykdęs procedūrą rezultatus grąžina klientui (per tinklo terpę).

Šie veiksmai paprastai vyksta kompiuterių tinkle, todėl ypatingai svarbus parametras yra tinklo patikimumas. Naudojant žinučių technologijas šio parametro vertę galima sumažinti.

2.4 Servisais (paslaugomis) besiremianti architektūra (SOA)

Tai paskirstytos programinės įrangos architektūros modelis, kuris gali padėti įmonei lengviau prisitaikyti prie nuolat kintančių verslo sąlygų. Ši architektūra įgalina kurti tokias verslo sistemas, kurios būtų lengvai ir greitai papildomos naujais komponentais ir naujom sistemomis [6].

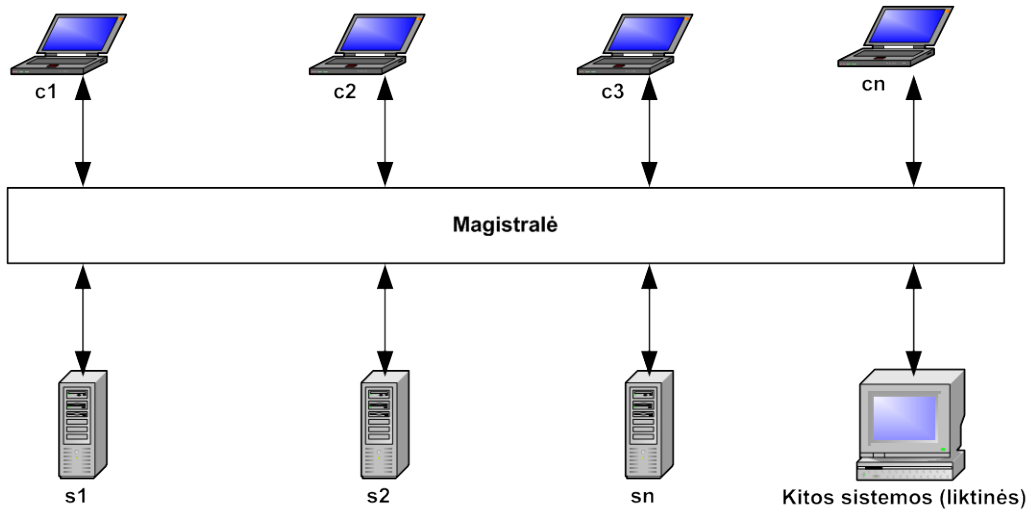
Paslauga – tai standartizuota pasikartojanti verslo užduotis. Kiekviena SOA paslauga realizuoja konkrečią veiklos funkciją [7]. Kiekvienas servisas – tai tam tikras procesas, kuris realizuoja tam tikrą veiklą. Į SOA pavyzdžiui įeina: santykių su klientais valdymas, įmonės išteklių planavimas, elektroninės prekybos, finansų valdymo paslaugos. Servisais besiremianti architektūra leidžia padaryti pagrindinį įmonės verslą nepriklausomą nuo jos informacinių technologijų ir programinės įrangos gamintojų. Šios architektūros pagrindas - Įmonės servisų

magistralė, prie kurios jungiami likę komponentai, atliekantys tam tikras paslaugas (žiūrėti 2.5 Įmonės servisų magistralė).

2.5 Įmonės servisų magistralė (angl. Enterprise Service Bus)

Įmonės servisų magistralė - tai tam tikrų, specifinių (įmonės), servisų infrastruktūra, kuri yra valdoma naudojant vadinamąją centrinę magistralę [8]. Centrinė magistralė yra tarpinė programinė įranga, skirta atskirų programų arba sistemų integracijai ir atliekanti duomenų transformavimo ir persiuntimo funkcijas [8]. Ji sujungia visas įmonės paslaugas ir leidžia joms bendradarbiauti. Ši magistralė gali atlikti adapterio funkciją ir užtikrinti, kad kiekvienas prie jos prijungtas servisas ar sistema gaus informaciją reikalingu formatu (aktualu, kada jungiamos lygtinės sistemos).

Taigi iš klientinės programinės įrangos servisai pasiekiami naudojant sąsają - magistralę. Bendravimas vyksta per magistralę naudojant žinutes: klientinė programinė įranga ir servisai siunčia ir gauna žinutes. Dingsta tiesioginis kliento-serviso bendravimas. Pati magistralė ir klientas turi „žinoti“ tam tikrus žinučių tipus, kad galėtų pateikti juos suprantančiam servisui.



2 pav. Įmonės servisų magistralė.

2 pav. vaizduoja prie magistralės „prijungtus“ servisus ir klientus, be to liktinę sistemą, kuri integruota į šią infrastruktūrą.

Įmonės servisų magistrale paremtų sistemų savybės ir reikalavimai [8]:

- Žinutės gali būti siunčiamos sinchroniškai arba asinchroniškai.
- Pagal žinutės tipą turi būti nustatomas maršrutas.
- Privalo vykti transformavimas (paruošimas siuntimui) ir gautos žinutės apdorojimas.
- Šioje sistemoje per servisus yra įgyvendinami verslo procesai.
- Atliekamas servisų koordinavimas.
- Apsauga. Palaiko žinučių šifravimą, patikimą pristatymą (naudojant eiles), transakcijas.
- Galimybė stebėti ir registruoti įvykius (angl. logging).
- Daugelio programavimo kalbų palaikymas.
- Standartinis žinutės formatas yra XML.
- Žiniatinklio servisų palaikymas.
- Integracija su liktinėm sistemom.
- Žinučių validavimas naudojant XML schemas.

Įmonės servisų magistrale paremtų sistemų privalumai:

- Greitesnė ir pigesnė egzistuojančių sistemų integracija.
- Lankstumas, pasikeitus reikalavimams.
- Standartų laikymasis.
- Apima įmonės procesus.
- Daugiau konfigūracijos negu kodo rašymo (skirto integravimui).
- Servisai gali būti pakartotinai panaudojami.

Įmonės servisų magistrale paremtų sistemų trūkumai:

- Dažnai palaikomas tik žinutėmis paremtas bendravimas.
- Sudėtingas valdymas, sunku perprasti tokių sistemų konfigūravimą, diegimą, palaikymą.
- Įvedamas papildomas sluoksnis, kuris sumažina spartą, lyginant su tiesioginiu kliento serviso bendravimu.
- Kartais sistema tampa jautria defektams. Atsiradus defektui, sutrinka visa sistema (ne jos dalys).
- Servisai turi būti atskirai sukurti ir pritaikyti.

2.6 Įvykiais paremta architektūra

Įvykiais paremta architektūra - tai reikalavimų visuma besiremianti įvykiais, kuria vadovaujantis yra kuriama programinė įranga [9]. Ji turi apimti:

- Įvykio inicijavimą.
- Įvykio aptikimą.
- Reakciją į įvykį (būsenos pasikeitimas).

Įvykis čia suprantamas, kaip būsenos sistemoje pasikeitimas. Pasikeitus būsenai tam tikriems sistemos elementams yra transliuojamas pranešimas apie jos pasikeitimą. Šie elementai į būsenos pasikeitimus reaguoja pagal tam tikras ir iš anksto apibrėžtas taisykles.

Pavyzdžiui, kai vartotojas, naudodamasis elektronine sistema, perka prekę, prekės būsena pakinta iš parduodamos į parduotą. Tokioje sistemoje toks būsenos pakeitimas gali būti registruojamas kaip įvykis. Tokia informacija apie įvykį yra transliuojama tam tikriems sistemos elementams (pavyzdžiui servisams), kurie atitinkamai į tai reaguoja.

Supaprastintai tokia sistema susideda iš [9]:

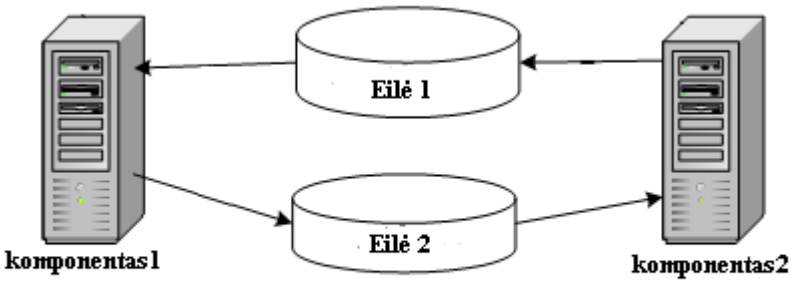
- Įvykio transliuotojo.
- Reaguotojo į įvykį.
- Įvykių perdavimo kanalų.

Reaguotojas papildomai gali transformuoti ir persiųsti įvykį kitiems. Kadangi tokios sistemos yra asinchroninės ir veikia heterogeniniuose tinkluose, įvykio persiuntimui dažniausiai naudojamos eilių žinutėmis paremtos technologijos (2.7 Žinučių siuntimo technologija naudojanti eilių duomenų struktūrą). Ši paradigma naudojama į paslaugas orientuotoje architektūroje, kur servisai yra reaguotojai į įvykius, ateinančius per Įmonės servisų magistralę (2.4 Servisais (paslaugomis) besiremianti architektūra (SOA)).

2.7 Žinučių siuntimo technologija naudojanti eilių duomenų struktūrą

Tai bendravimo tarp paskirstytos sistemos komponentų protokolas paremtas žinutėmis, kurios saugomos tam tikroje saugykloje vadinamoje eile. Saugojimas vyksta tol, kol servisas atsakingas už tą eilę, jas nuskaito (3 pav.). Šis modelis įgalina bendrauti sistemas,

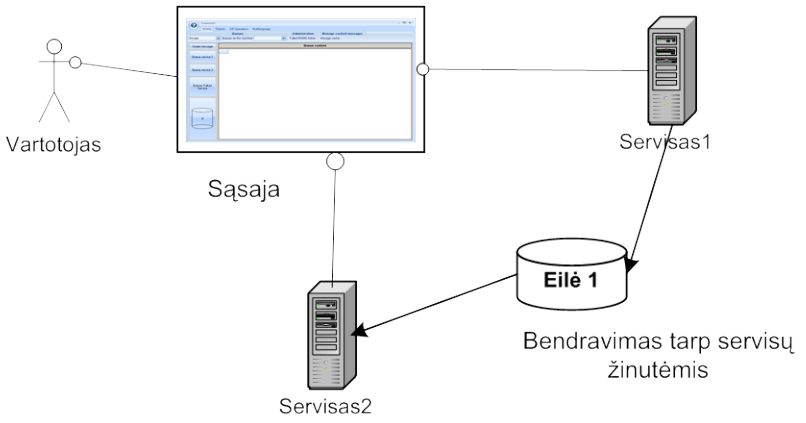
kurios randasi heterogeniniuose tinkluose, kai viena sistema ne visada gali susisiekti su kita, tačiau jos visada susisiekia su žinučių saugykla [10].



3 pav. Eilių žinutėmis paremtas bendravimas tarp komponentų.

Pavyzdžiui, vienas komponentas gauna duomenis iš išorės, su jais atlieka tam tikras manipuliacijas bei rezultatus, žinutės pavidalu, siunčia antrajam komponentui. Tačiau jei antrasis komponentas tuo metu yra nepasiekiamas (pvz. veikiantis servisas „nulūžo“ ir pan.) rezultatai yra siunčiami į jo eilę. Antrasis komponentas, kai tik turi galimybę, skaito tą eilę ir „paima“ kiekvieną šios eilės žinutę, galiausiai ją apdoroja pagal tam tikras taisykles (pvz. atvaizduoja vartotojui). Žinutės čia gali turėti įvairius duomenis net ir tekstą. Tokiu būdu tarp komponentų vyksta eilėmis paremtas bendravimas.

Toks bendravimo modelis yra naudojamas į paslaugas orientuotoje architektūroje (2.4 Servisais (paslaugomis) besiremianti architektūra (SOA)), kuomet vienas servisas turi pranešti kitam apie tam tikrą įvykį, t.y. vyksta servisų sinchronizacija (4 pav.).



4 pav. Tipinės paskirstytos sistemos struktūra.

4 pav. matome vartotoją, kuris per grafinę sąsają (pvz. žiniatinklio) naudojasi servisais. Grafinėje sąsajoje kiekvienas servisas atsakingas už savo sritį. Todėl, kuomet pakinta vieno serviso būsena (vartotojas inicijuoja įvykį), apie tai turi žinoti kitas servisas ir atitinkamai imtis veiksmų. Veiksmai gali būti tokie: serviso būsenos pasikeitimas, duomenų iš duomenų bazės nuskaitymas ir jų užkrovimas tam tikroje grafinės sąsajos srityje.

Toks servisų bendravimas vyksta žinutėmis, naudojant eiles bei įgalina servisus tapti nepriklausomais ir atitikti į paslaugas orientuotos architektūros savybes.

Tokios sistemos pavyzdys yra bet kuri sudėtingesnė svetainė, kurioje yra navigacijos funkcionalumas bei turinio užkrovimo, pagal navigacijos duomenis, galimybė. Čia servisai gali būti atsakingi už navigaciją, turinio užkrovimą, elementų išdėstymą sąsajoje. Pagal navigacijos serviso būseną, yra užkraunamas turinys ir atliekamas šių duomenų išdėstymas sąsajoje. Tokiu būdu vyksta bendravimas tarp servisų naudojant žinutes ir eiles, užtikrinant, kad jie tikrai gaus pranešimus. Žinoma, servisai gali būti ir tiesiogiai nesusiję su vartotojo sąsaja, pavyzdžiui, klaidas apdorojantys, duomenų importavimo ir kt.

Galime identifikuoti tokius žinutėmis paremtos technologijos privalumus [11]:

- Užtikrina komponentų bendravimą nevienalyčiuose tinkluose.
- Užtikrina, kad įvykus defektui (tinkle, kompiuteryje, programoje) žinutė nebus prarandama.
- Toks bendravimas užtikrina komponentų nepriklausomumą vienas nuo kito.
- Žinutėmis gali būti perduodami ir duomenys, ir rezultatai. Pagal pranešimų tipą yra kviečiamos atitinkamos procedūros.
- Komponentas, išsiuntęs žinutę, gali toliau sėkmingai veikti; jam nereikia rūpintis ar ji bus gauta ar ne. Atėjus atsakymui, jo darbas yra pertraukiamas ir vyksta atėjusio pranešimo apdorojimas (sinchroniškai arba asinchroniškai), kviečiama tam tikra apdorojimo procedūra.
- Galimybė atidėti atėjusių žinučių apdorojimą.
- Atliekamas žinučių registravimas žurnalų tipų failuose (angl. logging).
- Suteikiama galimybė kurti atsargines žinučių kopijas.

Papildomos programinės įrangos (papildomo sluoksnio) įdėjimas į sistemą dažnai sulėtina jos darbą, taigi tokios technologijos netinkamos kai reikia labai greito atsako. Metodas netinkamas kuomet be atsakymo iš komponento sistema negali toliau veikti. Be to

tokios technologijos yra netinkamos, kuomet yra naudojamos paskirstytos transakcijos (pvz. bankinėse sistemose) [11].

2.8 Žinučių siuntimo tipai

Bus aptariami du žinučių siuntimo modeliai:

- Užklausos atsako siuntimas (angl. request/response).
- Siuntimas transliuojant (angl. publish).

Šiuose siuntimo modeliuose, kada servisas išsiunčia žinutę, valdymas grąžinamas pagrindinei proceso gijai. Todėl, kol ateis atsakymas, servisas toliau gali veikti – nepertraukiamas jo darbas. Siuntimo valdymas perduodamas komponentui, kuris atsakingas už žinučių siuntimą, saugojimą, šifravimą, pristatymą [12]. Ši technologija sumažina sistemos veikimo greitį, kadangi įvedama papildoma programinė įranga, kuri aptarnauja žinutes, tačiau padidina bendrą sistemos patikimumą.

2.8.1 Užklausos atsako siuntimas

Tai nuotolinės procedūros kvietimo savybes turintis siuntimas (2.3 Nuotolinių procedūrų kvietimo technologija) [12]. Tik šiame modelyje išsiuntus žinutę, valdymas grąžinamas pagrindinei proceso gijai, be to programinė įranga pasirūpina išsiųstos žinutės pristatymu. Jei servisas (komponentas), kuriam skirta žinutė, nepasiekiamas, ji išsaugojama tam tikroje vietoje (dažniausiai atskirame serveryje) [12]. Taigi toks siuntimas tinka probleminiuose tinkluose, kurie turi mažą patikimumą [12]. Naudojant paprastą nuotolinių procedūrų kvietimo technologiją, pagrindinė gija kurį laiką lauks atsakymo, nors serveris ir negalės jo atsiųsti – taigi bus stabdomas sistemos darbas. Pagrindinė šio siuntimo metodo savybė – komponentas, kuriam siunčiama žinutė, turi būti žinomas (žinomas jo adresas) [12]. Gavęs žinutę, komponentas atitinkamai į ją reaguoja - pakeičia savo būseną. Šis metodas pavaizduotas 4 pav. Čia užklausos/atsako tipo bendravimas vyksta tarp pirmo ir antro serviso.

