KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Donatas Beniušis

Nuotolinių programinių objektų paskirstytose sistemose tyrimas

Magistro darbas

Darbo vadovas
doc. dr. Vytautas Pilkauskas

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Donatas Beniušis

Nuotolinių programinių objektų paskirstytose sistemose tyrimas

Magistro darbas

Recenzentas
doc. dr. Robertas Damaševičius

2008-05-26

Vadovas
doc. dr. Vytautas Pilkauskas

2008-05-26

Atliko
IFM 2/2 gr. stud.
Donatas Beniušis
2008-05-26

Kaunas, 2008

TURINYS

Paveikslėlių turinys

Lentelių turinys

Research of distributed objects. Summary

Many of today's enterprise computing systems are powered by distributed object middleware. Such systems, which are common in industries such as telecommunications, finance, manufacturing, and government, often support applications that are critical to particular business operations. Because of this, distributed object middleware is often held to stringent performance, reliability, and availability requirements. Fortunately, modern approaches have no problem meeting or exceeding these requirements.

The goal of this research was to familiarize with middleware technology especially with CORBA, Microsoft .NET Remoting and Microsoft .NET Web Services. In experimental part was proved that the most efficient technology is Microsoft .NET Remoting with TCP protocol. That means our choice was correct during development of our project "Cool Test Tool".

1. Įvadas

1.1. Darbo aktualumas

Šiuo metu paskirstytos sistemos užima svarbią dalį informacinių technologijų (IT) srityje. Jos plačiai naudojamos ir yra sutinkamos įmonėse, mokslo įstaigose, namuose. Pagrindiniai tokių sistemų pavyzdžiai yra Web Server, Application Server. Tai yra tarpinė programinė įranga (angl. „Middleware“), kuri suriša atskiras programas, nepriklausomai nuo to, kokioje aplinkoje jos veikia.

Tarpinės PĮ sąvoka atsirado dar 1968 m. [Nic05], tačiau tuo metu buvo manoma, kad tai yra nereikalingas ir neturintis ateities dalykas. Praėjus dešimtmečiui buvo vystoma ši idėja. To meto sistemos buvo neefektyvios, griozdiškos, tačiau atlikdavo savo funkciją [VKZ05]. Pagrindinė jų paskirtis – bendravimas tarp naujų ir paveldėtų sistemų. 9-jame dešimtmetyje paskirstytų sistemų populiarumas išaugo, buvo sukurti Tarpinės PĮ sistemų standartai: OMG – CORBA, Microsoft – COM. CORBA ir COM technologijos turi nemažai panašumų: naudojami nuotolę programiniai objektai (angl. „Remote Objects“), kliento ir serverio moduliai, tarpininkai (angl. „Proxies“), Marshallers ir kiti [VSW02]. 2002 m. Microsoft pristatė naują platformą „.NET Framework 1.0“. Joje buvo nuotolinių objektų (angl. „Remote Objects“) ir Web Service komponentai, kurie palengvina darbą programuojant paskirstytas sistemas.

Pastaruoju metu kuriamos sistemos neišvengia tarpinės PĮ panaudojimo. Kadangi tarpinės PĮ realizavimui yra nemažai sukurtų technologijų, tai projektuotojams kyla problema, kurią iš jų pasirinkti.

1.2. Darbo tikslas

Magistrinio darbo projektinėje dalyje buvo sukurta nuotolinė žinių testavimo sistema „Cool Test Tool“. Ji realizuota remiantis kliento – serverio architektūra. Projektavimo metu buvo pasirinkta „Remote Objects“ technologija su TCP protokolu. Todėl šio darbo tikslas – išanalizuoti CORBA, Remote Objects ir Web Services veikimo principus, ištirti jų resursų išnaudojimą ir išrinkti efektyviausią technologiją, tinkančią projektui. Šios technologijos pasirinktos todėl, kad jas paprasta pritaikyti esamam projektui.

1.3. Darbo uždaviniai

Pagrindiniai uždaviniai yra:

- panagrinėti nutolusių programinių objektų bendrus aspektus;
- plačiau paanalizuoti CORBA, Microsoft .NET Framework Remote Objects ir Web Services technologijas;
- pateikti CORBA, Microsoft .NET Framework Remote Objects ir Web Services technologijų efektyvumo tyrimo eigą ir rezultatus.
- Aprašyti sukurtą nuotolinės žinių testavimo projektą „Cool Test Tool“.

