KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Tomas Tamašauskas

Elektroninių paslaugų kardiologijoje realizavimo tyrimas

Magistro darbas

Darbo vadovas
dr. doc. Vytautas Pilkauskas

Kaunas, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
prof. habil. dr. Henrikas Pranevičius
2005 05

Elektroninių paslaugų kardiologijoje realizavimo tyrimas

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių k. katedros lektorė
dr. J. Mikelionienė
2005 05

Vadovas
dr. doc. Vytautas Pilkauskas
2005 05

Recenzentas
.....
2005 05

Atliko
IFM-9/1 gr. stud.
T. Tamašauskas
2005 05

Kaunas, 2005

TURINYS

SUMMARY

This work analyzes the possible ways for implementing electronical services in cardiology. There are many technologies that can be used to implement electronical services. In this case the WEB page generation and WEB services technology was chosen. There was no appropriate design pattern that matches the requirements. Front Controller is one of the possible problem solutions, but it does not describe any way of distributing the system. Front Controller design pattern is not an appropriate solution when there are huge calculations to be done and expected user count will be growing constantly. Broker design pattern solves this problem, but it has its disadvantage, that every server needs to have a real IP address. One of the constraints in this work is that server does not need to have an external IP address.

The proposed broker design pattern implements task distribution through many servers solves the problem that every server does not need external IP address. The system also solves the aviliability problem, so it can handle huge number of users.

Theoretical experiments proved that, when the number of users using the system is low, the Front Controller approach is more effective, but when the number of users grows at some point this approach can not handle all the user requests. In this case the proposed Broker design pattern approach solves the problem.

The proposed broker design patter was successfully implemented in real life and has been tested to be functional.

ĮVADAS

Šio darbo pagrindinis objektas yra elektroninės paslaugos ir jų kūrimas. Elektroninė paslauga yra ta paslauga, kuri gaunama per tinklą ir kuri įvykdo užduotis, sprendžia problemas arba atlieka duomenų perdavimus. Elektronines paslaugas gali vartoti žmonės, verslas ir kitos elektroninės paslaugos, jos gali būti pasiekiamos įvairiais informacijos įtaisais.

Elektroninių paslaugų kūrimas yra besivystanti disciplina. Jų kūrimas skiriasi nuo tradicinės programinės įrangos kūrimo. Šiame procese galima naudoti daugelį tradicinių programinės įrangos kūrimo metodikų, tačiau atsirandant naujoms technologijoms bei plečiantis galimybėms iškyla nauji aspektai į kuriuos reikia atsižvelgti.

Paskutiniu metu elektroninių paslaugų įvairovė smarkiai išaugo, dėl jų paklausos sparčiai besiplečiančiame interneto tinkle. Tai galima paaiškinti tuo, jog elektroninės paslaugos tiekimo išlaidos yra daug mažesnės nei realaus pasaulio atitikmenų, pavyzdžiui, elektroninio pašto siuntimas yra žymiai pigesnis nei įprastinio laiško. Augant elektroninių paslaugų vartotojų skaičiui labai svarbūs pasidarė ekonominiai bei elektroninės paslaugos tiekimo spartos aspektai. Naudojantis jau žinoma elektroninių paslaugų kūrimo praktika, galima sukurti efektyviai veikiančias elektronines paslaugas, tačiau plečiantis paslaugų vartotojų skaičiui pasiekiama riba, kai šio efektyvumo nebepakanka. Taip atsitikus yra dvi galimybės – tobulinti programinę įrangą arba plėsti sistemos techninius pajėgumus. Reikia sukurti būdą, kaip suprojektuoti sistemą, kad būtų galimi abudu tobulinimo būdai. Vienas iš būdų - naudoti superkompiuterius ir jų programinę įrangą, tačiau ši galimybė tinkama tik labai populiarioms elektroninėms paslaugoms, kurios sugebėtų atpirkti dideles investicijas. Pastaruoju metu vis populiaresnė darosi paskirstytųjų skaičiavimų, panaudojant paprastus kompiuterius, idėja. Idėja graži, tačiau iškyla daug techninių apribojimų, trukdančių realizuoti šio tipo sistemas, keletas iš jų: ugniasienės, kompiuteriai neturintys realių IP adresų. Šiuos apribojimus galima įveikti, tam reikia naudoti specifinį elektroninių paslaugų projektavimo būdą. Šiame darbe bus nagrinėjami galimi elektroninės paslaugos projektavimo variantai, kaip rezultatas bus pateiktas projektavimo šablonas bei jo tinkamumo analizė.

Problema. Elektroninių paslaugų kūrimas reikalauja išskirtinai geros architektūros, kuri leistų neribojamai plėsti sistemos našumą atsižvelgiant į didėjantį vartotojų skaičių. Taip pat sistemos plėtra turi būti įgyvendinama minimaliais kaštais. Taip pat sistema turi garantuoti algoritmų, naudojamų duomenų apdorojimui ar analizei, slaptumą.

Pagrindinis uždavinys. Pasiūlyti projektavimo šabloną, kuris apibendrintų, padėtų minimizuoti problemas, su kuriomis susiduriama kuriant ir teikiant elektronines paslaugas. Kadangi projektavimo šablonų gali būti keletas, reikia numatyti kriterijus, pagal kuriuos bus nustatytas jų tinkamumas.

Tikslai:

- Išanalizuoti elektroninių paslaugų ypatybes;
- Susipažinti su egzistuojančiomis elektroninėmis paslaugomis bei jų projektavimo šablonais;
- Pasiūlyti savo projektavimo šabloną;
- Nustatyti kriterijus, pagal kuriuos įvertinti šablono tinkamumą;
- Sukurti programą pasiūlyto projektavimo šablono veikimui patikrinti;
- Patikrinti projektavimo šablono veikimą eksperimentu.

Darbo struktūra.

- **Pirmoji dalis - analitinė.** Analitinėje dalyje analizuojamos elektroninių paslaugų ypatybės, egzistuojančios elektroninės paslaugos, egzistuojantys projektavimo šablonai bei suformuojamos pradinės uždavinio sprendimo idėjos.
- **Antroji dalis - teorinė.** Čia pateikiami projektavimo šablonai, aprašomi jų modeliai. Taip pat identifikuojami kriterijai, bei atliekamas teorinis projektavimo šablonų tinkamumo įvertinimas.
- **Trečioji dalis - eksperimentinė.** Eksperimentinėje dalyje aprašomas įvykdytas eksperimentas. Tai atliktas tyrimas, ar naudojantis pasiūlytu šablonu, galima sukurti realią elektroninę paslaugą. Šioje dalyje analizuojama kardiologinius tyrimus atliekanti elektroninė paslauga, įgyvendinanti automatizuotą kardiogramų tyrimą.

1 ELEKTRONINIŲ PASLAUGŲ ANALIZĖ

1.1 Elektroninės paslaugos

Elektroninė paslauga yra ta paslauga, kuri gaunama per tinklą ir kuri įvykdo užduotis, sprendžia problemas arba atlieka duomenų perdavimus. Elektronines paslaugas gali vartoti žmonės, verslas ir kitos elektroninės paslaugos, jos gali būti pasiekiamos įvairiais informacijos įtaisais.

1.1.1 Esamų elektroninių paslaugų apžvalga

Populiariausia elektroninių paslaugų tiekimo terpė yra internetas. Elektroninės paslaugos taip pat teikiamos mobiliojo ryšio ar kitais komunikacijų tinklais. Šiame darbe toliau bus nagrinėjamos internetinės elektroninės paslaugos.

Plačiausiai žinomos elektroninės paslaugos yra elektroninis pašas, klasifikuoti skelbimai, orų prognozės internetu ir daugelis kitų. Šios paslaugos yra viešai pasiekiamos, dažniausiai nemokamos, skirtos viešam vartotojui. Tačiau yra ir tokių elektroninių paslaugų, kurios teikiamos tik uždaram vartotojų ratui. Šios paslaugos dažniausiai būna mokamos arba turi kitų apribojimų, pavyzdžiui banko sąskaitų tvarkymas internetu. Paminėtos paslaugos yra pasiekiamos naudojantis interneto naršykle ir jų tiesioginis vartotojas yra žmogus.

Elektroninės paslaugos gali būti tiekiamos ne tik tiesiogiai žmonėms, tačiau ir specializuotoms informacinėms sistemoms, kurios gali naudotis keletu elektroninių paslaugų ir pateikti apdorotą bei sujungtą informaciją. Elektronines paslaugas gali vartoti ir kitos elektroninės paslaugos. Šios elektroninės paslaugos pasiekiamos naudojantis žiniatinklio paslaugomis (*WEB services*), CORBA ar kitomis technologijomis.

Pagal elektroninių paslaugų paskirtį bei naudojamas technologijas jos yra skirstomos i kelis tipus:

- Inteneto svetainės;
- Inteneto portalai;
- Verslas-verslui sistemos (angl. *Busines to Busines*);
- Verslas-klientui sistemos (angl. *Busines to Client*);
- Klientas-klientui sistemos (angl. *Person to Person*).

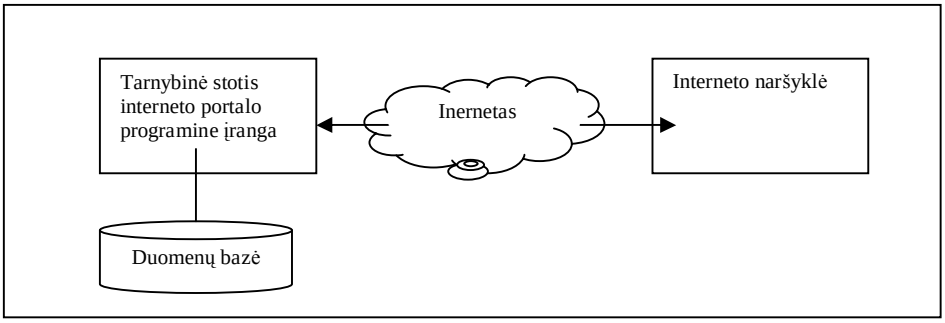
1.1.2 Elektroninių paslaugų tipai

Inteneto svetainės

Žiniatinklio svetainės yra skirtos tik informacijai atvaizduoti.

Inteneto portalai

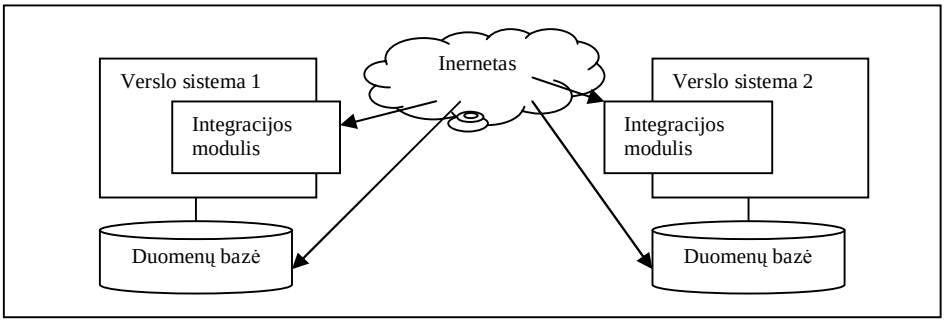
Inteneto portalai skiriasi nuo paprastos inteneto svetainės tuo, kad portalai yra skirti ne tik tam tikros informacijos pateikimui, bet ir tam tikros logikos vykdymui. Kitaip sakant, inteneto portalas yra inteneto svetainė, kur vartotojo veiksmai – naršymas portale ir duomenų įvedimas veikia portalo turinį.



1 pav. Iterneto portalo struktūra

Verslas-verslui paslaugos

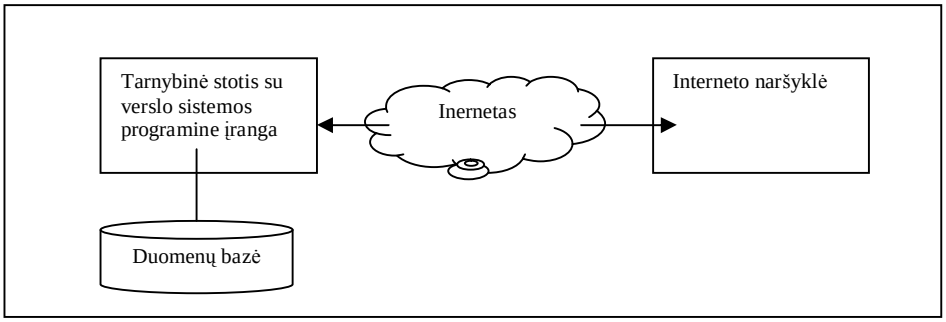
Šiam tipui priskiriamos paslaugos, kurių tiekėjas yra verslas, o vartotojas taip pat verslas. Šio tipo sistemos keičiasi verslo informacija be žmogaus dalyvavimo pačiame informacijos apsikeitimo procese. Verslas-verslui sistemomis galima vadinti atskiras verslo sistemas integruotas tarpusavyje elektroninėmis paslaugomis.