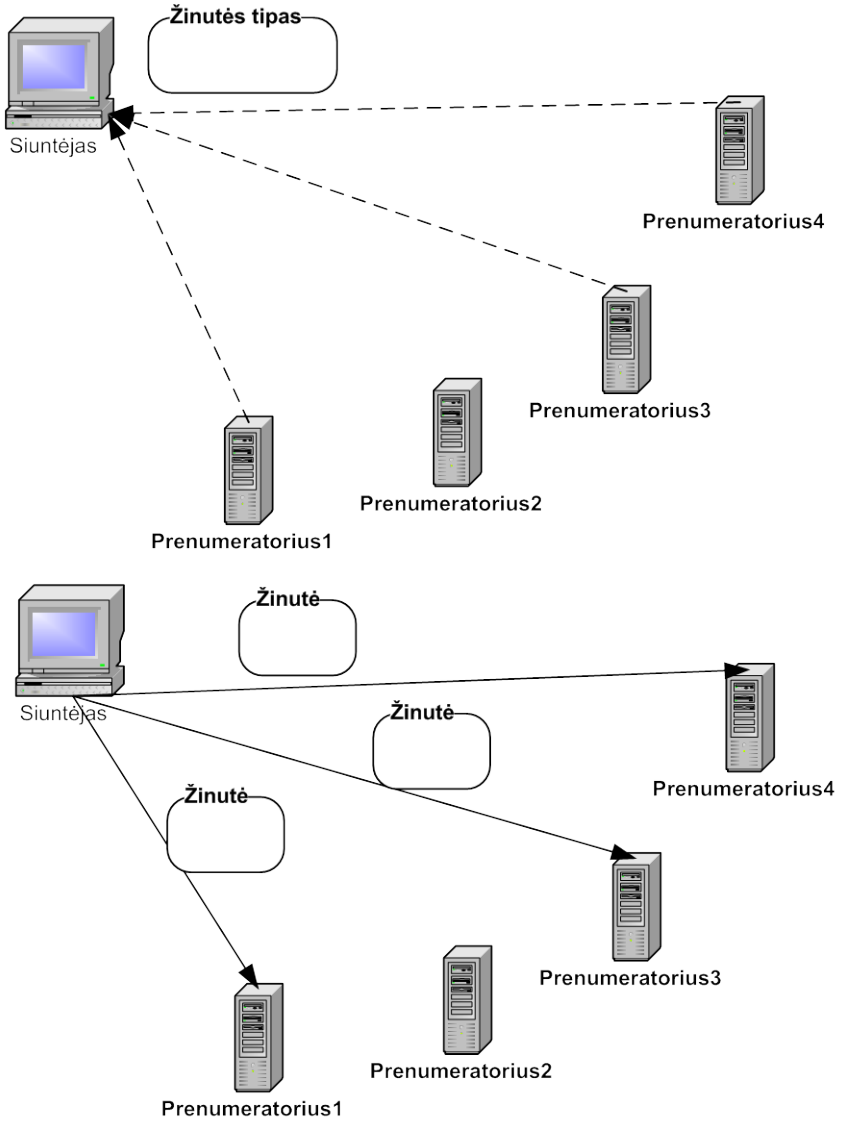
2.8.2 Siuntimas transliuojant

Šio metodo pagrindinė idėja - siuntėjas tiksliai nežino kam siunčia žinutę [12]. Todėl pirmiausia servisai, kurie nori gauti žinutes, „prenumeruojasi“ prie serviso, kuris jas siunčia. Taigi siuntėjas „turi“ gavėjų sąrašą [12].

Transliavimas taip pat gali vykti ir pagal žinutės tipą. Vieno tipo žinutes prenumeruojasi vieni servisai, kito tipo kiti. Šiuo atveju siuntėjas žino, kokio tipo žinutes, kurie gavėjai nori gauti. Transliavimo metu, papildoma programinė įranga, padaro žinutės kopijas ir jas pristato kiekvienam prenumeratoriui. Taigi šio modelio veikimas susideda iš tokių etapų (5 pav.):

1) Kiekvienas gavėjas siuntėjui praneša apie norą gauti tam tikras žinutes (pagal tipą).
2) Siuntėjas „žino“ gavėjus.
3) Siuntėjas siunčią žinutę visiems jos gavėjams (pagal žinutės tipą).

Šis siuntimo modelis naudojamas verslo sistemose, kada vykdomi masiniai pranešimai. Pavyzdžiui, sistemos komponentams, naudojantis šiuo siuntimo metodu, galima pranešti apie užsakymo atšaukimą, apie tai, jog prekės nebėra pardavime ir pan. [12].



5 pav. Grafiškai vaizduojamas prenumeravimas ir siuntimas prenumeratoriams (transliavimas)

2.9 Trečių šalių komponentai skirti bendravimui tarp servisų

Šiame darbe dėmesys bus skiriamas komponentams, kurie sukurti trečių šalių, yra atviro kodo ir nemokami, be to palaiko Microsoft žinučių siuntimo technologiją (MSMQ). Kadangi karkasas veikia Windows platformoje, nagrinėjami tik tie komponentai, kurie remiasi .NET ir WCF technologijomis. Kitos jų savybės:

- Turi priklausyti įmonės servisų magistralės tipui, (2.5 Įmonės servisų magistralė (angl. Enterprise Service Bus)), t.y. turi turėti jos savybes.
- Komponentas turi palaikyti užklausos/atsako bendravimą tarp programos komponentų.
- Turi palaikyti transliavimo-prenumeravimo mechanizmus.
- Komponentas turi palaikyti XML žinučių formatą.

2.10 Biznio sluoksnio karkasas – magistrinio darbo metu kurta sistema

Šiais laikais programavimui yra keliami aukšti reikalavimai, kadangi klientai reikalauja kokybiškos programinės įrangos ir prašo ją sukurti kuo greičiau [13]. Taigi šio projekto tikslas kuo daugiau supaprastinti projektavimo, testavimo ir programavimo darbus, kurie yra dažniausiai pasikartojantys kuriant naują programinę įrangą bei suteikti programuotojui daugiau galimybių. Programavimo įmonėms greitas ir efektyvus sistemų kūrimas yra aktuali problema [13].

Kadangi vis daugiau įmonių kuria į verslą orientuotą programinę įrangą, kuri įgyvendina tam tikrus procesus, vis dažniau yra reikalingos automatizuotos kūrimo priemonės [14].

Dėl šios priežasties buvo nagrinėtos verslui kuriamos sistemos. Surinkti reikalavimai, kurie dažniausiai pasitaiko kuriant tokias sistemas. Išanalizuoti dažniausiai naudojami komponentai, technologijos ir architektūros, tokios kaip klientas-serveris-duomenų bazė ir pan. Tuomet iškelti esminiai reikalavimai naujai sistemai [13]:

- Ši sistema bus naudojama kaip verslui skirtos programinės įrangos pagrindas – karkasas.
- Teiks pakartotinio naudojimo komponentus.
- Kuriama sistema įgyvendins tam tikras funkcijas ir paslaugas (programuotojams iš naujo nebereikės jų įgyvendinti).
- Kuriama sistema supaprastins verslui skirtos programinės įrangos architektūrą, nes bus naudojamas kaip architektūrinis pagrindas.
- Supaprastins bendravimą su duomenų baze, bendravimą tarp programos atskirų modulių, palengvins kuriamų programų plečiamumą.
- Pagreitins dažniausiai pasitaikančius ir pasikartojančius programavimo darbus.
- Sumažins naujų sistemų kūrimo laiką, nepakenkiant jų kokybei.
- Užtikrins greitą ir kokybišką naujų paskirstytų sistemų kūrimą ir testavimą.
- Leis programuotojams ir sistemos projektuotojams praleisti daugiau laiko koncentruojantis į kuriamos programos specifines sritis, o ne į programos architektūrą.

Karkaso pirminiai vartotojai: sistemų analitikai, projektų vadovai, programų kūrėjai, testuotojai ir kiti asmenys atsakingi už naujos informacinės sistemos įgyvendinimo darbus [13].

Galiausiai buvo sudaryta specifikacija, architektūros modelis bei įgyvendinta ir ištestuota sistema.

Šis magistrinis projektas buvo grupinis, jį atliko du žmonės, todėl bus pateiktas tik autoriaus įgyvendintas funkcionalumas (Lentelė 1).

Lentelė 1. Biznio sluoksnio karkaso funkcionalumas.

Funkcionalumas	**Įgyvendino**
Neteisingų duomenų apdorojimas	Žygimantas Šilanskas
Sistemos funkcionalumo pakartotinis panaudojimas	Žygimantas Šilanskas
Kešavimas ir srautų apdorojimas	Žygimantas Šilanskas
Ataskaitų kūrimo komponentas	Žygimantas Šilanskas
Žinutėmis paremtas bendravimas tarp komponentų	Žygimantas Šilanskas

Lentelėje nurodytas funkcionalumas realizuotas operacijų pagrindu. Užtenka kuriamoje sistemoje įtraukti šio karkaso teikiamas bibliotekas ir panaudoti atitinkamus interfeisus, toliau viskuo pasirūpina vidiniai karkaso komponentai. Plačiau apie karkaso architektūrą skyriuje 3 BIZNIO SLUOKSNIO KARKASO ESMINIAI ASPEKTAI.

Lentelėje taip pat matome, kad viena iš šio karkaso teikiamų funkcijų yra bendravimas tarp komponentų naudojant žinutes ir eiles. Iš reikalavimų, pateiktų aukščiau, galime spręsti, kad toks bendravimo tipas ypač reikalingas įmonėje, kurioje:

- Įmonės procesams automatizuoti naudojama programinė įranga, paremta paskirstyta architektūra.
- Veikia daug servisų.
- Servisai išdėstyti nevienalytėje aplinkoje.
- Servisai turi sinchronizuotis tarpusavyje.
- Jais naudojasi daug klientų.
- Yra servisų, kurie importuoja duomenis į sistemą iš išorės ir atlieka duomenų tikrinimą bei transformaciją.
- Servisus planuojama panaudoti kitose verslo sistemose.

2.11 Žinučių siuntimo technologija naudojama Biznio sluoksnio karkase

Magistrinio projekto metu kurta sistema naudojama Windows operacinėje sistemoje, todėl joje buvo įgyvendintos .NET technologijos - tai Microsoft firmos programų kūrimo karkasas. Naudojantis šios technologijos poaibiu, pavadintu WCF, galima kurti paskirstytas sistemas. Sukurti komponentai bendrauja tarpusavyje naudodami nuotolinių procedūrų kvietimo technologiją. Tačiau WCF pagrindiniai komunikaciniai modeliai neapima bendravimo paremto žinutėmis. Vienas iš Biznio sluoksnio karkaso reikalavimų buvo bendravimas žinutėmis [13]. Taigi buvo atlikta analizė ir nuspręsta, kad kurti naujo komponento nėra prasmės, nes egzistuoja Microsoft technologija (MSMQ), kuri palaiko žinutėmis paremtą bendravimą. Taip pat egzistuoja įvairūs trečių šalių nemokami komponentai, kurie remiasi šia technologija ir siūlo:

- Daug pagalbinių funkcijų.
- Įvairius žinučių siuntimo mechanizmus, tokius kaip: transliavimas-prenumeravimas, siuntimas su atsaku.
- Klaidų valdymo ir toleravimo mechanizmus.

Taigi, nagrinėti tik tie nemokami komponentai, kurie remiasi Microsoft technologija.

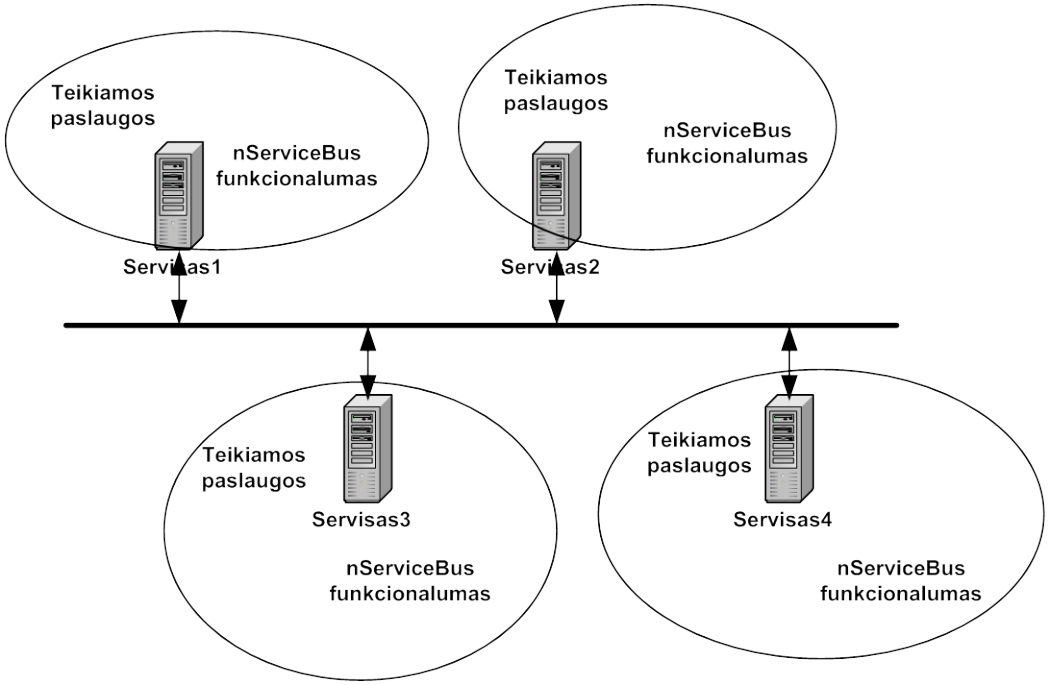
Buvo parinktas populiariausias, nemokamas, subrandintas, skirtas .NET, trečių šalių komponentas nServiceBus. Pagrindinis šio komponento tikslas yra palengvinti bendravimą tarp paskirstytos sistemos dalių. Komponentas suteikia galimybę sistemos elementams bendrauti tarpusavyje naudojant MSMQ protokolą.

2.11.1 Komponentas nServiceBus

Tai įmonės servisų magistralės tipui priklausantis komponentas (2.5 Įmonės servisų magistralė (angl. Enterprise Service Bus)). Tarp programos dalių palaiko užklausos/atsako bendravimą. Paskirstytos programos komponentams, vienam su kitu keičiantis duomenimis, užtikrina daug didesnį patikimumą. Tereikia duomenų/rezultatų objektą praplėsti specialiu interfeisu ir naudojantis nServiceBus funkcionalumu, jį galima siųsti tinklu kaip žinutę. Šis komponentas yra naudojamas kritinėse sistemose. Pagrindinės jo savybės, privalumai, trūkumai [15]:

- Sukurtas panaudojant objektinio programavimo modelį.
- Turi įvairius klaidų tolerancijos ir valdymo mechanizmus.

- Įvykus klaidai, neprarandant svarbių duomenų, paskirstytą sistemą galima atstatyti į ankstesnę būseną. Žinutės atstatomos ir patalpinamos į prieš tai buvusią eilę.
- Palaiko transliavimo-prenumeravimo mechanizmus.
- Palaiko XML žinučių formatą.
- Suderinamas su WCF. Per nServiceBus galima naudotis WCF technologijos teikiamais privalumais.
- Architektūra nėra centralizuota (6 pav.). Kiekvienas servisas įgyvendina nServiceBus teikiamus interfeisus ir tampa jo „dalimi“. Dėl šios priežasties sistema greičiau veikia (nėra centrinio serverio – nėra jo apkrovos) ir yra lengviau praplečiama.
- Sudėtinga integruoti egzistuojančias sistemas (lygtines ir pan.).
- Trūksta lokalizuotos dokumentacijos.