2. Nuotolinių programinių objektų analizė

2.1. Bendra analizė

Analizės pradžioje bus aptariami bendri paskirstytų sistemų aspektai. Paskirstytos sistemos yra įvairios – jos skiriasi tiek savo paskirtimi, tiek technologija. Pastaruoju metu dauguma tokių sistemų yra specializuotos ir sudėtingos. Kaip pavyzdį paimkime internetą. Internetas apima daugybę paskirstytų sistemų: HTTP, FTP, SMTP, Telnet serveriai ir daug aptarnaujamų klientų. Visi šie serveriai naudoja savo protokolus, kurie yra griežtai apibrėžti. Paskirstytose sistemose išskiriamos probleminė (*problem–related*) ir savybių (*property–related*) sritys, kurios yra tarpusavyje susijusios [TS02].

- *Probleminė sritis* – tai yra natūrali problema, kylanti dėl elementaraus atstumo tarp kliento ir serverio. Pavyzdžiui, yra klientas, kurio kompiuteris Europoje, ir serveris, kuris yra JAV ir publikuoja Web puslapius. Pagrindinis tikslas – perduoti informaciją iš JAV į Europą. Tai ir yra probleminė sritis.
- *Savybių sritis* – tai paskirstytos sistemos tam tikros savybės, kurias galima tobulinti siekiant pagerinti sistemos darbą.

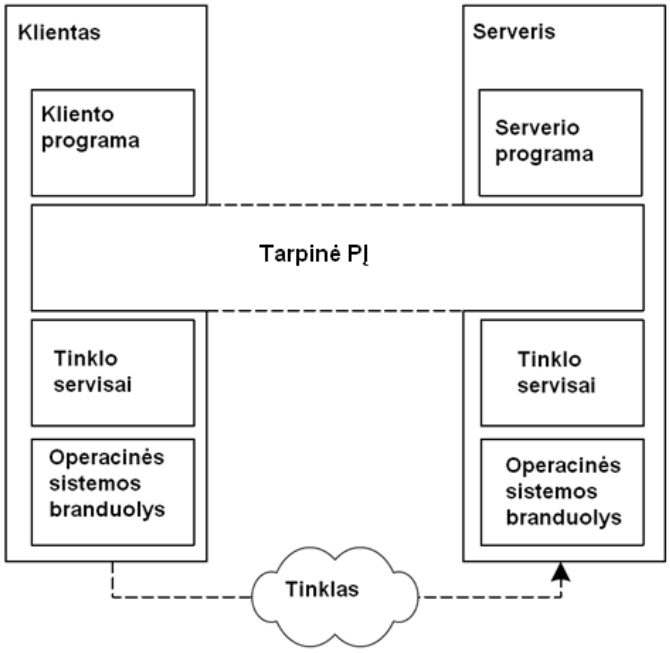
Keletas savybių srities pavyzdžių:

- *Našumas ir plečiamumas*. Jei sistema reikalauja didelių resursų, tai ją galima paskirstyti keliuose kompiuteriuose. Paskirsčius reikia papildomo proceso, kuris koordinuotų veikimą paskirstytoje sistemoje.
- *Klaidų toleravimas*. Tai ypač aktualu paskirstytose sistemose. Sistema turi būti tokia, kad iškilus programinei klaidai, klientai to nepastebėtų.
- *Serviso ir kliento vietos nepriklausomumas*.

- *Palaikymas ir priežiūra*. Paskirstytų sistemų palaikymas ir priežiūra yra sudėtingas ir daug kaštų reikalaujantis procesas. Todėl atliekant pakeitimus sistemoje reikia stengtis, kad tai kuo mažiau įtakotų funkcionalumą.
- *Saugumas*. Paskirstytose sistemose apsikeičiama informacija gali būti konfidenciali ir ji neturi būti pasiekiama pašaliniams asmenims.
- *Veikimas su kitomis sistemomis*. Dažnai būna, kad sistemą reikia įdiegti į jau esamą sistemą. Todėl jai turi būti būdingas integralumas.

Šiame darbe bus nagrinėjamas CORBA, Remote Objects ir Web Services technologijų našumas.