2 pav. Verslas-verslas paslaugos struktūra

Verslas-klientui paslaugos

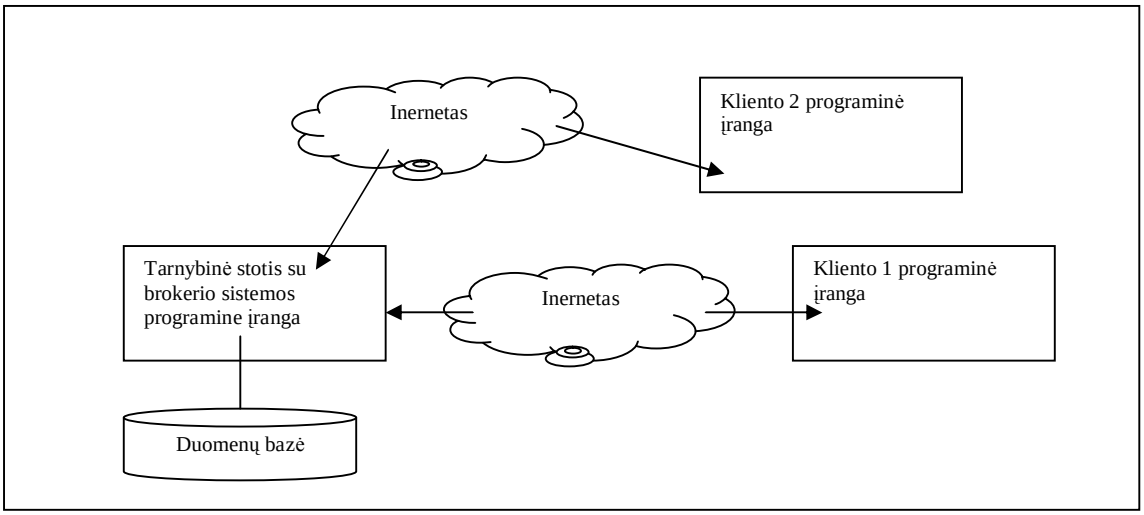
Verslas-klientui paslauga yra susijusi su elektronine prekyba. Naudodamasis šia paslauga klientas perka prekes elektroniniu būdu. Paslaugos pavyzdys – elektroninė parduotuvė. Nuo inteneto portalo skirias tuo, jog reikalauja padidintų saugumo priemonių bei įgyvendina verslo logiką.



3 pav. Verslas-klientui paslaugos struktūra

Klientas-klientui paslaugos

Klientas-klientui paslauga organizuoja informacijos apsikeitimą tarp klientų. Šią paslaugą teikiančios tarnybinės stotys (su brokerio programine įranga) veikia kaip tarpininkės organizuojančios informacijos perdavimą tarp klientų. Šio tipo paslaugos turi vieną labai naudingą savybę – klientai tarpusavyje gali keistis informacija neturėdami internete unikaliai identifikuojančio IP adreso bei be didesnių keblumų išvengti ugniasienių keliamų problemų.



4 pav. Klientas-klientui paslaugos struktūra

1.1.3 Elektroninių paslaugų realizavimo technologijos

Interneto svetainės, portalai bei **Verslas-klientas** paslaugos kuriamos naudojant programavimo priemones, leidžiančias generuoti informaciją HTML formatu.

Verslas-verslas sistemos integruojamos panaudojant CORBA, žiniatinklio paslaugas (*WEB services*) arba specifinius protokolus, veikiančius TCP/IP protokolo pagrindu. Taipogi verslo sistemos gali būti integruotos naudojant paskirstytąsias duomenų bazes.

Klientas-klientas paslaugos gali būti realizuojamos panaudojant CORBA, žiniatinklio paslaugas (*WEB services*), tačiau dažniausiai realizuojamos panaudojant specifinius, tai sričiai sukurtus informacijos perdavimo protokolus.

Lentelė Nr. 1 Technologijų palyginimas

Technologija	Privalumai	Trūkumai
HTML generatoriai	Gerai išvystyta technologija. Paslauga teikiama tiesiogiai vartotojui. Technologija standartizuota.	Nėra patogi duomenų apsikeitimui tarp sistemų, kadangi nėra griežtai struktūrizuota.
CORBA	Gerai išvystyta technologija, leidžianti realizuoti paskirstytas sistemas interneto tinkle. Technologija standartizuota.	Skirta tik duomenų apsikeitimui tarp sistemų. Sudėtinga, reikalaujanti aukštos kvalifikacijos specialistų. Naudojimui reikalingi specialūs komponentai.
Žiniatinklio paslaugos (*WEB services*)	Paprasta, gerai išvystyta technologija, turinti griežtą struktūrą. Technologijai realizuoti naudojamos standartinės, plačiai paplitusios priemonės. Technologija standartizuota.	Skirta tik duomenims tarp sistemų pasikeisti.
Specifiniai protokolai	Protokolai optimizuoti konkrečios problemos sprendimui, todėl yra spartesni nei kitos standartizuotos technologijos. Neturi perteklinės informacijos.	Panaudojami tik siauroje srityje, nėra universalūs todėl neturi standartinių priemonių. Šia technologija perduodamos informacijos struktūra dažniausiai nėra standartizuota.

1.1.4 Elektroninės paslaugos teikimo scenarijus

Vienas iš magistrinio darbo tikslų yra apibendrinti elektroninių paslaugų kūrimo aspektus ir atrasti projektavimo šablonus, tinkamus elektroninių paslaugų realizavimui. Projektavimo šablonų tinkamumui nustatyti turi būti naudojami palyginimo kriterijai.

Taigi, nagrinėjama elektroninė paslauga bus teikiama pagal tokį scenarijų:

1. Vartotojas užregistruoja sistemoje norimus apdoroti duomenis (kardiogramą);
2. Vartotojas užsisako duomenų (kardiogramos) tyrimą;
3. Sistema išanalizuoja duomenis (kardiogramą);
4. Vartotojas gauna analizės rezultatą.

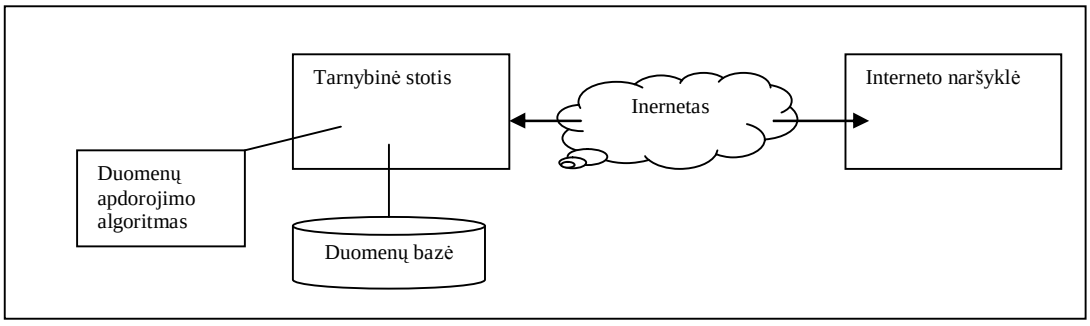
Aprašytas scenarijus yra tinkamas ne tik kardiologijos srityje, tačiau ir bet kurioje kitoje srityje kur teikiama duomenų apdorojimo paslauga. Pagal šį scenarijų gali būti teikiamos mokslinių tyrimų, realizuojančių originalius algoritmus, paslaugos.

1.1.5 Paslaugų tipo ir realizavimo technologijos parinkimas

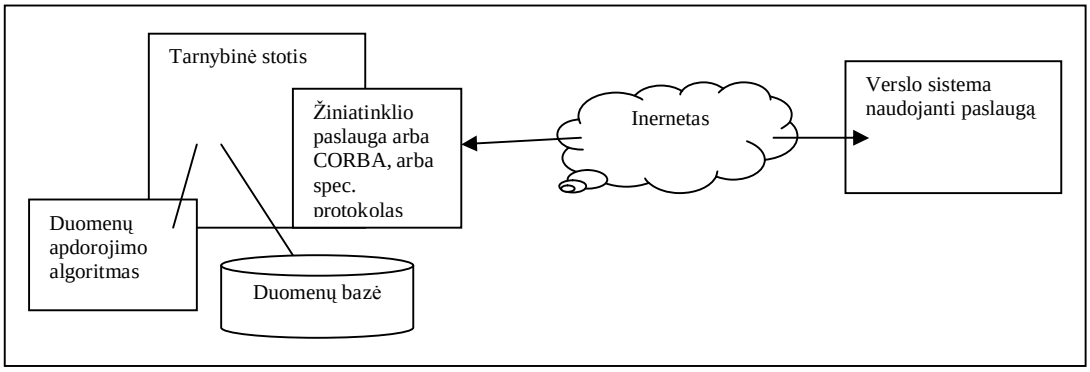
Remiantis anksčiau pateiktu elektroninės paslaugos scenarijumi galima apytiksliai nustatyti kokio tipo paslaugos yra tinkamos scenarijaus realizacijai ir pasiūlyti naują, jungtinį tipą, kuris spręstų darbo tiksle suformuluotas problemas.

Tinkami šie tipai:

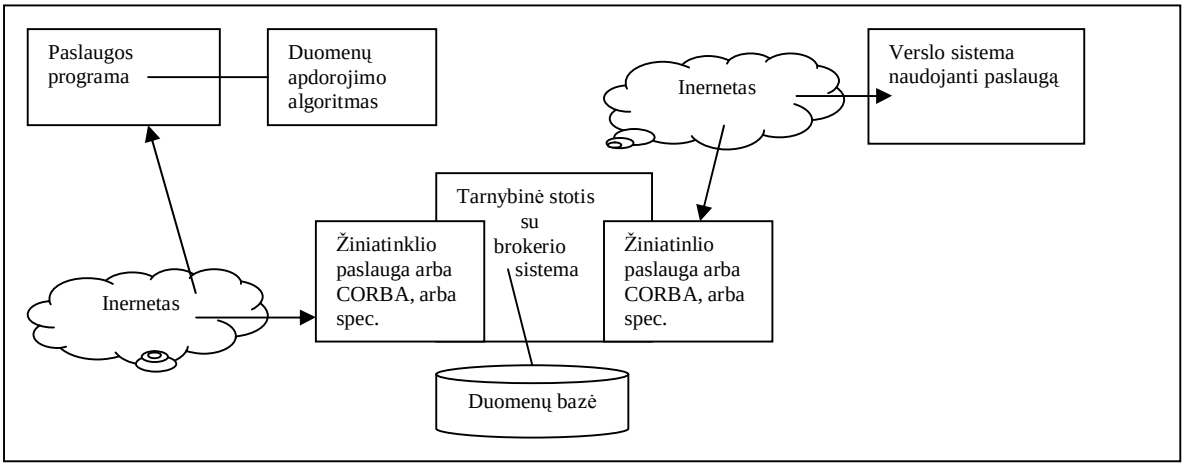
1. Verslas-klientui;
2. Verslas-verslui;
3. Verslas-klientui, verslas-verslui ir klientas-klientui paslaugų junginys.



Pav. 5 Verslas-klientas paslaugos struktūra



Pav. 6 Verslas-verslui paslaugos struktūra



Pav. 7 Paslaugų junginio struktūra

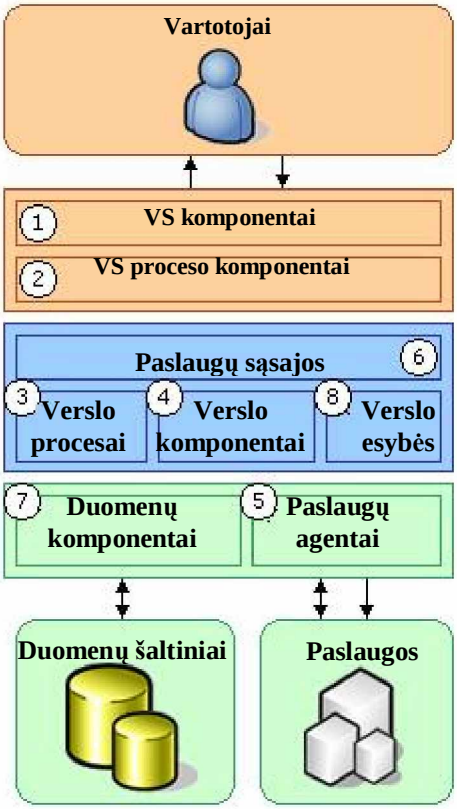
Sistemą realizuoti pasirinkta žiniatinklio paslaugų (*WEB Services*) technologija dėl anksčiau išvardintų privalumų, tačiau projektavimo šablonui tai neturės jokios įtakos.