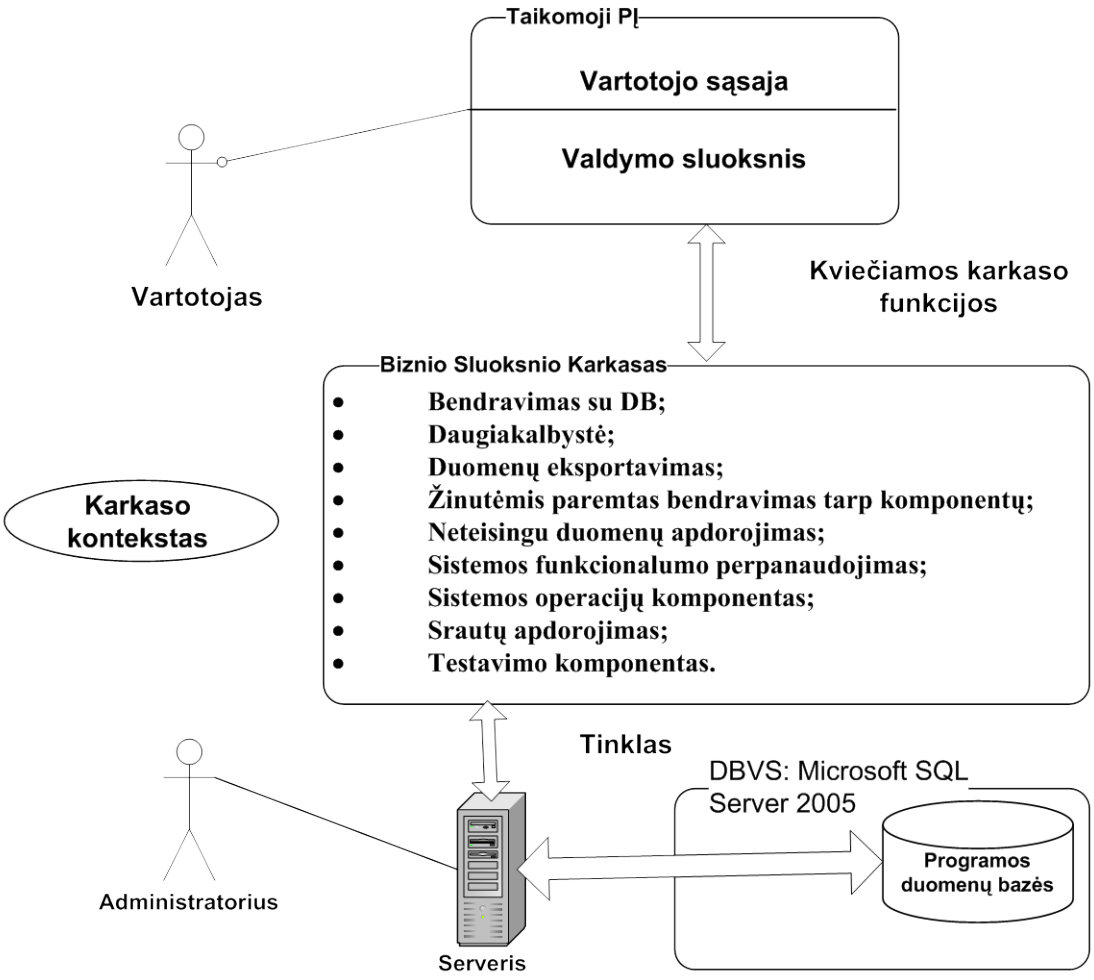
6 pav. Trečių šalių komponento nServiceBus integracija paskirstytoje sistemoje.

3 BIZNIO SLUOKSNIO KARKASO ESMINIAI ASPEKTAI

3.1 Sistemos veiklos kontekstas

Magistrinis projektas yra biznio sluoksnio programinės įrangos pamatas – karkasas (2.10 Biznio sluoksnio karkasas – magistrinio darbo metu kurta sistema) [13]. Santykius tarp karkaso bei tam tikros taikomosios programos, kurios pamatas jis yra, apibrėžia kita diagrama [13].

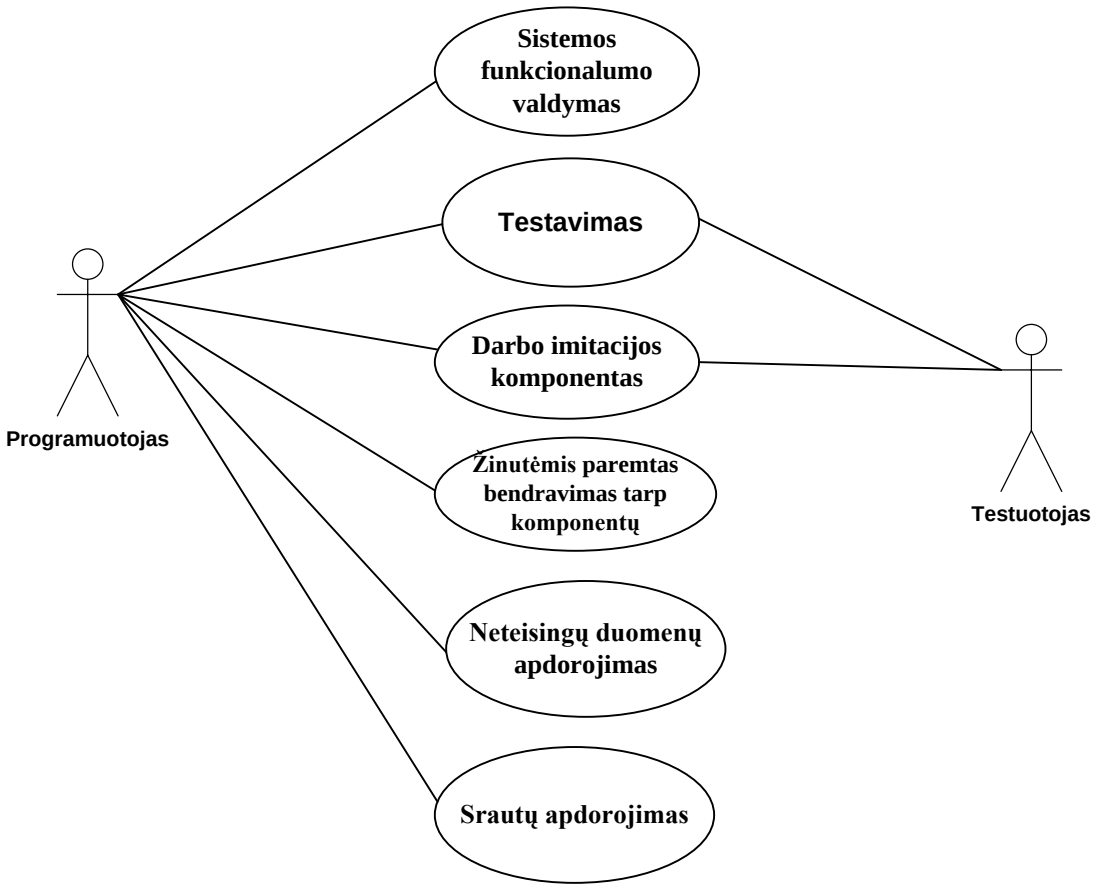
Diagramoje (7 pav.) matome, jog taikomoji programa, atlikdama verslo sluoksnio funkcijas, naudojasi karkaso komponentais. Karkasas apibrėžiamas kaip sluoksnis tarp duomenų bazės bei taikomosios programos.



7 pav. Santykiai tarp karkaso bei taikomosios programos, kurios pamatas jis yra.

Sistemos ribos ir funkcijos

Sistemos riboms atvaizduoti yra pateikta supaprastinta panaudos atvejų diagrama (8 pav.). Sistemos fizines ribas galime matyti 7 pav.



8 pav. Biznio sluoksnio karkaso supaprastinta panaudos atvejų diagrama (PAD).

Sistemoje identifikuojami du pagrindiniai aktoriai:

- Programuotojas;
- Testuotojas.

Programuotojas – tai aktorius, kuris gali naudotis visu sistemos teikiamu funkcionalumu. Prie PAD aktoriaus programuotojo priskiriamas sistemų analitikas ir architektas, kadangi kuriant programinę įrangą ir jie analizuos bei panaudos šį karkasą [13].

Testuotojas, remiantis panaudos atvejų diagrama, naudoja tik su sistemos testavimu susijusius komponentus [13].

Studento Žygimanto Šilansko nagrinėti panaudos atvejai (PA):

Lentelė 2. Pirmas PA.

Sistemos funkcionalumo pakartotinis panaudojimas	
Aprašas	Šio PA dėka daugumą sistemos elementų (tam tikras savybes, laukus, konstantas, komponentus) bus galima pateikti naudojant specialius konfigūracinius failus. Palengvės programos konfigūravimas ir išplėtimas, nes užteks atlikti atitinkamo konfigūracinio failo redagavimą.
Aktoriai	Programuotojas.
Naudojimas	Funkcionalumas į kuriamą sistemą įtraukiamas kaip papildoma biblioteka. Tinkamai sukonfigūruoti failai (pagal dokumentuotas taisykles) yra nurodomi iškvietus tam tikras, pakartotiniam panaudojimui skirtas, funkcijas. Šiuos failus minėtos funkcijos ir analizuoja, susiedamos juos su programos sąsajomis ir objektais.

Lentelė 3. Antras PA

Testavimas	
Aprašas	Naudojant šį funkcionalumą, kuriamoje programinėje įrangoje galima atlikti vieneto testavimą, integracinį testavimą, imituoti tam tikrus, dar nesukurtus, testavimo atvejus.
Aktoriai	Programuotojas, testuotojas.
Naudojimas	Funkcionalumas į kuriamą sistemą įtraukiamas kaip papildoma biblioteka. Kviečiami testavimą atliekantys metodai bei nurodomi testuojami objektai ir funkcijos. Šiems metodams yra pateikiami testavimui skirti duomenys ir laukiami rezultatai.

Lentelė 4. Trečias PA.

Darbo imitacijos komponentas	
Aprašas	Šis PA skirtas komponentų, kurie dar nėra pilnai sukurti arba lėtai veikiantys, testavimui. Pavyzdžiui, testuoti funkcijas, kurios naudojasi dar neįgyvendinta duomenų baze arba testuoti funkcijas, kurios naudojasi duomenų baze per lėtai veikiantį tinklą.
Aktoriai	Programuotojas, Testuotojas.
Naudojimas	Šis funkcionalumas naudojamas testuojant sistemą (per testavimo komponento funkcijas). Kviečiami veikimo imitavimą atliekantys metodai. Šiems metodams yra nurodomas imituojamo objekto interfeisas, imituojama funkcija, jos argumentai ir juos atitinkantys rezultatai.

Lentelė 5. Ketvirtas PA.

Žinutėmis paremtas bendravimas tarp komponentų	
Aprašas	Atskiri sistemos komponentai dažnai duomenis gauna ne tik iš duomenų bazės, bet ir iš kitų komponentų. Tokiais atvejais aktualu panaudoti žinutėmis paremtą bendravimą tarp komponentų. Šis PA naudoja trečių šalių žinučių komponentą paremtą MSMQ ir pateikia supaprastintą sąsają su daug pagalbinių funkcijų bei paprastesnio konfigūravimo galimybe.
Aktoriai	Programuotojas.
Naudojimas	Komponentas į kuriamą sistemą įtraukiamas kaip papildoma biblioteka. Naudojantis jo sąsajomis yra aprašomas žinutės objektas. Pasinaudojant jo teikiamomis funkcijomis ir konfigūraciniais failais šie objektai, pasirinktu siuntimo metodu, yra siunčiami kaip žinutės.

Lentelė 6. Penktas PA.

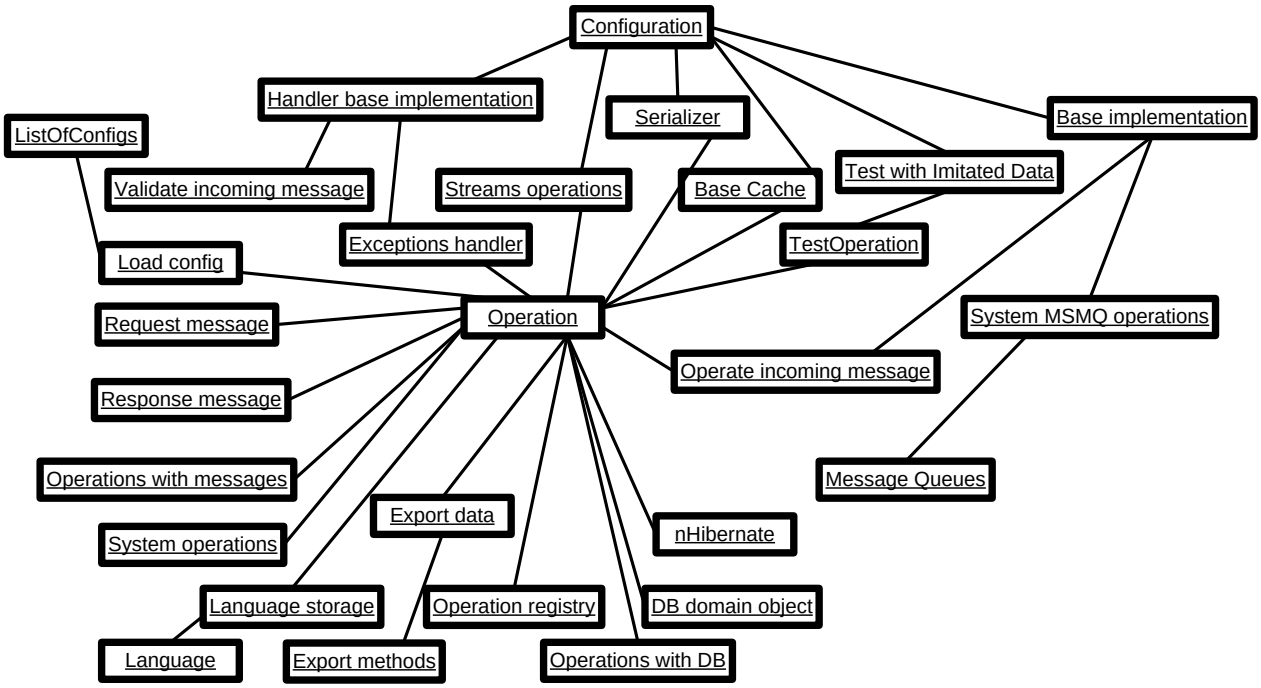
Neteisingų duomenų apdorojimas	
Aprašas	Dažnai sistemoje iškyla nenumatyti atvejai (angl. Exceptions). Tai objektai, kuriuos svarbu „pagauti“ ir apdoroti. Tuomet vartotojui galima pateikti suprantamą informaciją, apie tai kas įvyko bei nurodyti, ką sistema toliau turi daryti (išsijungti, pakartoti veiksmą, užbaigti transakciją ir pan.). Šis PA teikia daug funkcijų, kurių dėka galima atlikti taip vadinamą nenumatytų situacijų valdymą.
Aktoriai	Programuotojas.
Naudojimas	Komponentas į kuriamą sistemą įtraukiamas kaip papildoma biblioteka. Kviečiamos atitinkamos komponento funkcijos, kurios atlieka programuotojo įgyvendintą, nenumatytų situacijų valdymą, galiausiai pateikia informaciją apie klaidą ir grąžina rezultatus.

Lentelė 7. Šeštas PA.

Srautų apdorojimas	
Aprašas	Tai komponentas, kuris apima: objektų serializavimą į tam tikrą formatą (į bitų masyvą, į XML formatą), atstatymą iš šio formato (angl. deserialize), programinį darbą su failais, dažniausiai naudojamų duomenų kešavimą, kešo valdymą.
Aktoriai	Programuotojas
Naudojimas	Komponentas į kuriamą sistemą įtraukiamas kaip papildoma biblioteka. Panaudojant komponento sąsajas, funkcijas ir konfigūracinius failus, galima atlikti objektų serializavimą į tam tikrą formatą, objektų iš šio formato atstatymą, įgyvendinti veiksmus su failais, atlikti dažniausiai naudojamų duomenų kešavimą.

3.2 Sistemos duomenų modelis

Magistrinio projekto sistema visų pirma yra kaip naujos, verslui skirtos programinės įrangos pagrindas. Nėra žinoma galutinio produkto architektūra, naudojamų duomenų bazių schemos, todėl duomenys šiame karkase nėra apibrėžiami. Ši sistema yra kaip funkcijų rinkinys ir architektūrinis pagrindas, kuris yra pateikiamas kuriamai, į verslą orientuotai programinei įrangai. Taigi pagrindiniai šio karkaso duomenys yra jo teikiamos funkcijos. Pradinis duomenų modelis pateiktas klasių diagrama (9 pav.).



9 pav. Duomenų modelį atitinkanti klasių diagrama.