1.2 Elektroninių paslaugų projektavimo šablonai

Projektavimo šablonas aprašo patikrintą pasikartojančios problemos sprendimą, identifikuoja kontekstą ir jėgas, įtakojančias problemą ir nurodo, kokios bus šablono taikymo pasekmės.[5]

1.2.1 Paslaugų architektūra

Internetu teikiamų paslaugų architektūrą naudinga sudaryti pagal 8 pav. pavaizduotą struktūrą. Šios struktūros privalumas yra tas, jog sistema susideda iš trijų dalių. Kiekviena dalis turi savo paskirtį. Pirmoji dalis skirta duomenų vaizdavimui, antroji dalis skirta verslo logikos įgyvendinimui, duomenų apdorojimui, o trečioji - duomenų valdymui, saugojimui, nuskaitymui.



8 pav. Paslaugų architektūra [2]

1. **VS komponentai**. Dauguma sistemų turi vartotojo sąsają. Per ją vartotojas įveda duomenis į sistemą. Vartotojo sąsaja sudaryta iš komponentų, kurie vaizduoja duomenis bei surenka juos ir patikrina.

2. **VS proceso komponentai**. Daugeliu atvejų vartotojas sąveikauja su sistema pagal iš anksto žinomą scenarijų (procesą). Pavyzdžiui, patalpinus į sistemą kardiogramą, sekantys veiksmai kuriuos jis turės atlikti - nurodyti papildomą informaciją apie kardiogramą ir užsisakyti duomenų analizę. Tam, kad suderinti bei sinchronizuoti šiuos vartotojo veiksmus tarpusavyje, naudinga yra naudoti skirtingus vartotojo sąsajos proceso komponentus.

3. **Verslo procesai**. Po to, kai duomenys jau surinkti, jie gali būti panaudoti tam, kad vykdyti verslo procesus. Pavyzdžiui, po to, kai sistemoje buvo užsakyta konkreti kardiogramos duomenų analizės operacija, gali būti pradėtas jos apdorojimas ir kiti po to sekantys veiksmai. Šie veiksmai gali trukti ilgai, todėl veiksmai bei duomenys susiję su jais turi būti kontroliuojami.

4. **Verslo komponentai**. Net ir labai nesudėtingiems verslo procesams įgyvendinti reikalingi komponentai realizuoja verslo taisykles ir vykdo verslo užduotis. Verslo komponentai įgyvendina verslo logiką.

5. **Paslaugų agentai**. Jei verslo komponentai naudoja išorines paslaugas, reikia priemonių, kurios gaunamą informaciją paverstų į sistemai reikalingą formatą. Šią funkciją atlieka paslaugų agentai.
6. **Paslaugų sąsajos**. Jei verslo logiką norima pateikti kaip paslaugą, tam reikia sukurti išorines sąsajas.
7. **Duomenų komponentai**. Tam, kad centralizuoti duomenų paėmimo bei saugojimo funkcionalumą bei supaprastinti nustatymų valdymą ir priežiūrą, prasminga iškelti duomenų valdymo funkcijas į atskirą sluoksnį. Duomenų komponentai naudojami duomenų apsikeitimui tarp duomenų šaltinių ir sistemos.
8. **Duomenų esybės**: Daugelyje sistemų vyksta duomenų apsikeitimas tarp komponentų. Duomenų esybės naudojamos sistemos viduje dažniausiai būna standartinės duomenų struktūros, XML srautai, bet jos taip pat gali būti įgyvendintos kaip specifinės objektinės klasės atitinkančios realaus pasaulio esybes.

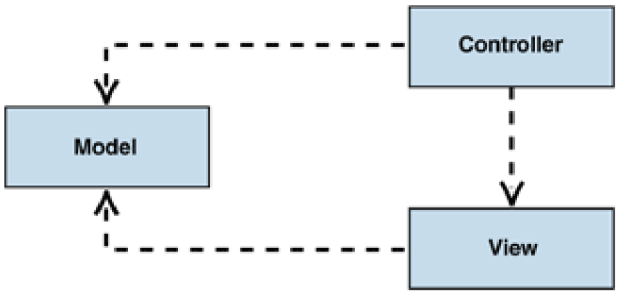
1.2.2 Priešakinis valdiklis (*front controller*)

Daugelio kompiuterinių sistemų uždavinys yra paimti duomenis iš duomenų šaltinio ir parodyti juos vartotojui. Po to, kai vartotojas pakeičia duomenis, sistema išsaugo pasikeitusią informaciją į duomenų saugyklą. Kadangi pagrindiniai informacijos srautai yra tarp duomenų saugyklos ir vartotojo sąsajos, gali pasirodyti, jog naudinga būtų sujungti šias dvi dalis. Problema yra ta, jog vartotojo sąsaja turi tendenciją žymiai dažniau keistis nei duomenų saugyklos sistema. Kita apjungimo problema yra ta, jog verslo sistemose neapsiribojama tik duomenų perdavimu tarp vartotojo sąsajos ir duomenų saugyklos. Dažniausiai vykdoma ir verslo logika, kuri yra žymiai sudėtingesnė nei minėtos operacijos. Tam, kad išspręsti šias problemas yra sukurtas projektavimo šablonas modelis-vaizdas-valdiklis (*model-view-controller*).

Modelis. Modelis valdo duomenis ir sistemos elgesį, atsako į užklausas apie sistemos būseną (dažniausiai iš vaizdo) bei reaguoja į instrukcijas pakeisti būseną (dažniausiai iš valdiklio).

Vaizdas. Vaizdas kontroliuoja informacijos vaizdavimą.

Valdiklis. Valdiklis interpretuoja pele ar klaviatūra įvestus duomenis ir informuoja modelį arba vaizdą apie pasikeitimus.

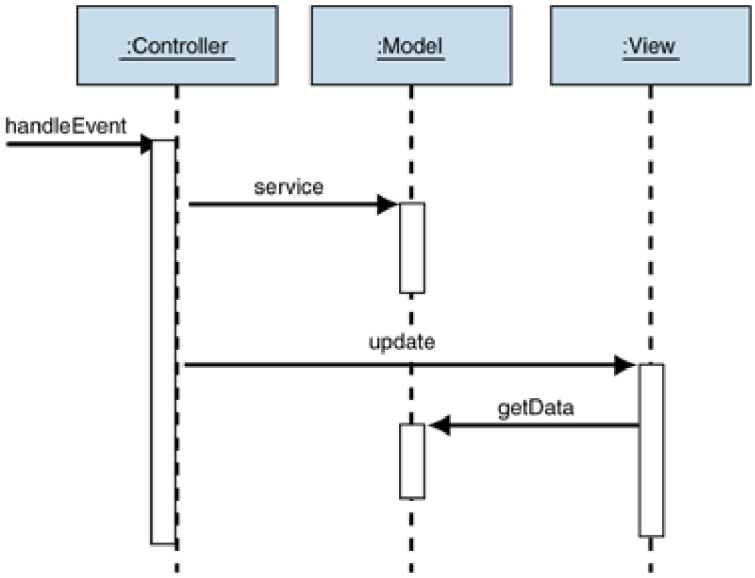


9 pav. Modelis-vaizdas-valdiklis[2]

Internetinio puslapio atveju vaizdas ir valdiklis yra priklausomi nuo modelio. Modelis nėra priklausomas nei nuo valdiklio, nei nuo vaizdo. Valdiklis yra priklausomas nuo vaizdo.

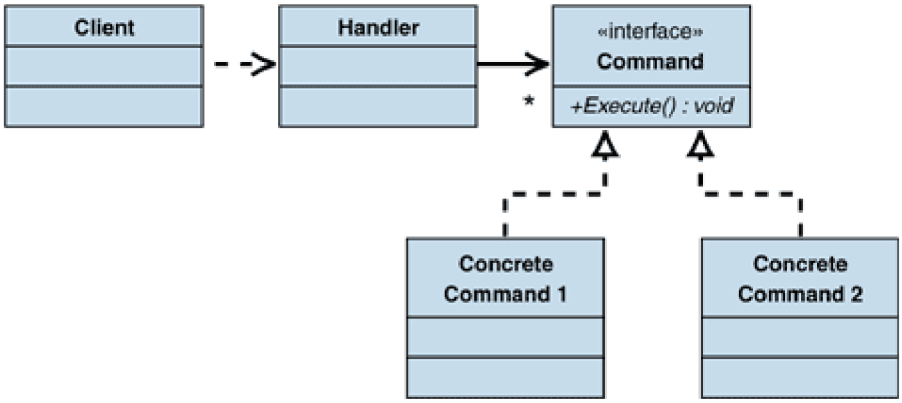
Modelis-vaizdas-valdiklis yra fundamentalus projektavimo šablonas naudotinas tam, kad atskirti vartotojo sąsajos valdymo logiką nuo verslo logikos.

Valdiklis (*controller*) pakeičia modelį (*model*) ir informuoja vaizdą (*view*) apie tai, jog modelis pasikeitė ir vaizdas turi būti atnaujintas. Modelis šiame scenarijuje yra visiškai nepriklausomas nuo vaizdo ir valdiklio, tai reiškia, jog nėra poreikio informuoti apie modelio būsenos pasikeitimus. Tai yra labai naudinga interneto puslapiuose, kur naršyklė parodo vaizdą ir reaguoja į vartotojo įvedamą informaciją, bet neseka ar duomenys pasikeitė serveryje. Tik tuomet, kai vartotojas pareikalauja, serveris pateikia informaciją apie pasikeitimus.



10 pav. Sekų diagrama[2]

Priešakinis valdiklis (*front controller)* yra patobulintas modelis-vaizdas-valdiklis valdiklio variantas. Šis valdiklis susideda iš dviejų dalių: valdytojo (*handler*) ir komandų hierarchijos).

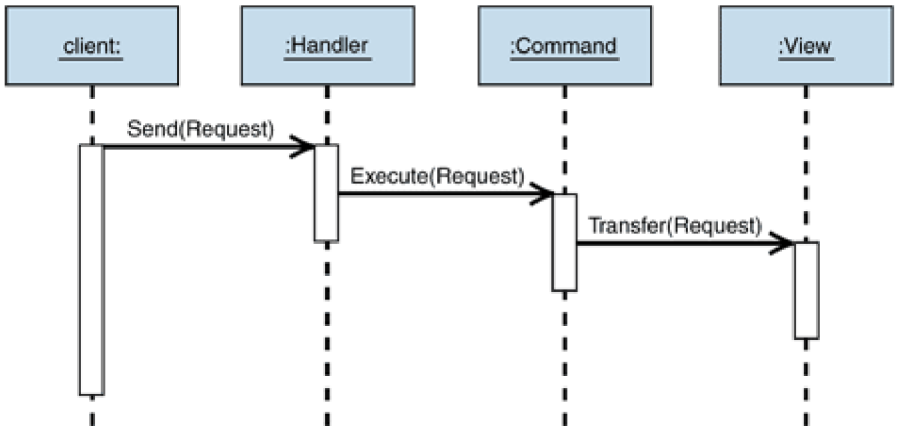


11 pav. Priešakinis valdiklis[2]

Valdytojas (*handler*) atlieka dvi funkcijas:

1. **Surenka parametrus**. Valdytojas gauna HTTP užklausas iš žiniatinklio serverio (*WEB server*) ir išgauna iš jų reikalingus parametrus;
2. **Parenka komandas**. Valdytojas naudoja parametrus iš užklausos tam, kad parinktų tinkamą komandą ir ją įvykdytų.

Paveikslėlyje 12 pavaizduotos šios dvi funkcijos.



12 pav. Priešakinio valdiklio sekų diagrama [2]

Komandos (*command*) yra taip pat valdiklio (*handler*) dalis. Komandos savyje turi specifinius veiksmus (*actions*). Valdiklis su visomis komandomis (*command*) elgiasi vienodai – parenka tinkamą ir aktyvuota jos vykdymą. Po to, kai komanda baigia vykdyti veiksmus, ji parenka tinkamą vaizdą (*view*).

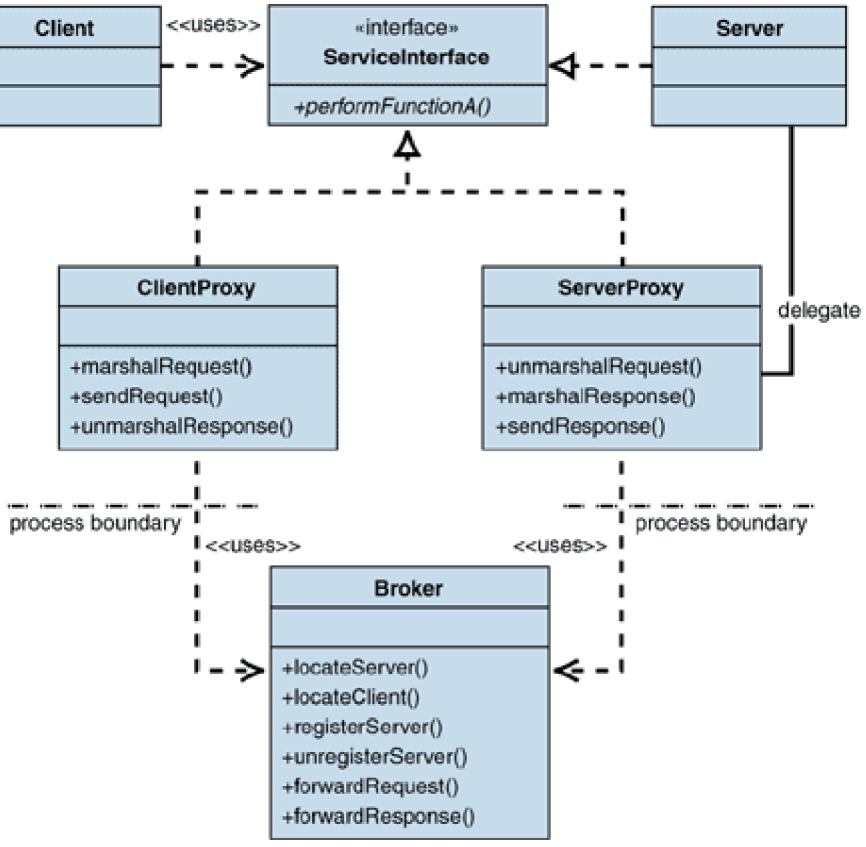
Šis projektavimo šablonas yra tinkamas įgyvendinti elektroninę paslaugą, tačiau jei sistema atliks didelius skaičiavimus esant dideliam klientų skaičiui, paslauga veiks lėtai, nes sistema veikia vienalytėje terpėje ir visi klientai dalinasi tarpusavyje ribotais sistemos resursais.

1.2.3 Brokeris

Daugelis sudėtingų sistemų turi daugiau nei vieną procesorių ar yra sudarytos iš paskirstytųjų kompiuterių (*distributed computers*). Tam, kad aprašyti paskirstytosios sistemos struktūrą, naudojamas brokerio projektavimo šablonas. Brokerio šablonas sprendžia problemą, nusakydamas kaip turi būti struktūrizuota sistema, kad jos kūrėjui nereikėtų rūpintis tuo, kaip bus komunikuojama su paskirstytosios sistemos komponentais.

Brokerio projektavimo šablonas naudojamas tam, kad atskirti nutolusių metodų iškvietimo funkcijas nuo verslo komponento į skirtingą sluoksnį. Šis sluoksnis klientui suteikia sąsają iškviesti nutolusių komponentų metodus taip, lyg jie nebūtų nutolę. Kliento viduje iškviesti metodai sužadina jų vykdymą nutolusiuose komponentuose. Brokerio projektavimo šablone klientu (*client*) vadinamas verslo komponentas, kuris inicijuoja nutolusios paslaugos iškvietimą, o serveriu (*server*) vadinamas nutolęs komponentas, kuris reaguoja į nutolusios paslaugos iškvietimą.

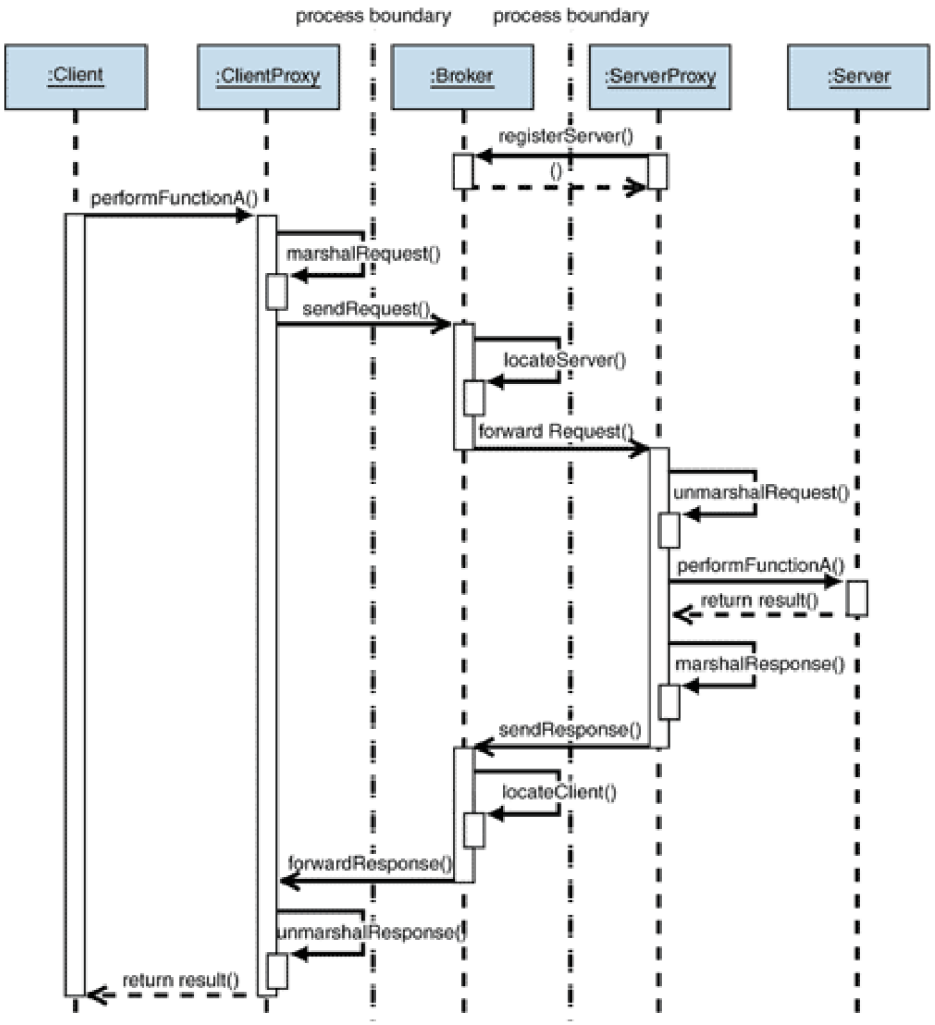
Brokeris gali atlikti ir sudėtingesnę, nei nutolusių metodų iškvietimo funkciją. Brokeris gali atlikti tarpininko tarp dviejų sistemų vaidmenį ir pats nepriklausyti nei vienai iš tų sistemų. Tokiu atveju brokeris tampa atskiru komponentu, kurio vieta yra gerai žinoma ir pasiekiama klientui. Brokeris atlieka serverių paiešką ir paskirsto klientų užklausas serveriams. Klientai neturi tiesioginio kontakto su serveriais ir komunikuoja su jais per tarpininką brokerį. Sistemos statinė schema pavaizduota 13 paveikslėlyje.



13 pav. Brokerio klasių diagrama[2]

Paveikslėlyje 14 pavaizduota brokerio sekų diagrama. Klientas (*Client*) iškviečia funkciją A. Klientas kreipiasi į *ClientProxy* komponentą, kuris supakuoja funkcijos parametrus į perdavimui tinklu tinkamą formatą ir nusiunčia užklausą brokeriui (*Broker*). Brokeris išrenka iš užsiregistravusių serverių sąrašo vieną serverį ir nusiunčia jo *ServerProxy* komponentui užklausą. *ServerProxy* komponentas išpakuoja siunčiamus duomenis ir iškviečia funkciją A serveryje. *ServerProxy* gauna funkcijos A vykdymo rezultatus, supakuoja juos į perdavimui tinklu tinkamą formatą ir nusiunčia brokeriui. Brokeris siunčia gautus duomenis klientui per jo *ClientProxy* komponentą. *ClientProxy* komponentas išpakuoja duomenis ir grąžina juos klientui kaip funkcijos A vykdymo rezultatą.

Klientas, brokeris ir serveris yra skirtinguose procesuose, kurie gali būti vykdomi skirtinguose kompiuteriuose. Jei komunikacija tarp skirtingų procesų vykdoma internetu, brokeris ir serveriai privalo turėti realius IP adresus.



14 pav. Brokerio sekų diagrama[2]

1.3 Išvados

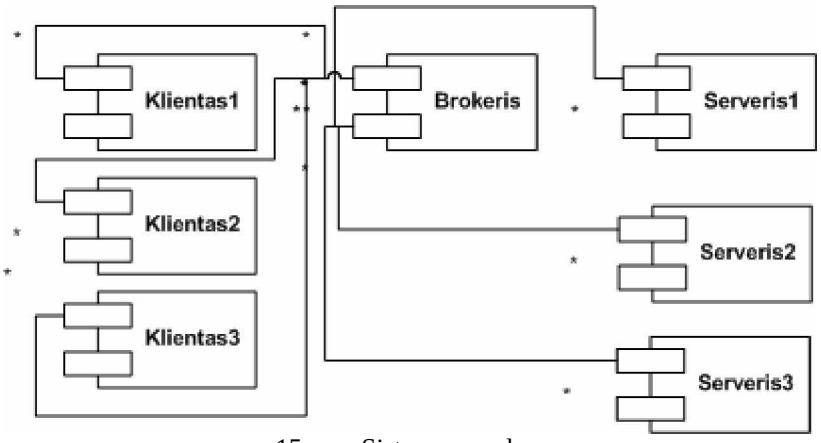
Yra keletas technologijų, kuriomis galima realizuoti elektronines paslaugas. Šiuo atveju buvo pasirinkta interneto puslapių generatorių ir žiniatiklio paslaugų (*WEB services*) technologijos. Pasirinkimo kriterijai yra aiškūs ir nesunkiai įvertinami, tačiau nėra tinkamo projektavimo šablono, kuris tenkintų keliamus reikalavimus. Priešakinis valdiklis (*Front controller*) yra vienas iš problemos sprendimo būdų, tačiau jis nenumato paskirstytos sistemos architektūros, todėl jei bus vykdomi dideli skaičiavimai ir prie sistemos jungsis daug vartotojų, sistema gali nesugebėti apdoroti didelių informacijos srautų. Tam, kad realizuoti paskirstytą sistemą, yra sukurtas brokerio (broker) projektavimo šablonas, tačiau jis turi apribojimus. Brokerio projektavimo šablonas tiktų, jeigu nebūtų magistro darbe įvesto apribojimo, jog algoritmo programa (serveris) neturi išorinio IP adreso. Reikia surasti būdą, kaip patobulinti brokerio projektavimo šabloną, jog galima būtų išvengti šio apribojimo. Taip pat reikia įvertinti, kokiais atvejais tinka naudoti priešakinio valdiklio, o kokiais atvejais - brokerio projektavimo šablonus. Tam reikia surasti kriterijus.

2 TEORINIS ELEKTRONINĖS PASLAUGOS PROJEKTAVIMO ŠABLONAS

2.1Sistemos sandara

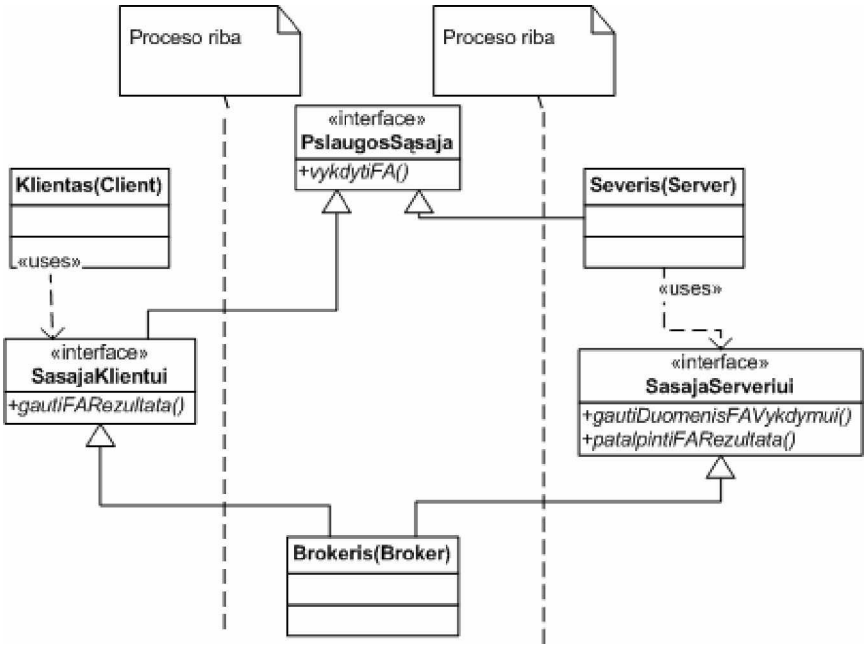
Sistemą galima realizuoti naudojant priešakinio valdiklio projektavimo šabloną, tačiau jei sistema atliks didelius skaičiavimus, esant dideliam klientų skaičiui, paslauga veiks lėtai.