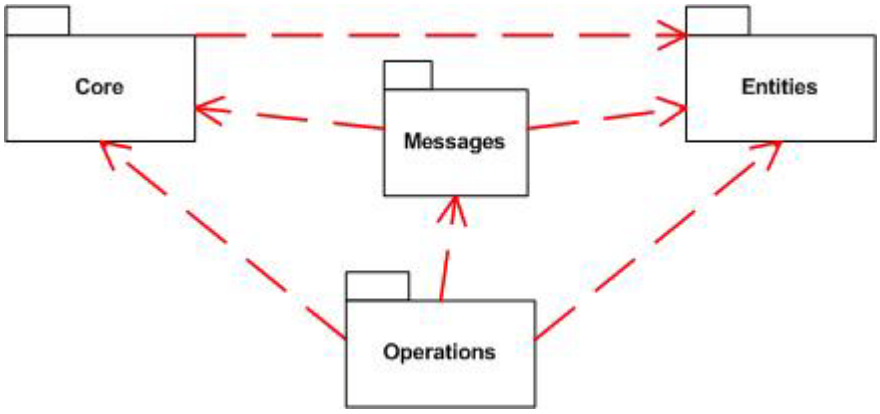
Klasių aprašymai:

- Request message – užklausos žinutės pamatinė klasė, per šią klasę Operation klasei yra paduodami duomenų objektai.
- Response message – atsakymo žinutės pamatinė klasė, per šią klasę iš Operation klasės yra gaunami atsakymo objektai.

- Operation – pagrindinė valdymo klasė, kuri atsakinga už bet kurią sistemoje vykdomą operaciją.
- Operations with DB – su duomenų baze atliekamos operacijos.
- nHibernate – taisyklių rinkinys, skirtas bendravimui tarp duomenų bazės ir sistemos.
- DB domain object – pagrindiniai duomenų bazės lentelių objektai.
- Operations with messages – operacijos atliekamos su žinutėmis.
- System operations – kitos, sistemoje egzistuojančios operacijos.
- Operation registry – operacijų ir veiksmų registravimo būdai.
- Language – kalbos nustatymo klasė.
- Language storage – veiksmų su kalbos objektais ir kalbų objektų saugojimo klasė.
- Export methods – eksportavimo metodo nustatymo klasė.
- Export data – saugomi eksportavimo veiksmai ir algoritmai.
- Message Queues – apibrėžtos sistemoje egzistuojančios eilės.
- System MSMQ operations – pagrindinės sisteminės žinučių operacijos.
- Operate incoming message – apdoroja ateinančią žinutę.
- Base implementation – paruošia darbui su žinutėmis (konfigūracijos failų naudojimas).
- Configuration; Load config – konfigūracinių failų skaitymas, validavimas.
- Exceptions handler - nenumatytų situacijų valdymas, paruošiamieji veiksmai.
- Handler base implementation – pagrindinės nenumatytų situacijų valdymo funkcijos.
- Validate incoming message – tikrina duomenis pagal nenumatytos situacijos tipą, atitinkamai paruošia sistemos būseną.
- ListOfConfigs – saugomas konfigūracinių failų sąrašas.
- Streams operations – operacijos su srautais (failai, atmintis).
- Base Cache – saugo tam tikrus, nuolat naudojamus duomenis (kešavimas).
- Test with Initated Data – vykdo nurodyto metodo/klasės/komponento, su nurodytais imitaciniais duomenimis testavimą.
- TestOperation – bazinė testavimo operacija: komponentus paruošia testavimui, atlieka konfigūraciją, kviečia kitus testavimo objektus.

3.3 Sistemos architektūra

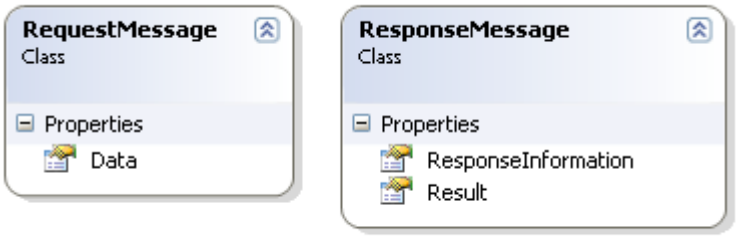
10 pav. matome, kad sistema išskaidyta į keturis paketus. Šie paketai bus aprašyti atskirai.



10 pav. Klasių apjungimas į paketus, pagal jų vaidmenis.

Paketas „Messages“

Paketą sudaro klasės, kurios atsakingos už duomenų bei rezultatų pernešimą. „RequestMessage“ yra bazinė klasė, kurios objektai perneša duomenis, rezultatai gaunami „supakuoti“ į klasės „ResponseMessage“ objektą.



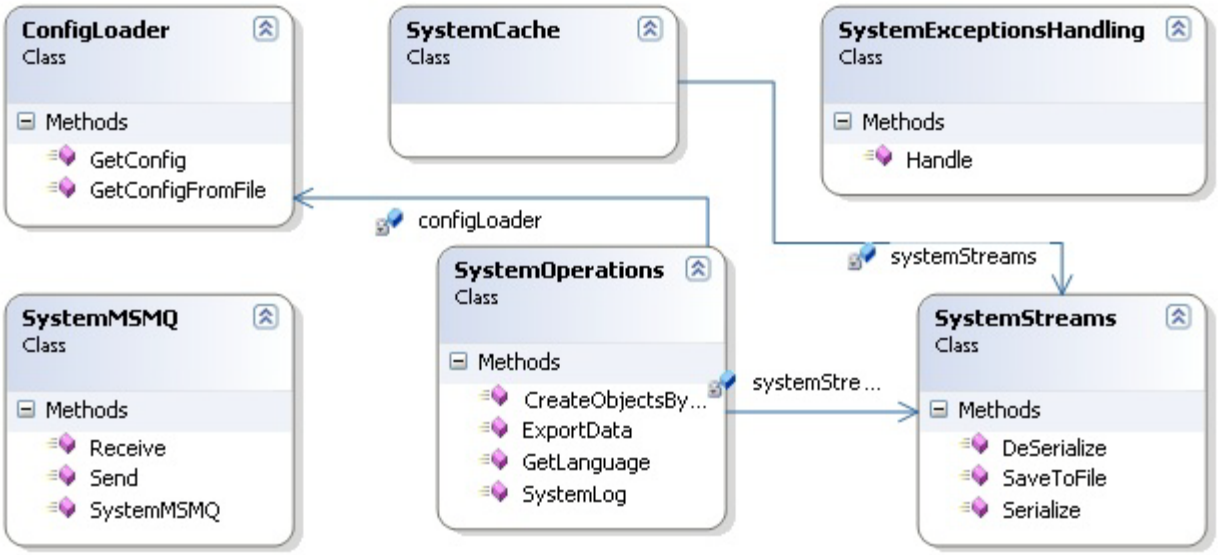
11 pav. Paketas „Messages“ skirtas duomenis ir rezultatams.

Paketas „Operations“

Tai paketas, kuris apima karkaso sąsajas. Čia įgyvendinti visi panaudos atvejai. Šiose klasėse atliekamos visos karkaso operacijos (funkcijos). Naudojant šias klases galima prieiti prie kitų paketų klasių. Taigi, tai yra karkaso funkcionalumas, prie kurio galės „prieiti“ programų kūrėjai.

Paketas „Core“

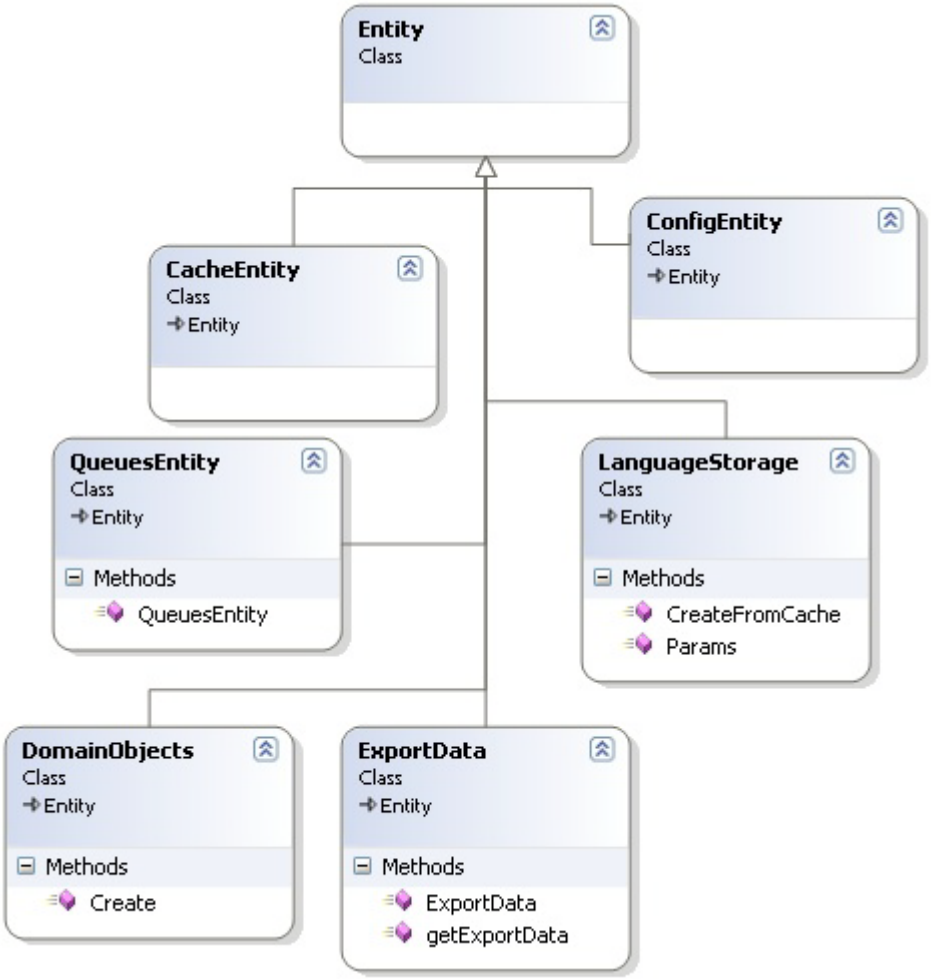
Realizuotos visos sisteminės operacijos, tai tarsi viršutinis sisteminių operacijų lygmuo. Šios klasės naudoja .NET karkaso, įvairias Windows bei trečių šalių bibliotekas. Turi įvairių pagalbinių funkcijų.



12 pav. Paketas „Core“ apima esmines karkaso funkcijas.

Paketas „Entities“

Pakete yra klasės, kurios vienaip ar kitaip saugo duomenis (esybės). Jos paveldėtos iš bazinės klasės „Entity“. Ši klasė turi įvairias, susijusias su esybėmis, operacijas. Šių klasių objektuose saugomais duomenimis naudosis kitose paketuose esančios klasės.



13 pav. Paketas „Entities“ apima klases, skirtas tarpinių duomenų saugojimui.

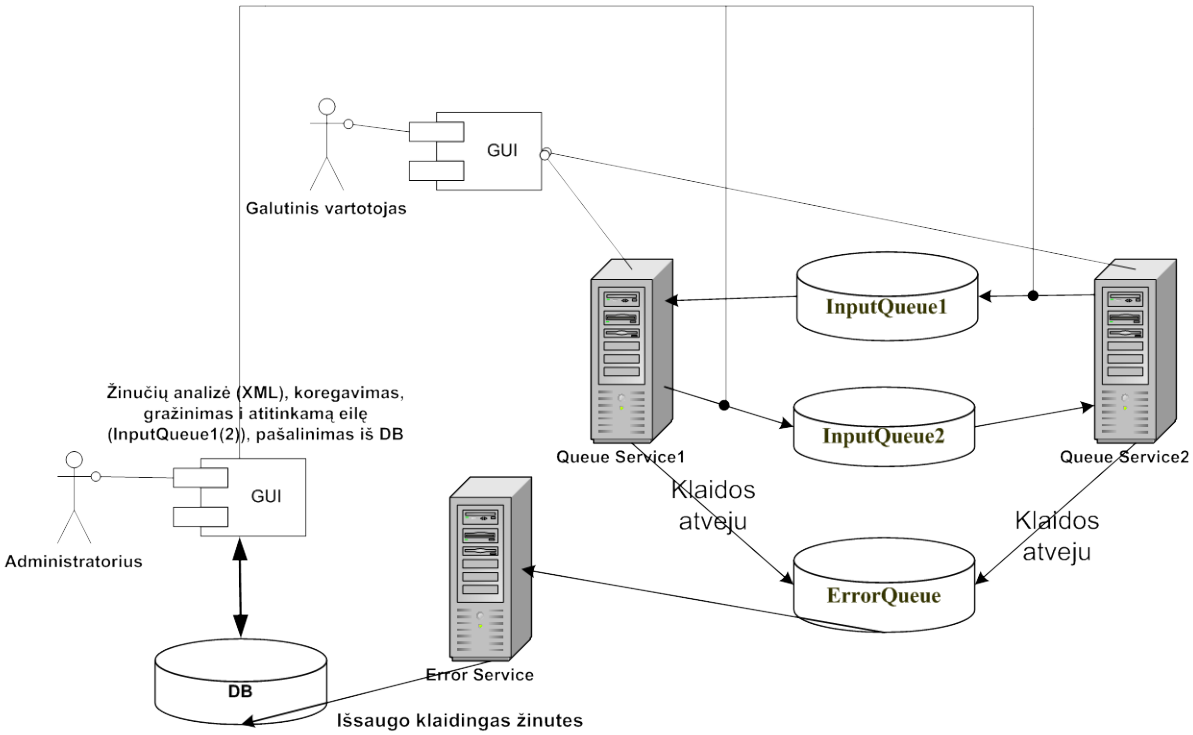
3.4 Karkaso siūlomas žinutėmis paremtas bendravimas

Biznio sluoksnio karkase yra įgyvendintas žinutėmis paremtas funkcionalumas kartu su klaidų valdymo servisu. Nepristatytos žinutės saugomos tam tikroje duomenų saugykloje, kuri organizuota eilės principu. Čia (14 pav.) ateinančios žinutės validuojamos ir apdorojamos pagal tam tikras taisykles. Jei atėjusiai žinutei negali būti atliktas sėkmingas validavimas, ji traktuojama kaip klaidinga ir yra siunčiama į klaidų eilę, kurią stebi klaidų servisas. Minėtas servisas tokio tipo žinutes išsaugo duomenų bazėje. Vartotojas, turintis priėjimą prie duomenų bazės, gali atlikti tolimesnę šių žinučių analizę ir gražinti jas į pirminę eilę (InputQueue1,

InputQueue2). Žinučių validavimo taisyklės priklauso nuo kuriamos sistemos konteksto. Validavimo algoritmas yra specifinis ir aprašomas atskirai.

Toks žinučių apdorojimo scenarijus dažnai naudojamas paskirstytoje programinėje įrangoje, kurioje vyksta duomenų importavimas iš išorinių sistemų. Duomenys transformuojami į žinutes, kurios yra patikrinamos pagal tam tikras taisykles.

Duomenimis čia gali būti ir įvairus tekstas, pateiktas XML formatu ir dvejetainė informacija. Teksto atveju, validavimo algoritmas gali nustatyti teksto kalbą, atitikimą XML taisyklėms, XML schemoms ir pan.



14 pav. Karkaso siūlomas žinutėmis paremtas funkcionalumas su klaidų valdymu.

4 ŽINUČIŲ KOMPONENTO NSERVICEBUS ĮVERTINIMAS

4.1 Bendros karkaso problemos

4.1.1 Paskirstytų sistemų problemos

Paprastai paskirstytas sistemas sudaro ne viena duomenų bazė, veikia daug procesų, įvairių programinės įrangos komponentų ir visa tai yra galimų klaidų šaltinis. Taigi viena iš pagrindinių šių laikų paskirstytų sistemų problemų yra tokių sistemų trapumas. Visų pirma paskirstytos sistemos susideda iš tam tikrų atskirų mažesnių sistemų, kurios veikia atskirai ir tik tam tikrais momentais priklauso viena nuo kitos. Jei kokia nors sistemos dalis „nulūžta“ ar įvyksta kritinė klaida yra lengva prarasti svarbius duomenis, dėl to gali sutrikti ir visos sistemos darbas.

Kitas svarbus dalykas yra reakcijos laikas, t.y. kiek laiko komponentas turi laukti atsakymo į išsiustą žinutę. Kaip užtikrinti, kad komponentas gavo visas žinutes ir reikiama eilės tvarka.

4.1.2 Bendrų reikalavimų sudarymo problemos

Kuriant bendrą programos karkasą yra labai sudėtinga surinkti bendrus reikalavimus, nes kiekviena sistema yra skirtinga. Taigi galima tik nuspėti bendrus dalykus tokius kaip: bendravimas su duomenų baze, bendravimas tarp komponentų, sistemos ir duomenų saugumo užtikrinimas, sistemos testavimas, neteisingų duomenų apdorojimas, duomenų eksportavimas ir kt. Tačiau sunku nuspėti, kokie reikalavimai yra skirti bendravimo greičiui, suderinamumui su jau sukurtomis sistemomis.

4.1.3 Architektūros pasirinkimo problemos

Programinės įrangos architektūra yra vienas iš svarbiausių dalykų. Nustatyti bendrą programinės įrangos architektūrą yra sudėtinga, be to bendrai sukurta programos architektūra, naudojama konkrečiam uždaviniui realizuoti, niekada nebus geresnė negu skirta konkrečiam uždaviniui spręsti. Pagrindiniai bendros programinės įrangos architektūros trūkumai yra greitis ir nepakankamas funkcionalumas. Greitis šiais laikais nėra toks svarbus, kadangi jis kompensuojamas šiuolaikiniais greitais kompiuteriais ir greitu ryšiu, tačiau nepakankamą funkcionalumą teks realizuoti papildomomis programinėmis priemonėmis.

4.1.4 Karkaso gyvavimo problemos

Greitai keičiantis technologijomis, atsirandant naujoms, įvairių šio karkaso trečių šalių komponentų versijomis (.NET karkasas), iškyla problema - kiek šis karkasas gyvuos. Reikės keisti naujesniais tam tikrus šio karkaso komponentus, plėsti funkcionalumą. Taip pat reikia atsižvelgti į tai, jog rinkoje atsiranda naujo tipo karkasai, kuriuose įgyvendintas naujesnis funkcionalumas, nes atsiranda vis naujesnės programų projektavimo idėjos ir naujo tipo taikomųjų programų kūrimo poreikiai. Po kelėtos metų šio karkaso teikiamų funkcijų gali ir nebeprireikti.

4.1.5 Sistemos veikimo greitis

Programų veikimo greitis visada buvo svarbus parametras. Tačiau šiais laikais yra derinami du parametrai:

- Sistemos veikimo greitis.
- Sistemos sukūrimo laikas.

Manipuliuojant šiais parametrais yra nustatoma sistemos kaina ir yra svarbu rasti priimtiną sprendimą. Naudojant daug įvairių trečių šalių komponentų, programos veikimo greitis mažėja. Kiek mažėja, labai priklauso nuo kelėtos parametrų: komponentų skaičiaus, jų tipų, komponento naudojamų funkcijų kiekio ir jo veikimo greičio (vidinio). Atitinkamai galima nagrinėti, kiek tokie komponentai sumažina laiko sąnaudas bei išlaidas. Šiais laikais kompiuterinės technikos sparta labai greit auga, tačiau greitesnių komponentų pasirinkimo tendencija taip pat lieka. Taigi yra labai svarbu atsižvelgti į kuriamos sistemos spartą ir stengtis visais įmanomais būdais ją didinti. Jei kuriamos programos veikimo greitis pastebimai krenta, tai programos karkasas gali būti keičiamas greitesniu, t.y. ieškoma alternatyvų.

4.1.6 Atviro kodo komponentų panaudojimo problemos

Atviro kodo komponentai yra neatskiriama priemonė šių dienų programinės įrangos kūrime, kadangi jie leidžia sukurti programinę įrangą greičiau ir kokybiškiau. Tačiau kuriant naują sistemą, ne visuomet atviro kodo komponentai sutaupo laiką. Jie gali sukelti didelių problemų. Pavyzdžiui, komponentas veikia ne taip kaip tikėtasi, jį sunku palaikyti, konfigūruoti. Gali tekti jį pakeisti naujesniu, kuris turi visai kitokią sąsają, nei tikėtasi. Gali būti ir taip, kad sukūrusi kompanija šio komponento toliau nebevysto, o atsiradusios klaidos gali sukelti nepataisomų problemų. Reikia atsižvelgi ir į tai, kad ateityje gali atsirasti daug

geresnių ir greitesnių trečių šalių sistemų. Todėl renkantis reiktų atkreipti dėmesį ne tik į jo funkcionalumą, bet ir į tai, kas jį sukūrė ir kokios šio komponento perspektyvos ateityje.

4.2 Atviro kodo žinučių komponento nServiceBus problemos

Naudojant Biznio sluoksnio karkaso nServiceBus komponentą iškilo tokios problemos, buvo pastebėti tokie trūkumai:

- Sudėtingas konfigūravimas.
- Lokalios dokumentacijos trūkumas.
- Labai daug nepanaudotų galimybių.
- Sudėtingas valdymas, sunku perprasti diegimą, palaikymą.
- Įvedamas papildomas sluoksnis, kuris sumažina spartą.
- Kaip ir daugelyje trečių šalių komponentų aptikta klaidų, dėl kurių reikėjo nežymiai pakeisti karkaso architektūrą.
- Noras išbandyti naujoves. Rinkoje atsirado daugiau žinučių siuntimo sistemų, apie kurias gerai atsiliepiama.

4.3 Tyrimo tikslai

Remiantis analitinėje dalyje aptartais žinučių siuntimo metodais, trečių šalių žinučių komponentais, karkaso siūlomu žinutėmis paremtu bendravimu, bendromis karkaso problemomis ir komponento nServiceBus įvertinimu sudaryti tokie tiriamojo darbo tikslai:

- Atlikti naujų komponentų, kurie skirti bendravimui tarp sistemos dalių, analizę.
- Pasirinkti iš jų kelis, kurie labiausiai dera Biznio sluoksnio karkaso žinučių siuntimo kontekste ir juos ištirti.
- Tyrimui parinkti siuntimo metodus, kurie yra aptarti analitinėje dalyje.
- Remiantis karkaso siūlomu žinutėmis paremtu bendravimu, sudaryti tinkamus tyrimo modelius, kurie apimtų kiekvieną siuntimo metodą.
- Išmatuoti parinktų trečių šalių žinučių komponentų veikimo greitį.
- Ištirti komponentų veikimą tyrimo modelyje imituojant klaidas.
- Įvertinti šių komponentų naudojimo paprastumą.
- Pateikti komponentų pasirinkimo rekomendacijas vartotojams.
- Išrinkti mažiausiai problemų turintį komponentą.

4.4 Reikalavimai komponentams

Rinkoje tokių sistemų pasiūla palyginus didelė. Nagrinėsime tik tuos trečių šalių komponentus, kurie labiausiai tinka Biznio sluoksnio karkaso žinučių siuntimo kontekste. Jie turi turėti tokias savybes:

- Turi būti nemokamas ir jei įmanoma atviro kodo.
- Turi veikti Windows aplinkoje.
- Turi palaikyti MS .NET technologiją.
- Suderinami su WCF.
- Komponentas turi būti skirtas bendravimui tarp paskirstytos sistemos dalių (servisų, klientų).
- Turi užtikrinti, kad įvykus defektui arba kada adresatas yra nepasiekiamas, žinutė nebūtų parandama.
- Išsiuntusi žinutę, sistema gali toliau sėkmingai atlikti kitas užduotis - jai nereikia „rūpintis“ ar žinutė bus gauta ar ne.
- Būtų galimybė atidėti atėjusių žinučių apdorojimą.
- Žinutės siunčiamos sinchroniškai arba asinchroniškai.
- Komponentas turi palaikyti žinučių šifravimą
- Privalo turėti įvairius klaidų tolerancijos ir valdymo mechanizmus: patikimą pristatymą (naudojant eiles), transakcijas.
- Palaikyti transliavimo-prenumeravimo mechanizmus.
- Architektūra neturi būti centralizuota.

Taigi toliau bus pateiktas populiariausių komponentų, kurie atitiko minėtus kriterijus, trumpas aprašas.

4.5 Komponentas MassTransit

Ganėtinai naujas, įmonės servisų magistralės architektūra paremtas komponentas. Pagrindinės jo savybės, trūkumai:

- Palaiko .NET technologijas.
- Siūlo įvairių papildomų funkcijų.
- Palaiko MSMQ.

- Žinutėms saugoti ir optimizuoti siuntimus papildomai naudoja Microsoft SQL Server.
- Palaiko transliavimo-prenumeravimo žinučių siuntimo mechanizmą.
- Įgyvendinant šį komponentą, didelis dėmesys skirtas užklausos/atsako mechanizmo palaikymui.
- Nepakankamai dokumentuotas.
- Turi klaidų.

4.6 Komponentas Rhino Service Bus

Autorius įgyvendino šį komponentą, išnagrinėjęs nServiceBus ir MassTansit. Įdiegė tam tikrus patobulinimus, tačiau žymiai susiaurino taikymo sritį. Taigi jo architektūra panaši į prieš tai minėtų komponentų. Kitos savybės:

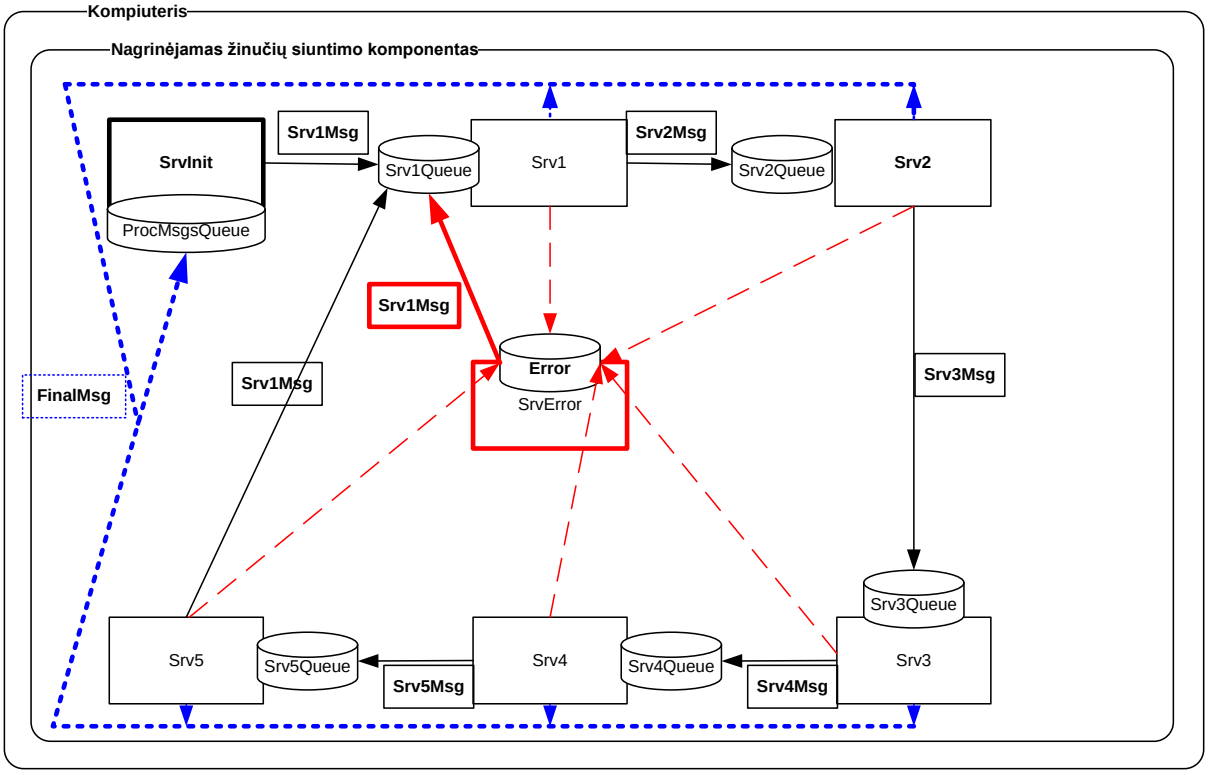
- Supaprastintas konfigūravimas, sumažintas sudėtingumas.
- Lengva perprasti ir naudoti.
- Palaiko žinučių perėmimą.
- Klaidų valdymas, žurnalo vedimo galimybė.
- Palaiko sinchroninį ir asinchroninį siuntimo modelį.
- Žinučių formatas XML.
- Palaiko MSMQ.
- Supaprastinta priėjimo prie šio komponento funkcijų sąsaja.

5 KOMPONENTŲ MASSTRANSIT, RHINO SERVICE BUS TYRIMAS

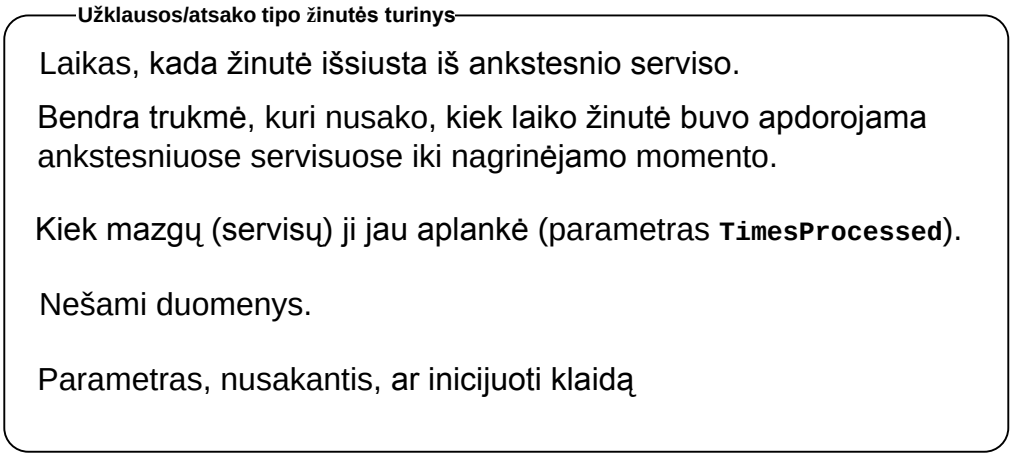
5.1 Tyrimo kontekstas ir tyrimo metodai

Tyrimo metu, naudojantis nagrinėjamais komponentais, yra siunčiamos žinutės ir skaičiuojamas jų apdorojimo servisuose laikas. Visa tai atliekama priklausomai nuo įvairių parametrų, kurie pateikti kitame skyriuje. Prie tyrimo aptarimo (5.6 Išnagrinėtų trečių šalių komponentų, skirtų žinučių, naudojančių eiles, siuntimui, palyginimas ir įvertinimas) taip pat bus įvertinami ir tokie dalykai, kaip: komponento naudojimo ir konfigūravimo paprastumas, dokumentacijos pasiūla.

Tyrimas atliktas pagal servisų sujungimo schemas, kurios pateiktos 15 pav. ir 17 pav. Taip pat čia matome, kokiu maršrutu siunčiamos žinutės. Paveikslėlių paaiškinimai pateikti Lentelė 8 ir Lentelė 9.



15 pav. Žinučių siuntimo ir servisų išdėstymo schema, kada nagrinėjamas užklausos/atsako siuntimas.



16 pav. Užklausos/atsako tipo žinutės formatas.

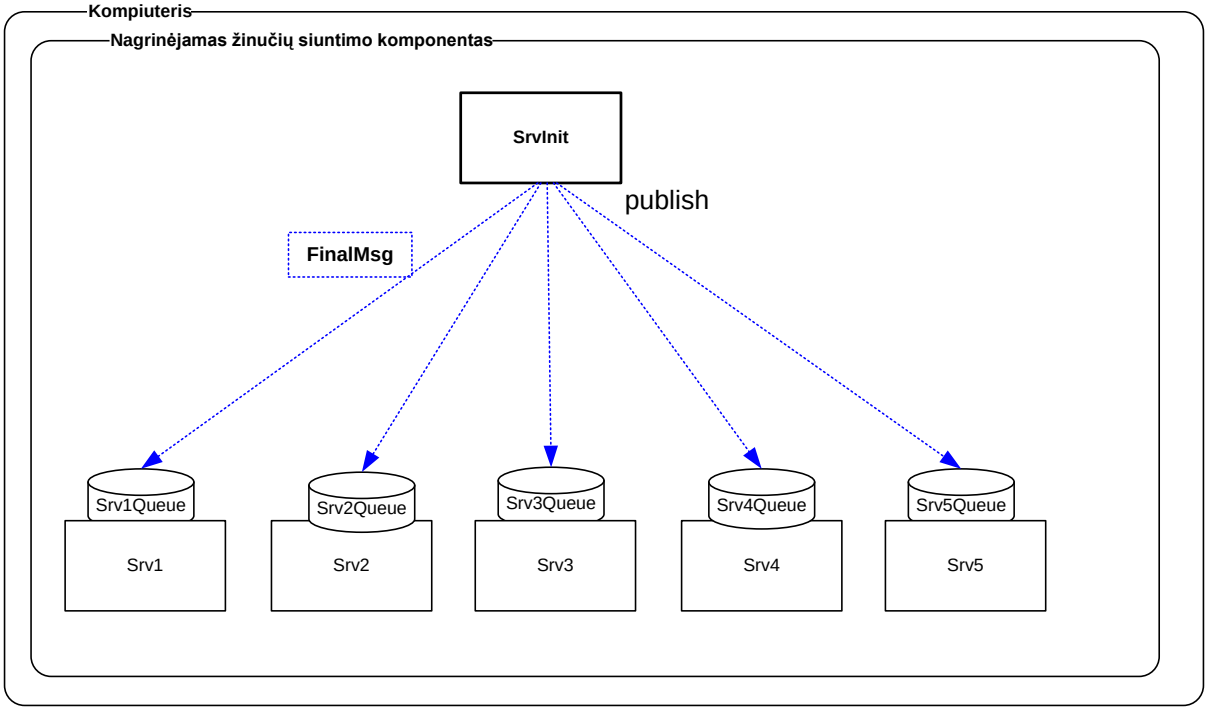
Lentelė 8. Servisų veiksmai kada vyksta užklausos/atsako siuntimas.