Šiai problemai išspręsti siūlomas naujas elektroninės paslaugos projektavimo šablonas, kurio pagrindinis komponentas būtų brokeris. Sistema sudaryta iš klientų, brokerio ir serverių. Vienintelis brokeris turi realų IP adresą, o serveriai, kaip ir klientai, neprivalo būti pasiekiami tiesiogiai, kadangi jie patys bus duomenų pasikeitimo su brokeriu iniciatoriai. Klientai jungsis prie brokerio ir prašys elektroninių paslaugų, o serveriai jungsis prie brokerio ir prašys duomenų apdorojimui. Brokeris atliks tarpininko ir duomenų bei rezultatų maršrutizatoriaus vaidmenį.



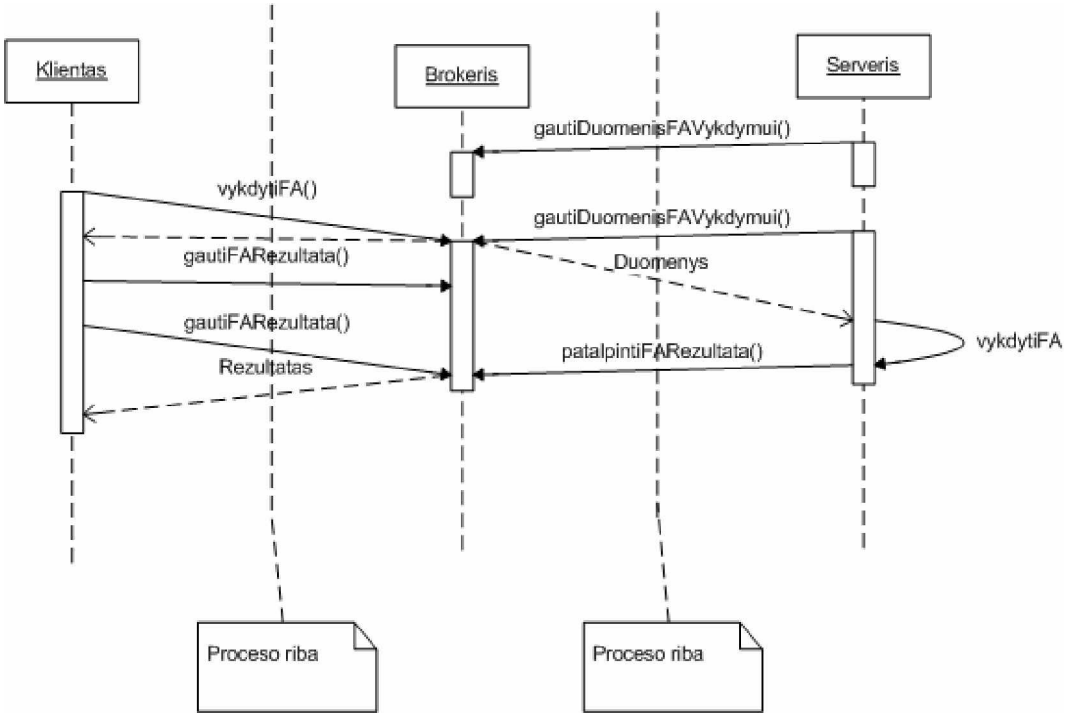
15 pav. Sistemos sandara

2.2 Projektavimo šablonai



16 pav. Statinis modelis

16 paveikslėlyje pavaizduotas statinis sistemos modelis. Klientas, brokeris ir serveris yra atskiruose procesuose. Daroma prielaida, kad tarpusavyje jie komunikuoja žiniatinklio paslaugų (*WEB services*) technologijos pagalba. 17 paveikslėlyje pavaizduotas tipinis scenarijus, kaip turėtų būti teikiama elektroninė paslauga. Klientas jungiasi prie brokerio ir prašo suteikti jam paslaugą FA, iškviesdamas funkciją *vykdytiFA()*. Per funkcijos parametrus jis perduoda reikiamus apdoroti duomenis brokeriui. Serveris kas tam tikrą laiko tarpą klausia brokerio ar yra duomenų apdorojimui, kviesdamas funkciją *gautiDuomenisFAVykdymui()*. Jei funkcija gražina duomenis, serveris apdoroja juos įvykdydamas savo vidinę *vykdytiFA()* funkciją. Baigęs apdoroti informaciją, serveris patalpina rezultatus į brokerį. Klientas kas tam tikrą laiko tarpą klausia brokerio ar jau apdoroti duomenys, kviesdamas funkciją *gautiRezultata()* ir tuo atveju jei duomenys jau apdoroti, gauna juos.



17 pav. Sekų diagrama

Šio sprendimo privalumas yra tas, kad kontakto iniciatoriai yra klientas ir severis, kitaip sakant, brokeris atlieka pasyvų vaidmenį tik atsakinėdamas į užklausas. Dėl šios priežasties brokeriui nereikia žinoti nei kliento, nei serverio IP adreso. 13 paveikslėlyje pavaizduotas brokeris privalo žinoti serverio IP adresą. Tam, kad brokeris galėtų geriau paskirstyti darbus ir identifikuoti serverius, kurie nesugebėjo įvykdyti skirtos jiems užduoties, brokeryje galima numatyti dar vieną funkciją - *alive()*. Šią brokerio funkciją serveris periodiškai iškvietinėtų tol, kol vykdoma *vykdytiFA()* funkcija. Taip brokeris gali nustatyti ar serveris atlieka paskirtą jam užduotį ir ar reikia ją paskirti kitam serveriui.

2.3 Šablonų palyginimo kriterijai

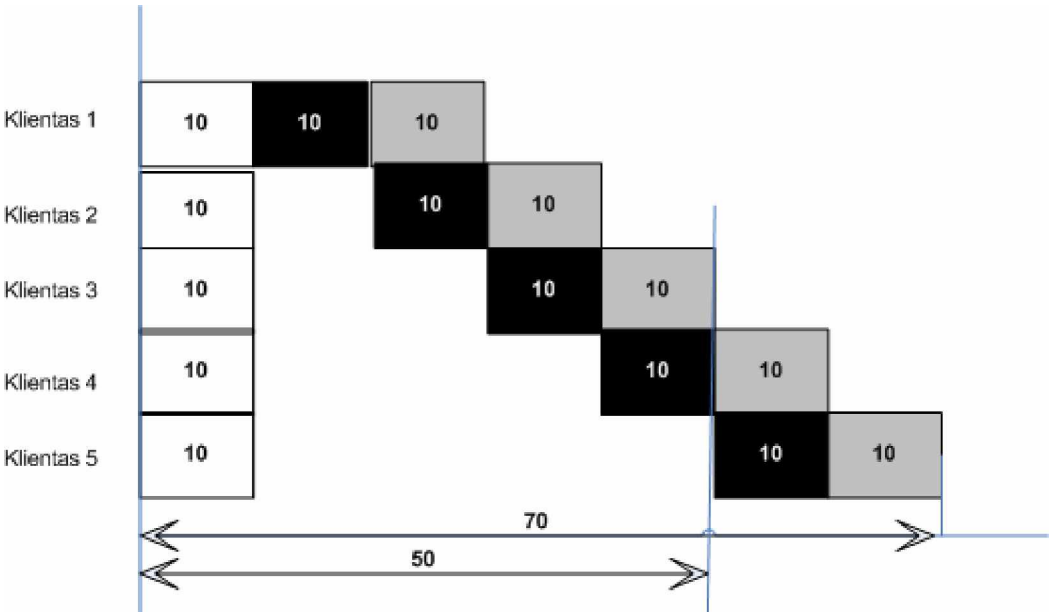
Elektroninės paslaugos teikimo atveju pagrindinis matavimo kriterijus yra laikas per kurį klientas gaus paslaugą.

2.4 Šablonų palyginimas

Be papildomo tyrimo galima nustatyti, jog naudojant pasiūlytą projektavimo šabloną, sistemą galima plėsti prijungiant daugiau serverių. Naudojant priešakinio valdiklio projektavimo šabloną, galima tik padidinti procesorių skaičių sistemoje, o tai įmanoma tik superkompiuteriuose.

2.4.1 Priešakinis valdiklis

Šiame skyrelyje nagrinėsiu laiko sąnaudas teikiant elektronines paslaugas, sistemą realizavus pagal priešakinio valdiklio projektavimo šabloną.



18 pav. Matavimas 1

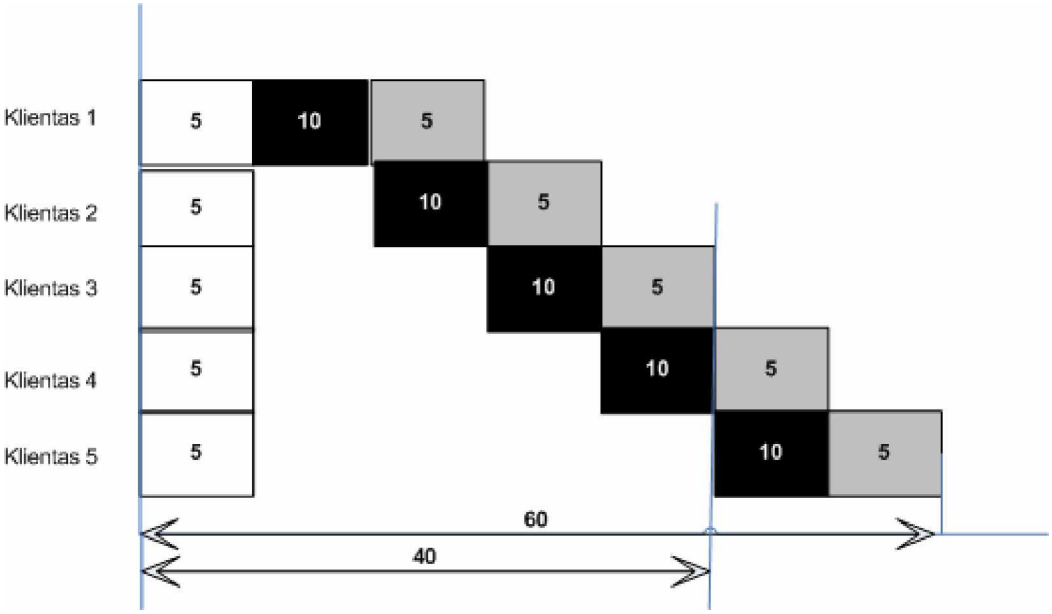
18-20 paveikslėliuose pavaizduotos operacijų trukmės, kai sistema aptarnauja kelių klientų užklausas. Laikas matuojamas sąlyginiais mato vienetais. Baltais stačiakampiais žymimos pradinių duomenų siuntimo į sistemą operacijos, pilkais kvadratėliais – rezultato, o juodais – duomenų apdorojimo. Skaičius stačiakampio viduje žymi operacijos trukmę.

18 paveikslėlyje matome atvejį, kai siuntimo bei duomenų apdorojimo operacijos trunka vienodą laiko tarpą. Tokiu atveju sistema aptarnaus penkis klientus per 70 laiko matavimo vienetų, o tris klientus - per 50 laiko matavimo vienetų.