Servisas	**Funkcijos**	**Nagrinėjama eilė**	**Nagrinėjama žinutė**
SrvInit	Servisas sukuria pradines žinutes ir pradeda siuntimo operaciją, siųsdamas (Srv1Msg) servisui Srv1. Taip pat stebi eilę ProcMsgsQueue, jei joje yra žinučių, atspausdina jų turinį ekrane. Šioje eilėje atsiduria jau išnagrinėtos žinutės, kurios savyje turi su tyrimu susijusius rezultatus.	ProcMsgsQueue	FinalMsg
Srv1	Skaito eilės Srv1Queue žinutes. Kiekvienos žinutės parametrą `TimesProcessed` padidina vienetu. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Suformuoja naują žinutę ir siunčia kitam servisui (Srv2). Įvykus klaidai žinutė siunčiama i SrvError servisą (jei įjungtas klaidų imitavimas, kas nustatytą mazgų skaičių yra imituojama klaida). Jeigu parametro `TimesProcessed` reikšmė tampa didesne negu nustatyta (Lentelė 11, Bendras kiekvienos žinutės	Srv1Queue	Srv1Msg

	apdorojimo kiekis mazguose (servisuose)), žinutė konvertuojama į FinalMsg ir siunčiama į ProcMsgsQueue eilę.		
Srv2	Skaito eilės Srv2Queue žinutes. Kiekvienos žinutės parametrą `TimesProcessed` padidina vienetu. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Suformuoja naują žinutę ir siunčia kitam servisui (Srv3). Įvykus klaidai žinutė siunčiama i SrvError servisą (jei įjungtas klaidų imitavimas, kas nustatytą mazgų skaičių yra imituojama klaida). Jeigu parametro `TimesProcessed` reikšmė tampa didesne negu nustatyta (Lentelė 11, Bendras kiekvienos žinutės apdorojimo kiekis mazguose (servisuose)), žinutė konvertuojama į FinalMsg ir siunčiama į ProcMsgsQueue eilę.	Srv2Queue	Srv2Msg
Srv3	Skaito eilės Srv3Queue žinutes. Kiekvienos žinutės parametrą `TimesProcessed` padidina vienetu. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Suformuoja naują žinutę ir siunčia kitam servisui (Srv4). Įvykus klaidai žinutė siunčiama i SrvError servisą (jei įjungtas klaidų imitavimas, kas nustatytą mazgų skaičių yra imituojama klaida). Jeigu parametro `TimesProcessed` reikšmė tampa didesne negu nustatyta (Lentelė 11, Bendras kiekvienos žinutės apdorojimo kiekis mazguose (servisuose)),	Srv3Queue	Srv3Msg

	žinutė konvertuojama į FinalMsg ir siunčiama į ProcMsgsQueue eilę.		
Srv4	Skaito eilės Srv4Queue žinutes. Kiekvienos žinutės parametrą `TimesProcessed` padidina vienetu. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Suformuoja naują žinutę ir siunčia kitam servisui (Srv5). Įvykus klaidai žinutė siunčiama i SrvError servisą (jei įjungtas klaidų imitavimas, kas nustatytą mazgų skaičių yra imituojama klaida). Jeigu parametro `TimesProcessed` reikšmė tampa didesne negu nustatyta (Lentelė 11, Bendras kiekvienos žinutės apdorojimo kiekis mazguose (servisuose)), žinutė konvertuojama į FinalMsg ir siunčiama į ProcMsgsQueue eilę.	Srv4Queue	Srv4Msg
Srv5	Skaito eilės Srv5Queue žinutes. Kiekvienos žinutės parametrą `TimesProcessed` padidina vienetu. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Suformuoja naują žinutę ir siunčia kitam servisui (Srv1). Įvykus klaidai žinutė siunčiama i SrvError servisą (jei įjungtas klaidų imitavimas, kas nustatytą mazgų skaičių yra imituojama klaida). Jeigu parametro `TimesProcessed` reikšmė tampa didesne negu nustatyta (Lentelė 11, Bendras kiekvienos žinutės apdorojimo kiekis mazguose (servisuose)), žinutė konvertuojama į FinalMsg ir	Srv5Queue	Srv5Msg

	siunčiama į ProcMsgsQueue eilę.		
SrvError	Šis servisas skaito eilę Error. Paima iš ten žinutes, kiekvienos jos parametrą `TimesProcessed` padidina vienetu. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Konvertuoja į Srv1Msg formatą ir tiesiog persiunčia šias žinutes servisui Srv1 (į Srv1Queue).	Error	Visos



17 pav. Žinučių siuntimo ir servisų išdėstymo schema, kada nagrinėjamas transliavimo-prenumeravimo siuntimas.

Transliavimo-prenumeravimo tipo žinutės turinys

Laikas, kada žinutė išsiusta iš ankstesnio serviso.

Bendra trukmė, kuri nusako, kiek laiko žinutė buvo apdorojama ankstesniuose servisuose iki nagrinėjamo momento.

Nešami duomenys.

18 pav. Transliavimo-prenumeravimo tipo žinutės formatas.

Lentelė 9. Servisų veiksmai kada vyksta transliavimo-prenumeravimo siuntimas.

Servisas	**Funkcijos**	**Nagrinėjama eilė**	**Nagrinėjama žinutė**
SrvInit	Servisas sukuria pradines žinutes ir pradeda transliavimo operaciją, siųsdamas jas (FinalMsg formatu) visiems servisams kurie yra užsiprenumeravę, t.y. poaibiui Srv1, Srv2, Srv3, Srv4, Srv5.	ProcMsgsQueue	FinalMsg
Srv1	Skaito eilės Srv1Queue žinutes. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Galiausiai atspausdina žinučių turinį ekrane.	Srv1Queue	FinalMsg
Srv2	Skaito eilės Srv2Queue žinutes. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Galiausiai atspausdina žinučių turinį ekrane.	Srv2Queue	FinalMsg
Srv3	Skaito eilės Srv3Queue žinutes. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Galiausiai atspausdina žinučių turinį ekrane.	Srv3Queue	FinalMsg
Srv4	Skaito eilės Srv4Queue žinutes. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Galiausiai atspausdina žinučių turinį ekrane.	Srv4Queue	FinalMsg
Srv5	Skaito eilės Srv5Queue žinutes. Paskaičiuoja kiekvienos žinutės apdorojimo laiką šiame servise, įskaičiuodamas ir ankstesnių servisų apdorojimo laiką. Galiausiai atspausdina žinučių turinį ekrane.	Srv5Queue	FinalMsg

5.2 Tyrimo parametrai, techninė ir programinė įranga

Kai kurie iš šių komponentų siūlo labai daug papildomų funkcijų, todėl tyrimas atliktas, nagrinėjant svarbiausias komponento funkcijas – tas kurios aktualios magistrinio darbo sistemoje (žiūrėti skyrių 4.4 Reikalavimai komponentams).

Taigi nagrinėjami tokie siuntimo metodai: užklausos/atsako siuntimas, užklausos/atsako siuntimas su klaidų apdorojimu bei siuntimas transliavimo-prenumeravimo metodu.

Rezultatai – bendri žinučių apdorojimo servisuose ir eilėse laikai, priklausomai nuo žinučių skaičiaus, siuntimo metodo, klaidų apdorojimo. Kad kuo mažiau įtakos tyrimui darytų aplinkos veiksmai (tinklo sparta ir pan.), jis buvo atliktas naudojant vieną kompiuterį. Kiti su tyrimu susiję parametrai pateikti lentelėse.

Lentelė 10. Techninė ir programinė įranga, kuria naudojantis buvo atliktas tyrimas.

Techninė programinė įranga	**Parametrai**
Kompiuteris	Intel Core 2 Duo T7250 (2 GHz) CPU, 2 GB RAM.
Programinė įranga	MS Windows XP Professional SP3, MSMQ 3.0, Microsoft SQL Server 2005 Express Edition, Microsoft .NET Framework 3.5, nServiceBus 2.1.0.951, MassTransit 0.5.0.1927, Rhino Service Bus 1.8.0.0.

Lentelė 11. Fiksuoti parametrai.

Parametras	Vertė
Siunčiamos žinutės dydis (kb)	80
Bendras kiekvienos žinutės apdorojimo kiekis mazguose (servisuose)	50
Naudojamų servisų kiekis (įskaitant ir klaidų bei pradinį)	7
Aktyvių servisų kiekis (kuriuos naudojant yra atliekami matavimai)	5

Lentelė 12. Kintantys parametrai.

Parametras	Vertė, kas kiek matuota
Siunčiamų žinučių kiekis	10 – 100, 10
Klaidų apdorojimas	Išjungtas/Įjungtas
Užsiprenumeravusių aktyvių servisų kiekis (transliavimo-prenumeravimo siuntime)	1-5,1

5.3 Komponento nServiceBus tyrimo rezultatai

Norint įvertinti kaip komponentai dirba, pirmiausia buvo ištirtas Biznio sluoksnio karkase naudojamas komponentas – nServiceBus. Tada, pagal jo charakteristikas ir naudojantis tais pačiais metodais, įvertinti kiti nagrinėjami komponentai.

5.3.1 Žinučių siuntimo užklausos/atsako metodu tyrimas

Naudojantis komponentu nServiceBus ir žinučių siuntimo užklausos/atsako metodu, buvo atliktas tyrimas, kuris pateiktas lentelėse (Lentelė 13, Lentelė 14).

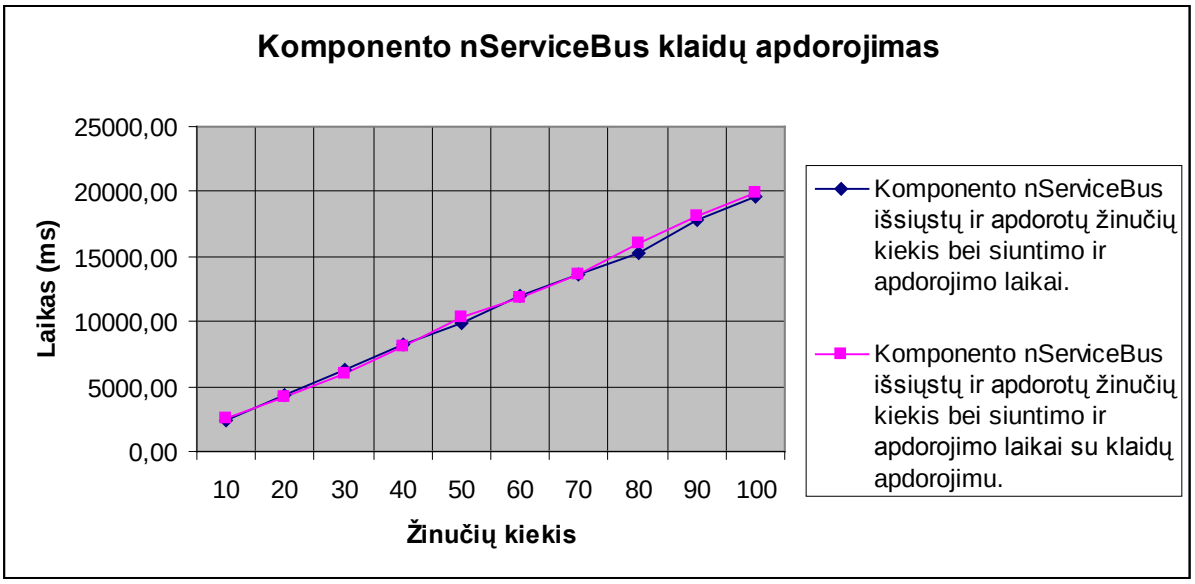
Lentelė 13. Komponento nServiceBus išsiųstų ir apdorotų žinučių kiekis bei siuntimo ir apdorojimo laikai.

Išsiųstų ir apdorotų žinučių kiekis	Vidutinis žinučių siuntimo (laukimo eilėse) ir apdorojimo servisuose laikas (ms)
10	2457,60
20	4349,50
30	6224,03
40	8221,85
50	9807,04
60	11930,33
70	13682,13
80	15297,23
90	17742,43
100	19541,33

Lentelė 14. Komponento nServiceBus išsiųstų ir apdorotų žinučių kiekis bei siuntimo ir apdorojimo laikai, su klaidos servise imitavimu.

Išsiųstų ir apdorotų žinučių kiekis	Vidutinis žinučių siuntimo (laukimo eilėse) ir apdorojimo servisuose laikas (ms)
10	2544,80
20	4177,70
30	6041,80
40	8117,53
50	10260,80
60	11851,83
70	13636,64
80	16035,35
90	18110,90
100	19839,24

Lentelėse (Lentelė 13 ir Lentelė 14) matome, kad kai žinučių, kurios siunčiamos ir apdorojamos mazge, kiekis artėja link begalybės, kiekvienos žinutės siuntimo ir apdorojimo laikas auga. Tai atsitinka dėl to, jog vis daugiau žinučių „laukia“ atitinkamų servisų eilėse. Laukimas vyksta, nes per tam tikrą laiką servisai nespėja šių žinučių nuskaityti iš atitinkamų eilių. Matome, jog lentelėje (Lentelė 14) žinutės siuntimo ir apdorojimo laikai šiek tiek didesni, tai yra dėl to, kad klaidos apdorojimo procedūra reikalauja papildomu resursų.



19 pav. nServiceBus veikimas imituojant klaidas ir palyginimas su normaliu veikimu.

19 pav. matome, jog nServiceBus veikimas su klaidų imitavimu lyginant su normaliu veikimu skiriasi tik nežymiai. Tai yra todėl, jog klaidų apdorotojai veikia labai greitai. Tačiau, jeigu procedūra, kuri apdoroja klaidą, būtų sudėtingesnė, tada raudona kreivė galėtų žymiai skirtis, t.y. būtų daug aukščiau.

5.3.2 Žinučių siuntimo transliavimo-prenumeravimo metodu tyrimas

Naudojantis nServiceBus komponentu ir žinučių siuntimo transliavimo-prenumeravimo metodu, buvo atliktas tyrimas, kuris pateiktas kitoje lentelėje (Lentelė 15).

Lentelė 15. Komponento nServiceBus žinučių transliavimo ir apdorojimo servisuose laikas, priklausomai nuo prenumeratorių skaičiaus.

Veikiančių prenumeratorių (servisų) kiekis	Vidutinis žinučių transliavimo ir apdorojimo servisuose laikas (ms)
1	125,50
2	54,66
3	46,72
4	24,17
5	22,52

Lentelė 15 matome, kad kai prenumeratorių skaičius auga, transliavimas tampa efektyvesniu. Taigi, galima spręsti, kad šis komponentas transliavimui, lyginant su užklausos/atsako siuntimu, naudoja visai kitokį algoritmą.

5.4 Komponento MassTransit tyrimo rezultatai

5.4.1 Žinučių siuntimo užklausos/atsako metodu tyrimas

Naudojantis komponentu MassTransit ir žinučių siuntimo užklausos/atsako metodu, buvo atliktas tyrimas, kuris pateiktas lentelėse (Lentelė 16, Lentelė 17).

Lentelė 16. Komponento MassTransit išsiųstų ir apdorotų žinučių kiekis bei siuntimo ir apdorojimo laikai.