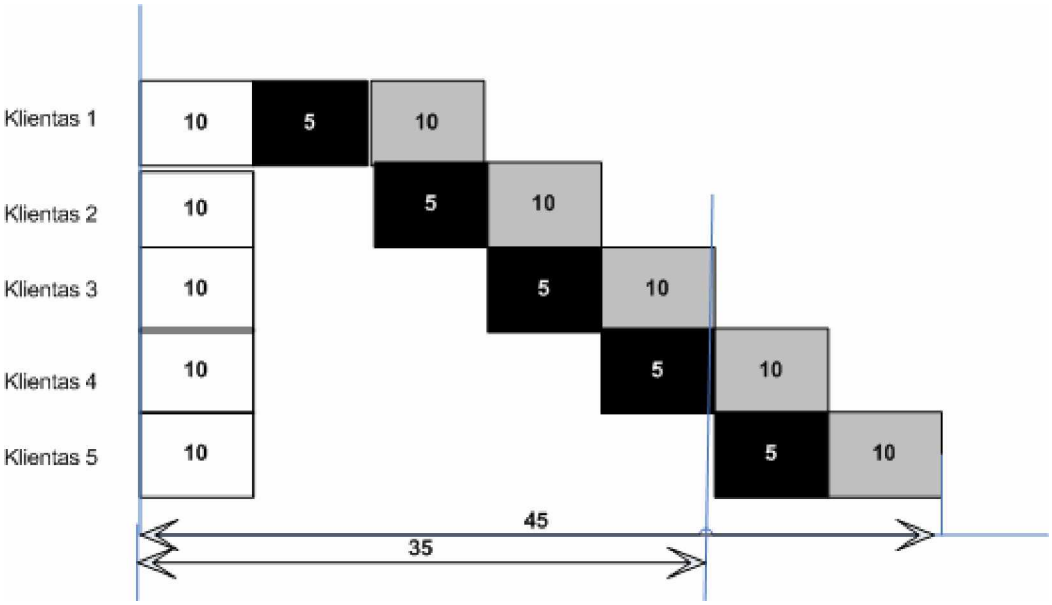
19 paveikslėlyje matome, kaip pasikeis klientų aptarnavimo trukmė paspartinus siuntimo operacijas.

20 paveikslėlyje matome kas nutiks, jei paspartinsime duomenų apdorojimo operaciją.

Iš sukurtų situacijų matyti, jog sistemos našumas labiausiai kyla spartinant duomenų apdorojimą, o augant klientų skaičiui sistemos našumas smarkiai krenta.



19 pav. Matavimas 2



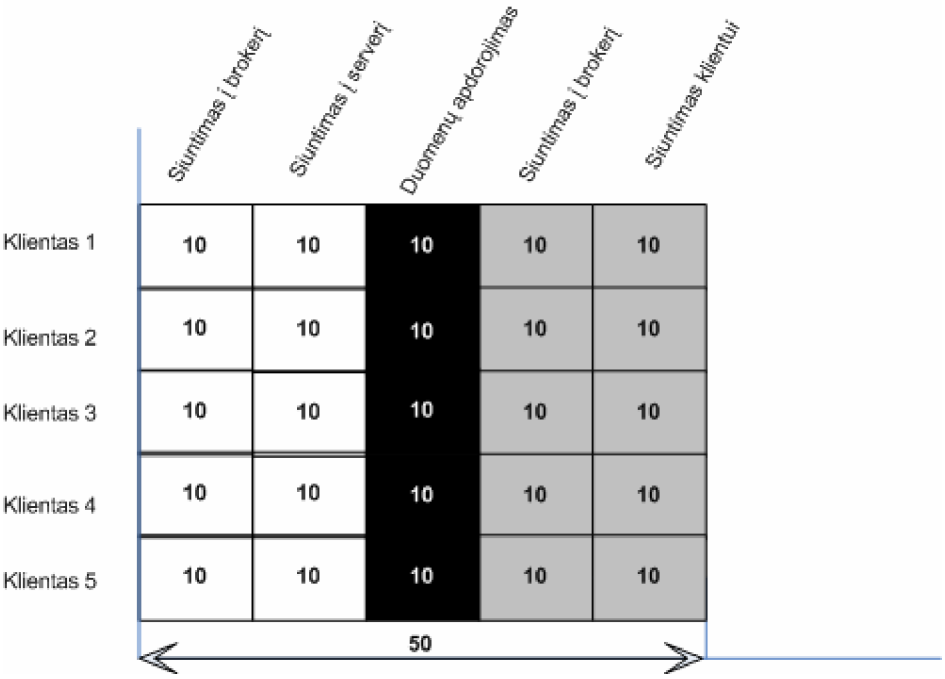
20 pav. Matavimas 3

2.4.2 Pasiūlytas projektavimo šablonas

Šiame skyrelyje nagrinėsiu laiko sąnaudas teikiant elektronines paslaugas, sistemą realizavus pagal pasiūlytą brokerio projektavimo šabloną.

21-25 paveikslėliuose pavaizduotos operacijų trukmės, kai sistema aptarnauja kelių klientų užklausas. Laikas matuojamas sąlyginiais mato vienetais. Baltais stačiakampiais žymimas pradinių duomenų siuntimo į sistemą operacijos, pilkais kvadratėliais – rezultato, o juodais – duomenų

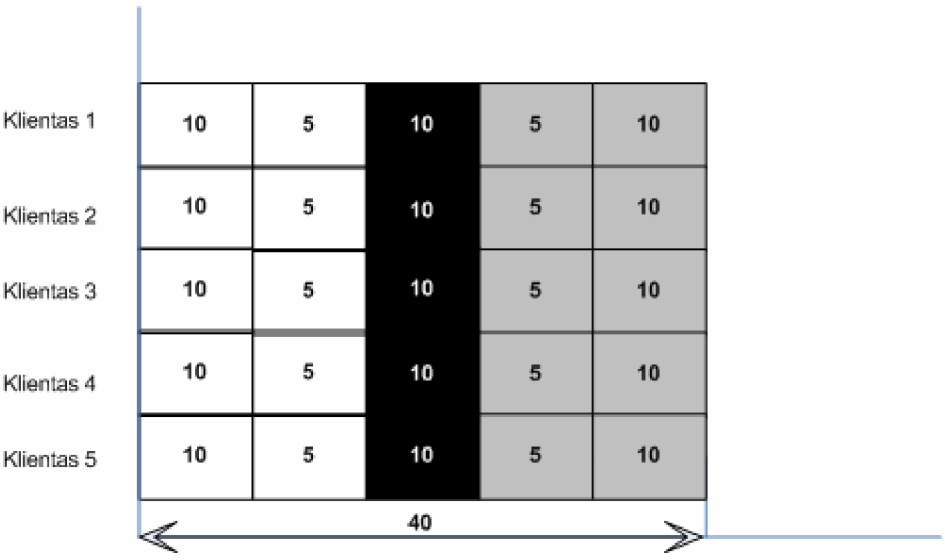
apdorojimo. Skaičius stačiakampio viduje žymi operacijos trukmę. Kadangi informacija siunčiama iš pradžių i brokerį, paskui į serverį ir po duomenų apdorojimo sugrįžta atbuline tvarka, tai bus vykdomos kelios duomenų siuntimo operacijos. Vienos - tarp kliento ir brokerio, kitos - tarp brokerio ir serverio.



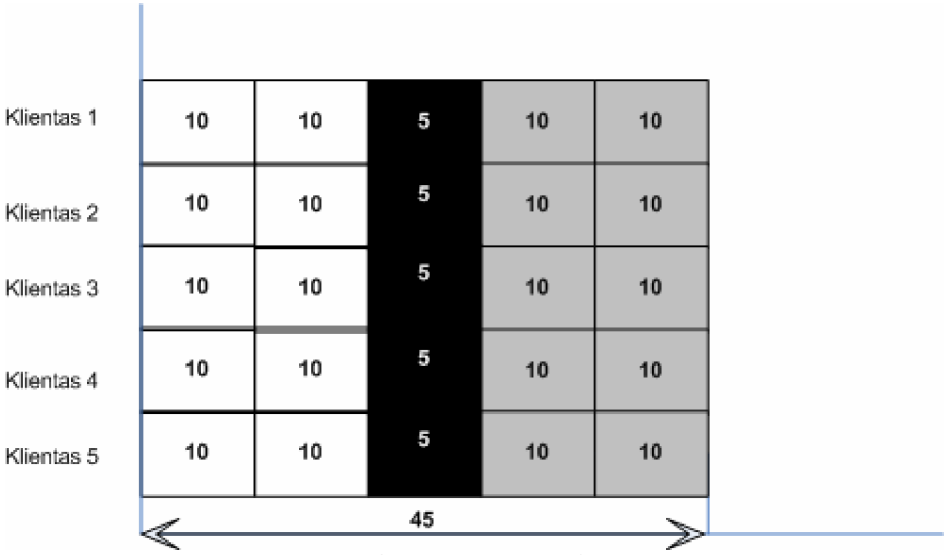
21 pav. Matavimas 4

21 paveikslėlyje matome situaciją, kai visos operacijos trunka vienodą laiko tarpą. Skirtingai nei priešakinio kontrolerio atveju, matome, jog klientų skaičius sistemos bendro našumo neįtakoja.

22 paveikslėlyje matome, kad, paspartinus ryšio kanalą tarp brokerio ir serverio, bendras sistemos našumas išauga.

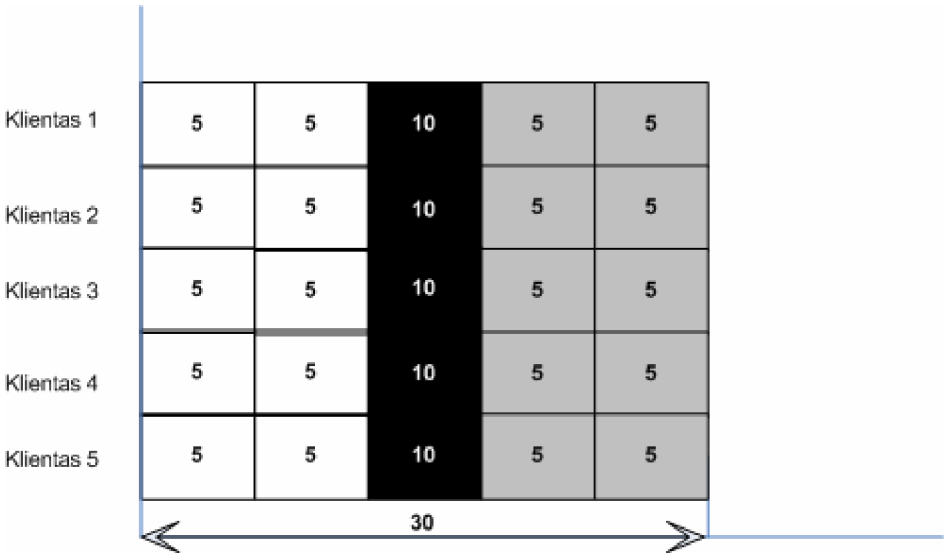


22 pav. Matavimas 5



23 pav. Matavimas 6

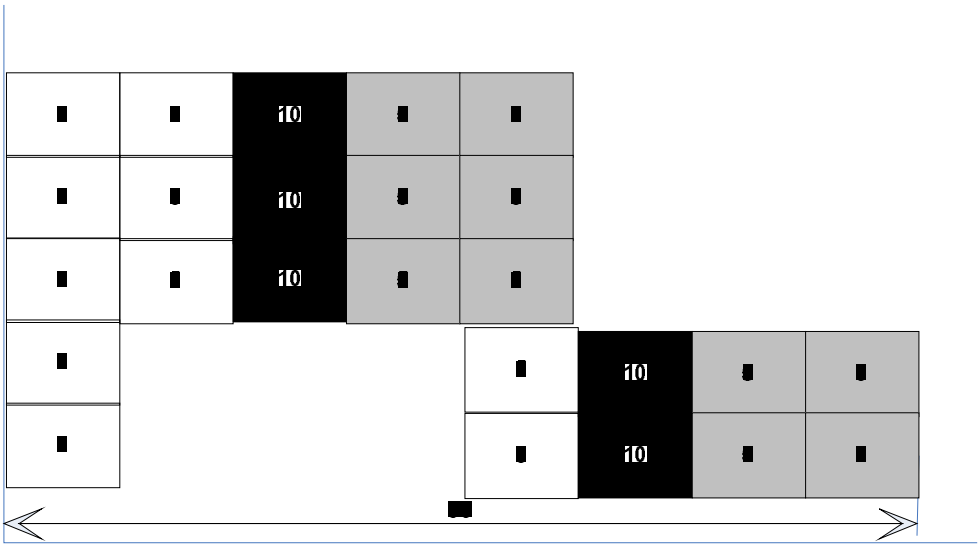
23 paveikslėlyje matome, kad, paspartinus duomenų apdorojimą, bendras sistemos našumas išauga, tačiau ne taip ženkliai.



24 pav. Matavimas 7

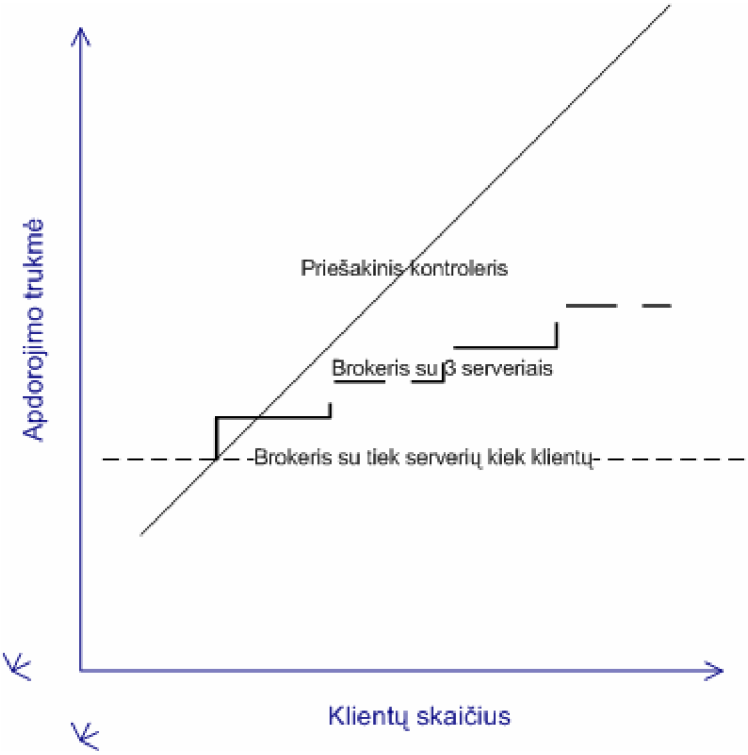
23 paveikslėlyje matome, kad paspartinus duomenų perdavimą tarp visų sistemos komponentų, bendras sistemos našumas išauga ženkliai.

24 paveikslėlyje matome situaciją, kai klientų skaičius yra didesnis nei galinčių juos aptarnauti brokerių skaičius. Šiuo atveju, kiekvienas iš trijų serverių galės pradėti nauja darbą tik tuomet, kai pabaigė buvusį.



25 pav. Matavimas 8

26 grafike matyti, kad, augant klientų skaičiui, bendras sistemos našumas yra didesnis naudojant pasiūlytąjį projektavimo šabloną. Norint padidinti pastarosios sistemos našumą, tikslinga yra didinti serverių skaičių.



26 pav. Rezultatų apibendrinimas

2.5 Išvados

Teoriškai įmanoma išspręsti iškilusias problemas teikiant elektronines paslaugas. Pasiūlytas brokerio projektavimo šablonas įgyvendina užduočių paskirstymą tarp daugelio serverių, sprendžia bendro sistemos našumo problemas, serveriai neprivalo turėti išorinių IP adresų ir sistema gali būti plečiama.

Esant mažam prisijungusių prie sistemos klientų skaičiui, elektroninės paslaugos teikimui tinkamesnis yra priešakinio valdiklio šablonas. Klientų skaičiui augant, pasiūlytojo brokerio šablonas yra efektyvesnis.

3 ELEKTRONINĖS PASLAUGOS REALIZACIJA

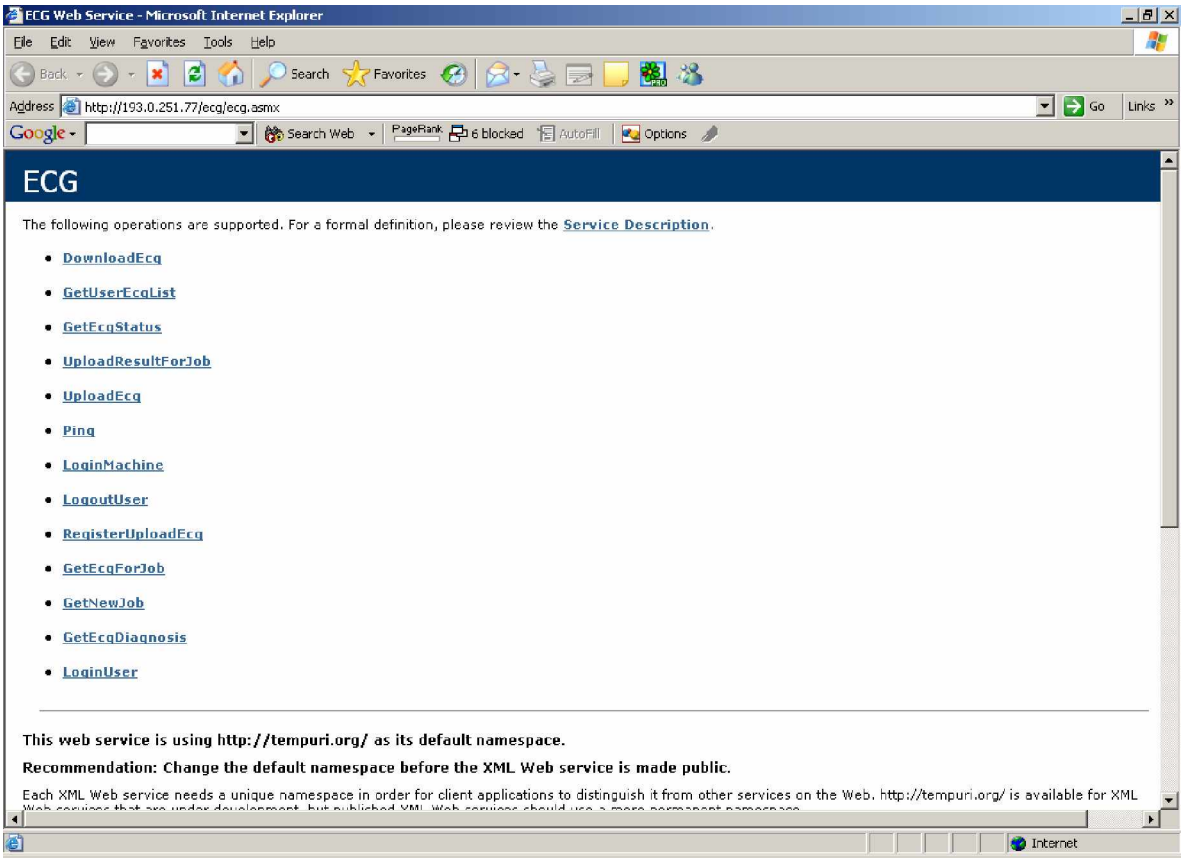
3.1 Eksperimento tikslas

Eksperimento tikslas - patikrinti ar pasiūlytą teorinį projektavimo šabloną įmanoma realizuoti praktiškai.

3.2 Eksperimento aplinka

Elektroninė paslauga realizuojama naudojant Microsoft .NET. Elektroninė paslauga sudaryta iš trijų struktūrinių dalių – kliento, brokerio ir serverio.

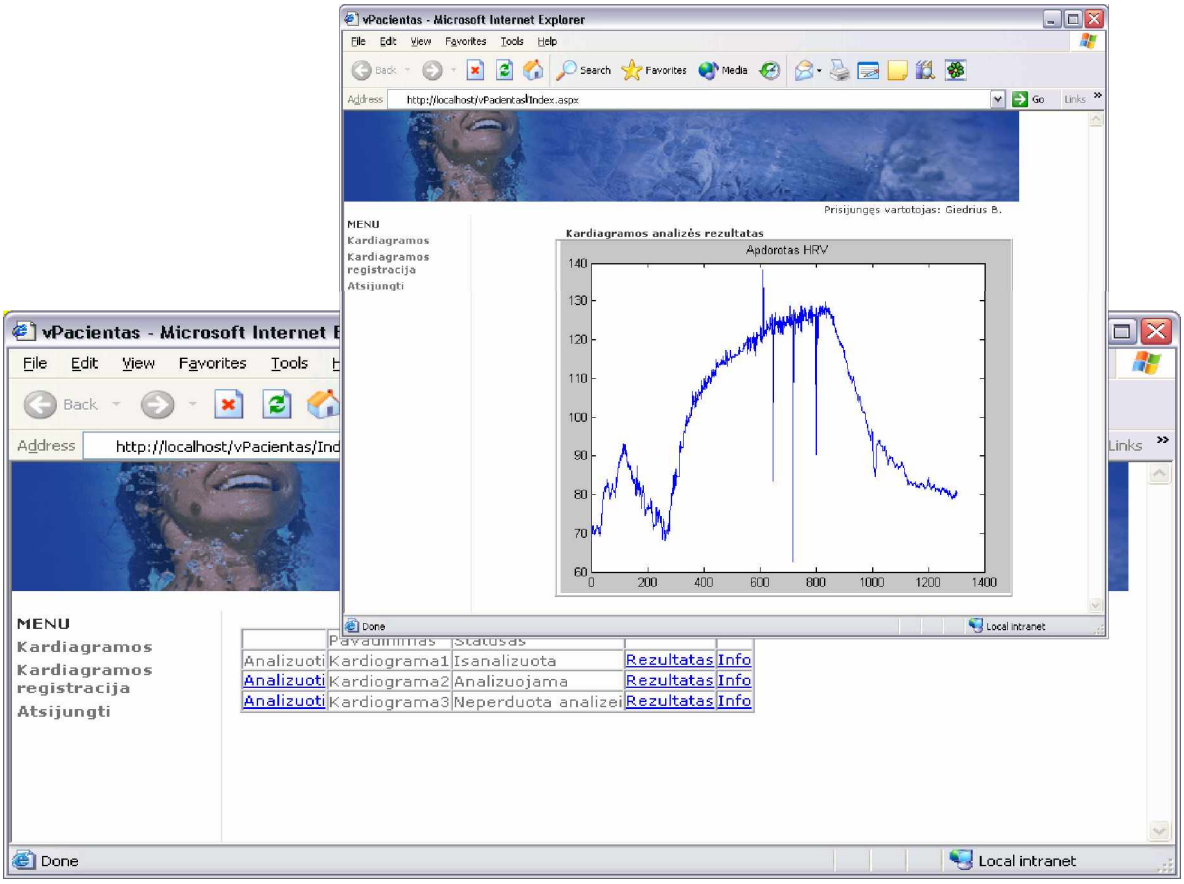
Brokeris realizuojamas panaudojant žiniatinklio paslaugų (*WEB Services*) technologiją. Brokerio žiniatinklio paslauga (*WEB Service*) yra viešai pasiekiama, nes kompiuterinė sistema, kurioje jis veikia, turi išorinį IP adresą. Brokeris naudoja duomenų bazę vartotojams, užduotims ir kitai informacijai saugoti. 27 paveikslėlyje matomas funkcijų realizuotų brokeryje sąrašas.



27 pav. Realizuotas brokeris

Klientas gali būti realizuotas kaip internetinis puslapis arba kaip savarankiška programa. Klientas brokerio funkcijas iškvietinėja naudodamasis žiniatinklio paslaugų (*WEB Services*)

technologija. Klientas gali būti ir didesnės sistemos dalimi. Šiuo atveju klientas realizuotas kaip internetinis puslapis. 28 paveikslėlyje pavaizduotas klientą realizuojantis internetinis puslapis.