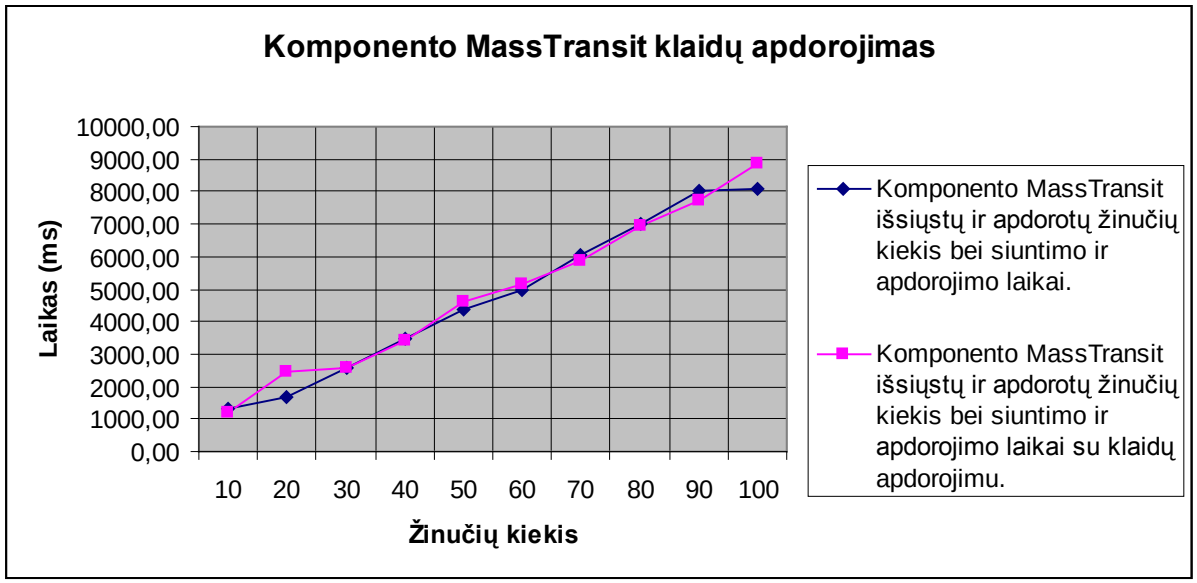
Išsiųstų ir apdorotų žinučių kiekis	Vidutinis žinučių siuntimo (laukimo eilėse) ir apdorojimo servisuose laikas (ms)
10	1294,70
20	1665,95
30	2581,43
40	3469,83
50	4385,52
60	4978,70
70	6053,60

80	6995,03
90	8003,08
100	8113,69

Lentelė 17. Komponento MassTransit išsiųstų ir apdorotų žinučių kiekis bei siuntimo ir apdorojimo laikai, su klaidos servise imitavimu.

Išsiųstų ir apdorotų žinučių kiekis	**Vidutinis žinučių siuntimo (laukimo eilėse) ir apdorojimo servisuose laikas (ms)**
10	1204,50
20	2434,70
30	2563,07
40	3394,03
50	4636,44
60	5156,83
70	5881,44
80	6922,23
90	7750,21
100	8853,04

Šios lentelės bus nagrinėjamos kitame skyriuje (5.6 Išnagrinėtų trečių šalių komponentų, skirtų žinučių, naudojančių eiles, siuntimui, palyginimas ir įvertinimas).



20 pav. MassTransit veikimas imituojant klaidas ir palyginimas su normaliu veikimu.

20 pav. matome, jog MassTransit veikimas su klaidų imitavimu lyginant su normaliu veikimu skiriasi tik nežymiai. Tai yra todėl, jog lyginant su žinučių siuntimo algoritmais MassTransit klaidų apdorotojai veikia labai greitai. Tačiau, jeigu procedūra, kuri apdoroja klaidą būtų sudėtinga, raudona kreivė, priklausomai nuo procedūros sudėtingumo, galėtų žymiai skirtis, t.y. būtų daug aukščiau.

5.4.2 Žinučių siuntimo transliavimo-prenumeravimo metodu tyrimas

Naudojantis komponentu MassTransit ir žinučių siuntimo transliavimo-prenumeravimo metodu, buvo atliktas tyrimas, kuris pateiktas kitoje lentelėje (Lentelė 18).

Lentelė 18. Komponento MassTransit žinučių transliavimo ir apdorojimo servisuose laikas, priklausomai nuo prenumeratorių skaičiaus.

Veikiančių prenumeratorių (servisų) kiekis	Vidutinis žinučių transliavimo ir apdorojimo servisuose laikas (ms)
1	6,08
2	7,44
3	11,29
4	12,03
5	16,47

Lentelė 18 matome, kad transliavimo efektyvumas atvirkščiai proporcingas prenumeratorių skaičiui. Taigi galima spręsti, kad šis komponentas transliavimui ir užklausos/atsako siuntimui naudoja panašius algoritmus.

5.5 Komponento Rhino Service Bus tyrimo rezultatai

5.5.1 Žinučių siuntimo užklausos/atsako metodu tyrimas

Naudojantis komponentu Rhino Service Bus ir žinučių siuntimo užklausos/atsako metodu, buvo atliktas tyrimas, kuris pateiktas lentelėse (Lentelė 19, Lentelė 20).

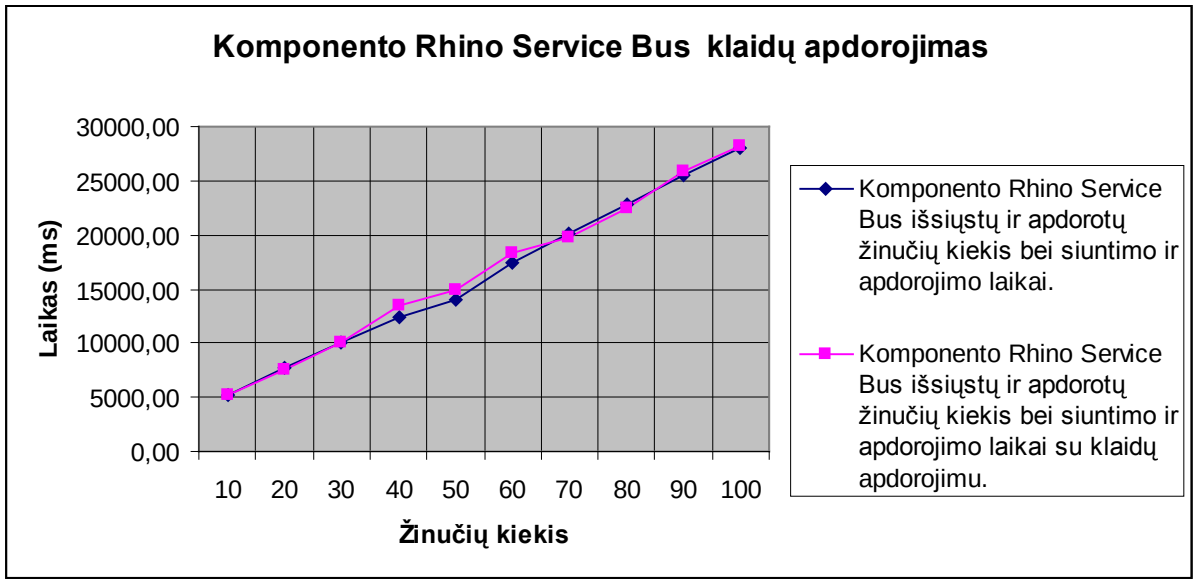
Lentelė 19. Komponento Rhino Service Bus išsiųstų ir apdorotų žinučių kiekis bei siuntimo ir apdorojimo laikai.

Išsiųstų ir apdorotų žinučių kiekis	**Vidutinis žinučių siuntimo (laukimo eilėse) ir apdorojimo servisuose laikas (ms)**
10	5276,10
20	7694,75
30	10060,47
40	12484,23
50	14070,52
60	17348,50
70	20199,81
80	22766,14
90	25486,93
100	27974,53

Lentelė 20. Komponento Rhino Service Bus išsiųstų ir apdorotų žinučių kiekis bei siuntimo ir apdorojimo laikai, su klaidos servise imitavimu.

Išsiųstų ir apdorotų žinučių kiekis	Vidutinis žinučių siuntimo (laukimo eilėse) ir apdorojimo servisuose laikas (ms)
10	5257,30
20	7545,70
30	10074,53
40	13433,50
50	14871,42
60	18349,88
70	19695,61
80	22498,61
90	25816,93
100	28177,50

Šios lentelės bus nagrinėjamos kitame skyriuje (5.6 Išnagrinėtų trečių šalių komponentų, skirtų žinučių, naudojančių eiles, siuntimui, palyginimas ir įvertinimas).



21 pav. Rhino Service Bus veikimas imituojant klaidas ir palyginimas su normaliu veikimu.

21 pav. matome, jog Rhino Service veikimas su klaidų imitavimu lyginant su normaliu veikimu skiriasi tik nežymiai. Tai yra todėl, jog lyginant su žinučių siuntimo algoritmais

Rhino Service Bus klaidų apdorotojai veikia labai greitai. Tačiau, jeigu procedūra, kuri apdoroja klaidą būtų sudėtinga, raudona kreivė, priklausomai nuo procedūros sudėtingumo galėtų žymiai skirtis, t.y. būtų daug aukščiau.

5.5.2 Žinučių siuntimo transliavimo-prenumeravimo metodu tyrimas

Naudojantis komponentu Rhino Service Bus ir žinučių siuntimo transliavimo-prenumeravimo metodu, buvo atliktas tyrimas, kuris pateiktas kitoje lentelėje (Lentelė 21).

Lentelė 21. Komponento Rhino Service Bus žinučių transliavimo ir apdorojimo servisuose laikas, priklausomai nuo prenumeratorių skaičiaus.

Veikiančių prenumeratorių (servisų) kiekis	**Vidutinis žinučių transliavimo ir apdorojimo servisuose laikas (ms)**
1	21,26
2	15,60
3	19,81
4	25,79
5	32,28

Šioje lentelėje (Lentelė 21) galime matyti, kad transliavimo efektyvumas atvirkščiai proporcingas prenumeratorių skaičiui. Taigi galima spręsti, kad šis komponentas transliavimui ir užklausos/atsako siuntimui naudoja panašius algoritmus. Taip pat matome, jog esant vienam prenumeratoriui, transliavimo laikas žymiai išauga. Tai gali atsitikti dėl to, jog transliavimo mechanizmas vienam prenumeratoriui nėra numatytas. Vietoj tokio mechanizmo reiktų naudoti užklausos/atsako metodą.

5.6 Išnagrinėtų trečių šalių komponentų, skirtų žinučių, naudojančių eiles, siuntimui, palyginimas ir įvertinimas

Šiame skyriuje bus įvertinti, ištirtų trečių šalių komponentų, kurie skirti bendravimui tarp servisų panaudojant žinutes, tyrimo rezultatai. Vertinimui bus naudojami grafikai, kurie atspindi žinučių siuntimo efektyvumą (22 pav., 23 pav., 24 pav.) , taip pat kiekvieno komponento naudojimo ir konfigūravimo paprastumas, dokumentacijos pasiūla.

5.6.1 Komponentų naudojimo ir konfigūravimo paprastumas, dokumentacijos pasiūla

Diegiant ir naudojant minėtus komponentus buvo susidurta su dokumentacijos trūkumu. Gausiausiai dokumentuotas su daug naudojimo pavyzdžių buvo nServiceBus. Autorius nuolat papildo šio komponento dokumentaciją. Mažiau dokumentuoti yra MassTransit ir Rhino Service Bus, todėl teko daug eksperimentuoti, kol pavyko juos įdiegti ir „priversti veikti“. Nors autorius teigia, jog Rhino Service Bus yra supaprastintas komponentas, dėl dokumentacijos trūkumo taip nėra. Be to MassTransit papildomai naudoja įvairius pagalbinius servisus ir duomenų bazę, todėl norint jį paleisti, reikia įsigilinti į jo ir šių servisų veikimą, o neturint dokumentacijos tai padaryt gana sunku. Dėl šios priežasties buvo nagrinėjamas MassTransit komponento išeities tekstas. Taigi lengviausiai įdiegiamas ir naudojamas yra nServiceBus komponentas.

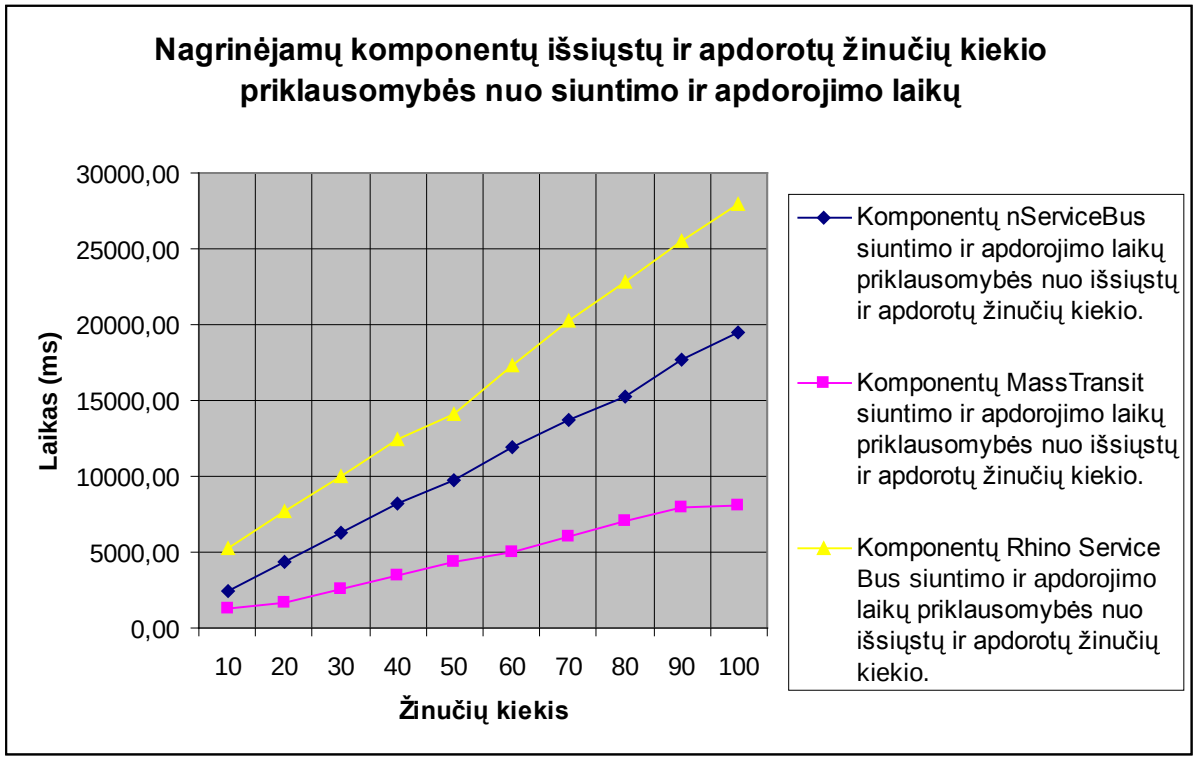
Konfigūruoti sudėtingiausia buvo MassTransit komponentą, dėl naudojamų pagalbinių servisų ir duomenų bazės. Lengviausiai - Rhino Service Bus, kadangi jis turi labai paprastus konfigūracinius failus ir nServiceBus.

Taigi galima daryti išvada, jog nServiceBus yra subrandintas ir dokumentuotas komponentas, todėl vartotojui nesunku jį įdiegti, konfigūruoti ir naudoti.

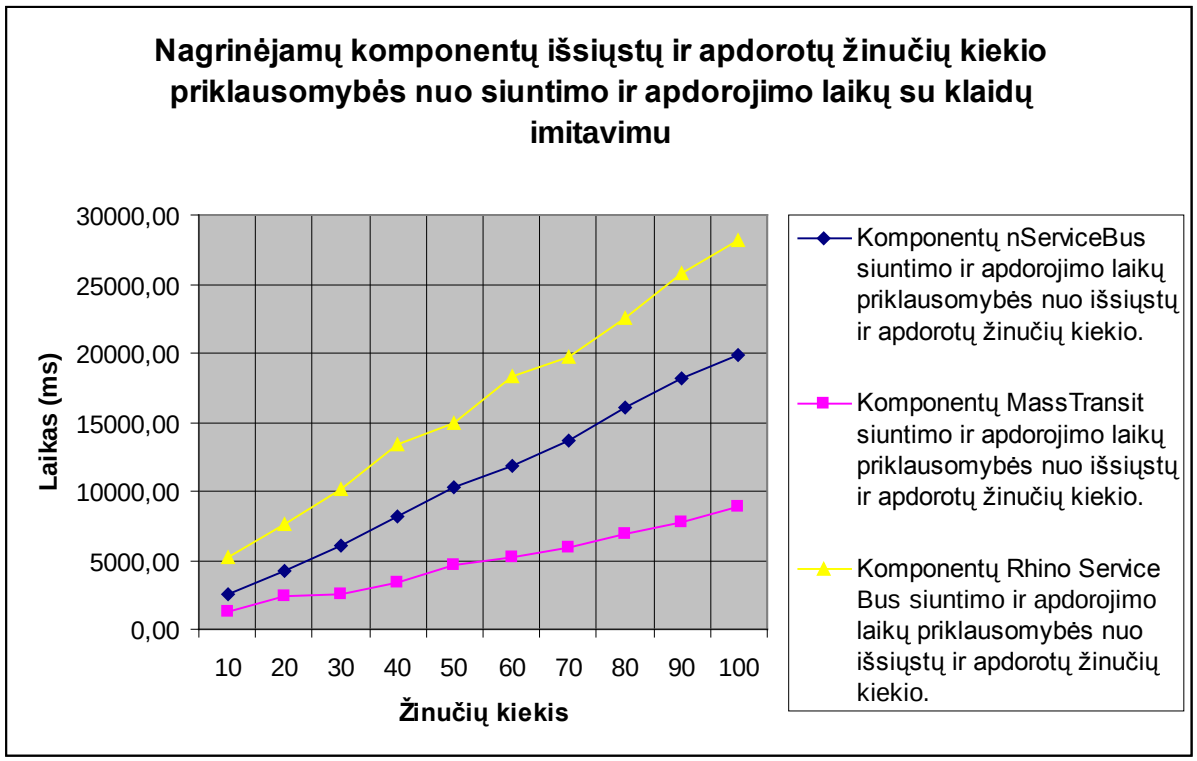
Taip pat reikia paminėti, jog šie komponentai siūlo daug daugiau galimybių negu buvo ištirtos šio magistrinio darbo metu. Dėl informacijos trūkumo jas visas perprasti taip pat yra gana sunku.

5.6.2 Ištirtų trečių šalių komponentų žinučių siuntimo efektyvumas

22 pav. ir 23 pav. matome, jog, naudojantis užklausos/atsako siuntimo metodu, greičiausiai veikė MassTransit, toliau nServiceBus ir Rhino Service Bus komponentai. MassTransit naudoja duomenų bazę ir atitinkamus pagalbinius servisus, kurie turi teigiamą įtaką siuntimo efektyvumui. nServiceBus bei Rhino Service Bus pagalbinių servisų neturi, viskas veikia tuose pačiuose procesuose, kaip ir nagrinėti žinučių siuntimo servisai. Priklausomybė yra tiesinė, nes kuo daugiau žinučių iššiunčiama, tuo daugiau jų, kol galės būti apdorotos, laukia eilėje (didėja siuntimo ir apdorojimo laikas). Efektyvumo didininimui dažnai pasitelkiamos proceso gijos, tačiau kadangi žinutėms saugoti yra naudojama eile paremta duomenų struktūra, didinant gijų skaičiu mes didesnio efektyvumo nepasiektume.

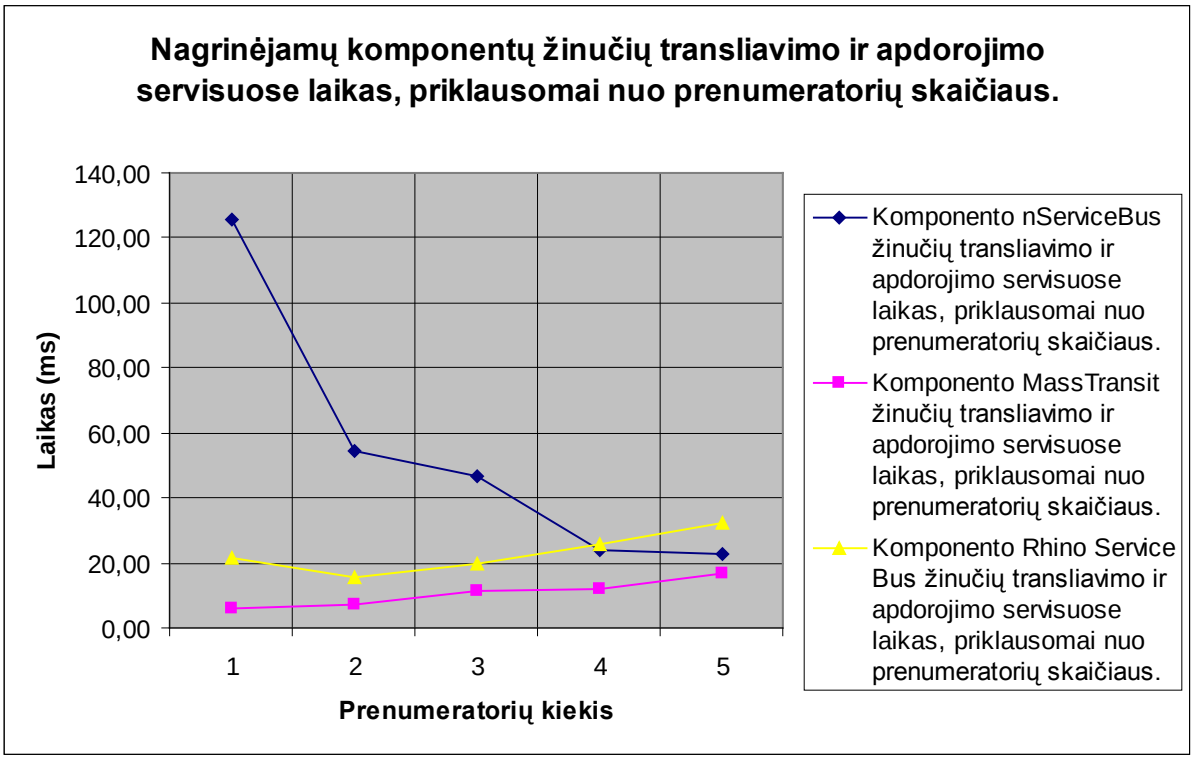


22 pav. Komponentų nServiceBus, MassTransit, Rhino Service Bus vykdyto siuntimo užklausos/atsako metodu palyginimas.



23 pav. Komponentų nServiceBus, MassTransit, Rhino Service Bus vykdyto siuntimo užklausos/atsako metodu su klaidų imitavimu palyginimas.

24 pav. matome, jog, naudojantis transliavimo-prenumeravimo siuntimo metodu, greičiausiai veikė MassTransit komponentas. Tačiau tokio tipo siuntimas (transliavimo-prenumeravimo) aktualus tada, kai servisų (prenumeratorių) skaičius yra didelis. Šiame grafike, taip pat matome, jog didėjant prenumeratorių skaičiui nServiceBus transliavimo ir apdorojimo servisuose laikas turi tendenciją mažėti, kol pasiekia įsisotinimo ribą. Kitų komponentų transliavimo ir apdorojimo servisuose laikas, didėjant prenumeratorių skaičiui, auga tiesiniu dėsniu. Todėl, esant didesniam prenumeratorių skaičiui, labiausiai tinka nServiceBus komponentas, esant vidutiniam – MassTransit. Kada prenumeratorių skaičius yra artimas vienetui, šis siuntimo metodas netinkamas, tokiu atveju reiktų naudoti įprastą užklausos/atsako siuntimą.



24 pav. Komponentų nServiceBus, MassTransit, Rhino Service Bus vykdyto siuntimo transliavimo-prenumeravimo metodu palyginimas.

Gauti rezultatai lyginti su Biznio sluoksnio karkase naudojamo komponento rezultatais (nServiceBus), padarytos išvados, pagal kurias pateikiama rekomendacinė lentelė (Lentelė 22).

Lentelė 22. Ištirtų komponentų naudojimo rekomendacijos.

		Komponento naudojimo paprastumas	
		Žemas	**Vidutinis**
Žinučių siuntimo ir apdorojimo greitis	**Aukštas**	MassTransit	nServiceBus
	Žemas	Rhino Service Bus	nServiceBus

Pastaba: tyrimo modelis buvo sudarytas ir tyrimas buvo atliktas siauroje srityje (5.1 Tyrimo kontekstas ir tyrimo metodai), todėl lentelė yra neuniversali ir tinka tik komponentų versijoms, kurios naudotos tyrimo metu (Lentelė 10. Techninė ir programinė įranga, kuria naudojantis buvo atliktas tyrimas.).

6 IŠVADOS

Šiuolaikinio verslo sektorius yra labai sudėtingas ir apima tokius uždavinius: daugybės programinės ir kompiuterinės įrangos naudojimą skirtingose platformose, internetinių išskaidytų sistemų bendravimą, verslo sistemų išplėtimą, palaikymą ir integravimą, todėl magistriniame darbe nagrinėjama sritis - verslo sistemos. Šiuos uždavinius padeda spręsti metodai ir technologijos apibrėžtos šios darbo analitinėje dalyje: paslaugomis besiremianti architektūra, Windows operacinės sistemos komunikacijų pamatas (WCF), metodai, skirti bendravimui tarp paskirstytų sistemų dalių. Magistrinio projekto sistema turi visas verslo programinės įrangos savybes. Ji buvo kurta su tikslu minėtoms sistemoms teikti bendras, standartizuotas funkcijas.

Paskirstytos programinės įrangos dalys tarpusavyje bendrauja naudodamos įvairius sprendimus, tame tarpe ir nuotolinių procedūrų kvietimo technologijas bei žinutes. Žinučių sąvoka ir jų siuntimo metodologijos taip pat buvo paaiškintos šio darbo analitinėje dalyje.

Šiame magistriniame darbe nagrinėjamas žinutėmis paremtas bendravimas tarp servisų. Išnagrinėta žinučių technologija, kuri yra naudojama Biznio sluoksnio karkase (Magistrinis projektas). Karkasas, šiai technologijai įgyvendinti, naudoja trečių šalių komponentą nServiceBus. Paskirstytos sistemos ir trečių šalių komponentai turi daug problemų, kurios šiame darbe buvo apibrėžtos. Remiantis trečių šalių komponentų problemomis suformuluoti tiriamojo darbo tikslai. Nuspręsta išbandyti ir palyginti kitus, rinkoje esančius, trečių šalių žinučių siuntimo komponentus. Pasirinktos nemokamos alternatyvos, kurios labiausiai tiko Biznio sluoksnio karkaso kontekste: MassTransit ir Rhino Service Bus.

Sudarytas tyrimo modelis, kuris apėmė dvi servisų išdėstymo ir jungimo schemas (5.1 Tyrimo kontekstas ir tyrimo metodai). Šios schemos atitinkamai apima du žinučių siuntimo metodus: siuntimą užklausos/atsako metodu ir siuntimą transliavimo-prenumeravimo metodu. Pagal šias servisų išdėstymo schemas, naudojant minėtus siuntimo metodus ir trečių šalių komponentus, buvo vykdomas žinučių siuntimas ir matuojama:

- Naudojant pirmą servisų išdėstymo schemą, buvo skaičiuojamas vidutinis žinučių siuntimo ir apdorojimo servisuose laikas priklausomai nuo žinučių kiekio ir nagrinėjamo komponento. Matavimai buvo

vykdomi dviem etapais: vykdant normalų siuntimą ir vykdant siuntimą su klaidų komponentuose imitavimu.

- Naudojant antrą servisų išdėstymo schemą, buvo skaičiuojamas vidutinis žinučių siuntimo ir apdorojimo servisuose laikas priklausomai nuo prenumeratorių kiekio ir nagrinėjamo komponento.

Matavimo rezultatai pateikti grafiškai. Taip pat įvertintas šių komponentų naudojimo paprastumas, kuris apima: diegimą, konfigūravimą ir naudojimą. Galiausiai visų trijų komponentų matavimų rezultatai ir naudojimo paprastumas palyginti, aptarti ir apibendrinti. Naudojant minėtus rezultatus sudaryta trečių šalių komponentų rekomendacinė lentelė (Lentelė 22).

Šioje lentelėje matome, jog tuomet, kai reikia aukšto žinučių siuntimo ir apdorojimo greičio (tai yra pagrindinis parametras) geriau naudoti MassTransit. Kuomet svarbesnis parametras yra programos sukūrimo laikas bei jos patikimumas (kuomet norima greitai ir kokybiškai sukurti sistemą) geriau naudoti nServiceBus. Be to nServiceBus laikomas subrandintu trečių šalių komponentu ir yra naudojamas kritinėse sistemose.

7 LITERATŪRA

[1] TANNENBAUM, A. *Distributed Systems: Paradigms and Principles*. 2002. 803 p. ISBN: 0130888931.

[2] REESE, G. Database Programming with JDBC and Java. 2000. 328 p. ISBN: 1565926161.

[3] TURTSCHI, A.; WERRY, J. *Develop and Deliver Enterprise-Critical Desktop and Web Applications with C# .NET*. Rockland, 2005. 800 p. ISBN: 1928994504.

[4] BAHREE, A., et al. *Pro WCF*: *Practical Microsoft SOA Implementation*. New York, 2007. 512 p. ISBN: 1590597028.

[5] THOMAS, S.; HARTMAN, J.; RADIN, J. IRIX Network Programming Guide. 2003, [žiūrėta 2010-05-23]. Prieiga internete: <http://techpubs.sgi.com/library/tpl/cgi-bin/getdoc.cgi?coll=0650&db=bks&srch=&fname=/SGI_Developer/IRIX_NetPG/sgi_html/ch04.html>.

[6] ERL, T. *Service-Oriented Architecture: Concepts, Technology, and Design.* New York, 2005. 792 p. ISBN: 0131858580.

[7] HASAN, J.; DURAN, M. *Expert Service-Oriented Architecture in C#.* 2-asis patais. ir papild. leid. Berkeley, 2005. 305 p. ISBN: 1590593901.

[8] CHAPPELL, D. *Enterprise Service Bus: Theory in Practice*. Sebastopol, 2004. 352 p. ISBN: 0596006756.

[9] HANSON, Jeff. *Event-driven services in SOA.* Javaworld. 2005, sausis [žiūrėta 2010-05-23]. Prieiga internete: <http://www.javaworld.com/javaworld/jw-01-2005/jw-0131-soa.html>.

[10] BOYD, Scot; COSTALL, Richard. *Pro MSMQ: Microsoft Message Queue Programming*. Berkeley, 2004. 448 p. ISBN: 1590593464.

[11] HOUSTON, P. Building Distributed Applications with Message Queuing Middleware. [žiūrėta 2010-05-23]. Prieiga internete: <http://technet.microsoft.com/en-us/library/cc723260.aspx>.

[12] DAHAN, Udi. nServicebus. Architectural Principles. [žiūrėta 2010-05-23]. Prieiga internete: <http://www.nservicebus.com/ArchitecturalPrinciples.aspx>.

[13] ŠILANSKAS, Žygimantas; GOGYS, Donatas. *BIZNIO SLUOKSNIO KARKASO, BENDRO KURIAMOMS PROGRAMOMS, SUKŪRIMAS IR TYRIMAS*: magistro projektinis darbas. KTU, Informatikos fakultetas. [Kaunas], 2010. 241 p.

[14] FOWLER, Martin; RICE, David. *Patterns of Enterprise Application Architecture*. Boston, 2002. 560 p. ISBN: 0321127420.

[15] DAHAN, Udi. nServicebus Overview. [žiūrėta 2010-05-24]. Prieiga internete: <http://www.nservicebus.com/Overview.aspx>.

8 TERMINŲ IR SANTRUMPŲ ŽODYNAS

Terminas/santrumpa	Paaiškinimas
Biznio sluoksnio karkasas	Magistrinis projektas.
MS	Microsoft trumpinys.
MSMQ (Microsoft message queue)	MS sukurtas žinučių naudojančių eiles protokolas.
XML	Praplečiamoji žymų kalba, savo specifika panaši į HTTP, tik bendresnio tipo.
SOA	I servisus orientuota architektūra. Kada paskirstyta sistema stengiamasi modeliuoti komponentais, kurie turi didelę pakartotinio panaudojimo vertę – servisais.
WCF	Windows operacinės sistemos komunikacijų pamatas, skirtas kurti paskirstytas sistemas. MS .NET karkaso dalis.
API (Application Programming Interface)	Tai sąsaja, kurią suteikia operacinės sistemos, funkcijų bibliotekos, programos tam, kad programuotojas kurdamas sistemą galėtų pasiekti jų funkcionalumus.
IIS	Tai aibė Internetinių servisų (žiniatinklio, pašto, failų persiuntimo) skirtų Windows operacinei sistemai.
Servisas	Tam tikrą paslaugą teikianti programa (programos komponentas). Darbe laikysime, kad vienas servisas veikia vienam serveryje. Kai kur minima: sistema, komponentas, servisas. Šie terminai šiame darbe yra traktuojami kaip sinonimai.
Trečių šalių komponentas	Komponentas, kuris sukurtas trečių šalių ir naudojamas sistemoje kaip papildoma biblioteka. Šiame darbe trečių šalių komponentas nėra tapatinamas su servisu.

Communication between services using third party components

Summary

Nowadays a lot of systems are residing in distributed environments. These systems are composed of the client side software and server side software. That is called distributed systems architecture. Those types of systems can contain many services and many clients and may be substituted of many databases and many services. Distributed software components do not always run at the same time, networks, especially wide-area networks, are not always available and reliable. These parts communicate using remote procedure calls, or request/response messages and message queues.

In this work we research third party components which are used to communicate between parts of distributed systems using messages and Microsoft message queues. The research is performed on two chosen components, which best fits in the developed system's context. The choice of components and research based on usability problems using developed system called Business layer framework. The obtained results evaluated using user recommendation form.