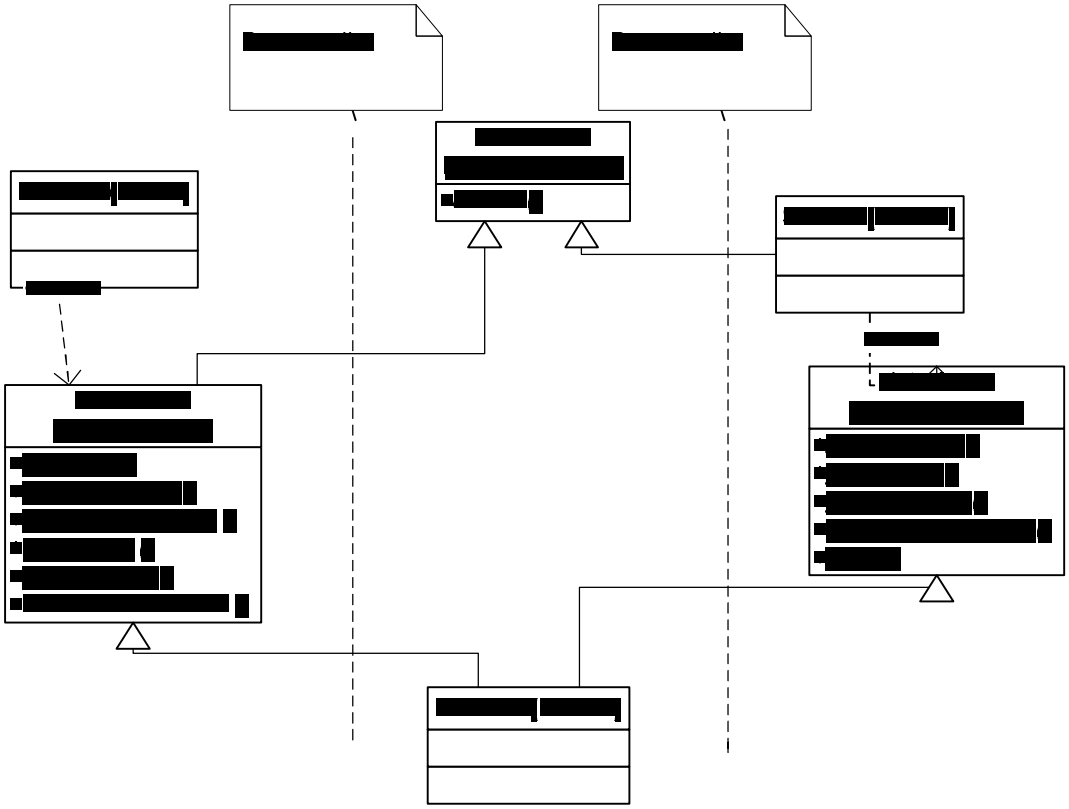
28 pav. Realizuotas klientas

Serveris realizuotas kaip savarankiškai veikianti programa, kuri brokerio funkcijas iškvietinėja naudodamasi žiniatinklio paslaugų (*WEB Service*) technologija. Serveryje vykdomas amžinas ciklas, kiekviena iteracija sudaryta iš tokių operacijų: naujų duomenų parsiuntamas iš brokerio, analizė, rezultatų į brokerį patalpinimas. Serverių gali būti daug – tai reiškia, kad gali būti panaudota daug kompiuterių, su įdiegta serverio programine įranga.

Kardiologinių duomenų analizė vykdoma panaudojant Matlab skaičiavimų paketą. Su Matlab pagalbinėmis priemonėmis yra sukuriamas COM objektas. Jis realizuoja matematiniais metodais paremtą analizės algoritmą. Serveris analizei naudoja Matlab COM komponentą

3.3 Realizuotas projektavimo šablonas

Pagal teorinį šabloną suprojektuota reali sistema pavaizduota 27 paveikslėlyje.



27 pav. Realizuoto šablono klasių diagrama

Sistema susideda iš trijų dalių: kliento, brokerio ir serverio. Kiekviena dalis vykdoma atskirame procese. Brokerio funkcijas iškviesti naudojama žiniatinklio paslaugų (WEB services) technologija. Klientas kviečia metodus, kurie priklauso sąsajai *SasajaKlientui*. Serveris kviečia funkcijas, kurios priklauso sąsajai *SasajaServeriui*. Brokeris realizuoja šias dvi sąsajas. Serveris realizuoja sąsają *PaslaugosSasaja*.

SasajaKlientui funkcijos

LoginUser() funkcija identifikuoja klientą ir prijungia jį prie sistemos.

RegisterUploadEcg() funkcija užregistruoja kardiogramą sistemoje jos nusiuntimui.

UploadEcg() funkcija nusiunčia kardiogramą į sistemą.

DownloadJobResult() funkcija parsiunčia kardiogramos analizės rezultatą.

GetUserEcgList() funkcija gauna vartotojui priklausantį kardiogramų sąrašą.

GetJobStatus() funkcija patikrina kardiogramos apdorojimo statusą.

SasajaServeriui funkcijos

LoginMachine() funkcija identifikuoja serverį ir prijungia jį prie sistemos.

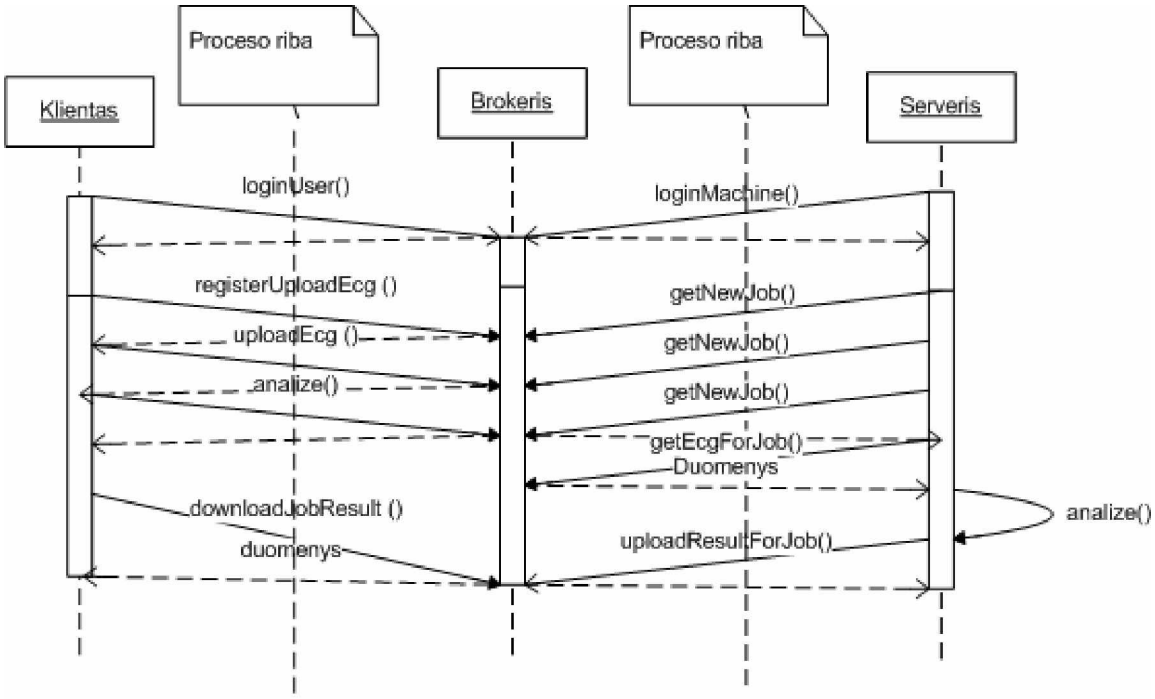
GetNewJob() funkcija gauna naują kardiogramos analizės užduotį

GetEcgForJob() funkcija parsiunčia kardiogramą analizei.

UploadResultForJob() funkcija nusiunčia kardiogramos analizę į brokerį.

PaslaugosSasaja funkcijos

Analize() funkcija vykdo kardiogramos analizę.



28 pav. Realizuoto šablono sekų diagrama

28 paveikslėlyje pavaizduota realizuoto šablono sekų diagrama. Diagramoje pavaizduotas toks scenarijus:

1. Klientas ir Serveris prisijungia prie brokerio iškviesdami *loginUser()* ir *loginMachine()* funkcijas;
2. Klientas užregistruoja kardiogramą, nusiunčia ją į brokerį ir inicijuoja analizę iškviesdamas tris funkcijas *registerUploadECG(), uploadECG()* ir *analize()*;
3. Serveris su funkcija *getNewJob()* tikrina ar yra jam paskirtų darbų;
4. Serveris iškviečia funkciją *getEcgForJob()* ir gauna kardiogramos duomenis;
5. Serveris įvykdo funkciją analize();
6. Serveris iškviečia funkciją *uploadResultForJob()* ir nusiunčia analizės rezultatą į brokerį;
7. Klientas iškviečia *downloadJobResult()* ir jei jau yra analizės rezultatas - gauna jį.

Teoriniame modelyje paslaugos funkcija *vykdytiFA()* yra vienalytė, tačiau realiai ji gali būti išskaidyta. Funkcijos *registerUploadECG(), uploadECG()* ir *analize()* atitinka teorinio modelio *vykdytiFA()* funkciją, išskaidytą į du etapus: pradinių duomenų nusiuntimą ir operacijų su jais inicijavimą.

Serveris, vykdydamas funkciją *ping()*, privalo periodiškai pranešinėti brokeriui apie tai, jog vis dar vykdo analizę. Tokiu būdu, brokeris gali nustatyti kokioje būsenoje yra serveris. Jeigu brokeris ilgesnį, negu susitarta, laiko tarpą negauna informacijos apie serverio būseną, darbas perduodamas kitam serveriui. Kiekviena kartą, kai koks nors serveris iškviečia funkciją *getNewJob()*, prašydamas brokerio, kad šis duotų duomenų analizei, patikrinama ar visi serveriai vykdantys analizę laiku pranešė apie savo būseną. Jeigu yra serveris kuris nepranešė savo būsenos, jo užduotis perduodama duomenų laukiančiam serveriui.

Galbūt įmanomi ir kiti užduočių paskirstymo tarp serverių būdai, tačiau praktiškai išbandytas tik paminėtas algoritmas.

IŠVADOS

- Nebuvo tinkamo projektavimo šablono, kuris tenkintų keliamus reikalavimus. Priešakinis valdiklis (*Front controller*) yra vienas iš problemos sprendimo būdų, tačiau jis nenumato paskirstytos sistemos architektūros, todėl kai vykdomi dideli skaičiavimai ir prie sistemos jungiasi daug vartotojų, sistema nesugeba greitai patenkinti klientų poreikių.
- Tam, kad realizuoti paskirstytą sistemą, nagrinėtas brokerio (*broker*) projektavimo šablonas, tačiau jis turi apribojimus. Brokerio projektavimo šablonas tiktų, jeigu nebūtų magistro darbe įvesto apribojimo, jog algoritmo programa (serveris) neturi išorinio IP adreso.
- Pasiūlytasis naujas brokerio projektavimo šablonas įgyvendina užduočių paskirstymą tarp daugelio serverių, sprendžia bendro sistemos našumo problemas. Serveriai neprivalo turėti išorinių IP adresų ir sistema gali būti plečiama.
- Esant mažam prisijungusių prie sistemos klientų skaičiui, elektroninės paslaugos teikimui tinkamesnis yra priešakinio valdiklio šablonas. Kai klientų skaičius auga, pasiūlytojo brokerio šablonas yra žymiai efektyvesnis.
- Realizuotas vienintelis užduočių paskirstymo tarp serverių algoritmas. Galbūt yra kitų būdų, kaip paskirstyti užduotis tarp serverių. Reikėtų juos ištirti ir palyginti su esamu.

LITERATŪRA

[1] Java Enterprise Design Patterns from Patterns in Java Volume 3 [interaktyvus]. Prieiga per internetą: <http://www.clickblocks.org/patterns1/pattern_synopses3.htm#Thread_Pool>

[2] Microsoft Corporation. Enterprise Architecture, Patterns and Practices [interaktyvus]. 2005. Prieiga per internetą: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_entdev.asp>

[3] Linda Rising AG Communications Systems. Patterns: A Way to Reuse Expertise [interaktyvus]. 2001. Prieiga per internetą: <http://members.cox.net/risingl1/articles/expertise.htm>

[4] Brad Appleton. Patterns in a Nutshell [interaktyvus]. Prieiga per internetą: <http://www.cmcrossroads.com/bradapp/docs/patterns-nutshell.html>

[5] Sun Microsystems, Inc. BluePrints Patterns [interaktyvus]. Prieiga per internetą: <http://java.sun.com/blueprints/patterns/>

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Santrumpa	Paaiškinimas
CORBA	Paskirstytų sistemų kūrimo technologija
HTML	Hyper Text Markup Language
LHPM	Lankstus hiperterpės proceso modeliavimas
Modeliavimas	Programinės įrangos projektavimo proceso dalis, kurioje kuriamas sistemos architektūros (loginės, fizinės, komunikavimo) modelis
Modelis	Elementų ir jų sąryšių visuma, apibūdinanti programinę įrangą tam tikru aspektu
Paketas	Programinės įrangos elementų (programų, dokumentų, bibliotekų ir t.t.) visuma, sudaranti ekslotacijai paruoštą programinės įrangos produktą
P2P	Peer to peer
Realizacija	Programos užrašymas tam tikra programavimo kalba ir jos suderinimas bei vykdomųjų elementų (failų) sukūrimas
Scenarijus	Vykdomasis specialių komandų rinkinys (vykdomas tiesiogiai secializuotų interpretatorių) tekstiniame formate.
Specifikacija	Detalus aprašymas.
MVC	Model-View-Controller
Šablonas	Elemento griaučiai, formavimo taisyklės ir pan.
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language