Paskirstytos sistemos remiasi tinklo protokolais. Pavyzdžiui sistema naudoja TCP/IP protokolą, kuriuo siuntinėjamos žinutės tiesiogiai naudojant *send* ir *receive* operacijas [Ste98]. Šioje vietoje iškyla viena problema: programuotojas turi išmanyti tinklo struktūrą ir veikimo principus. Todėl programavimas tampa sudėtingas procesas. Šiai problemai išspręsti naudojama tarpinė PĮ. Paveiksle parodyta kaip tarpinė PĮ paslepia tinklo servisus ir operacinę sistemą.

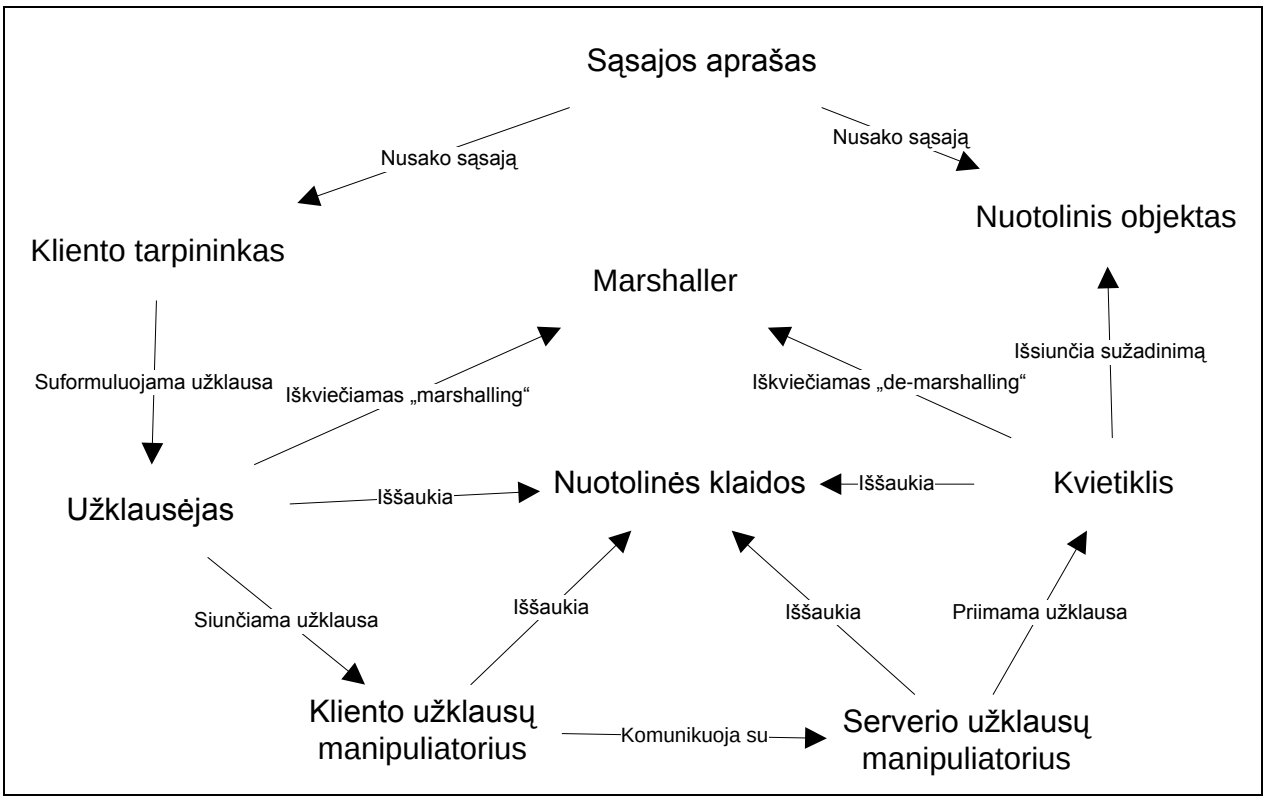


2.1 pav. Tarpinės PĮ vieta sistemoje

Tarpinė PĮ yra programinis sluoksnis tarp tinklo servisų ir kliento – serverio programų. Jis paslepia platformų heterogeniškumą. Kliento ir serverio programos negali apeiti tarpinės sluoksnio ir tiesiogiai naudotis tinklo servisais [KS08].

Tarpinė PĮ yra sudaryta iš blokų, dar kitaip vadinamų šablonų (angl. „patterns“):

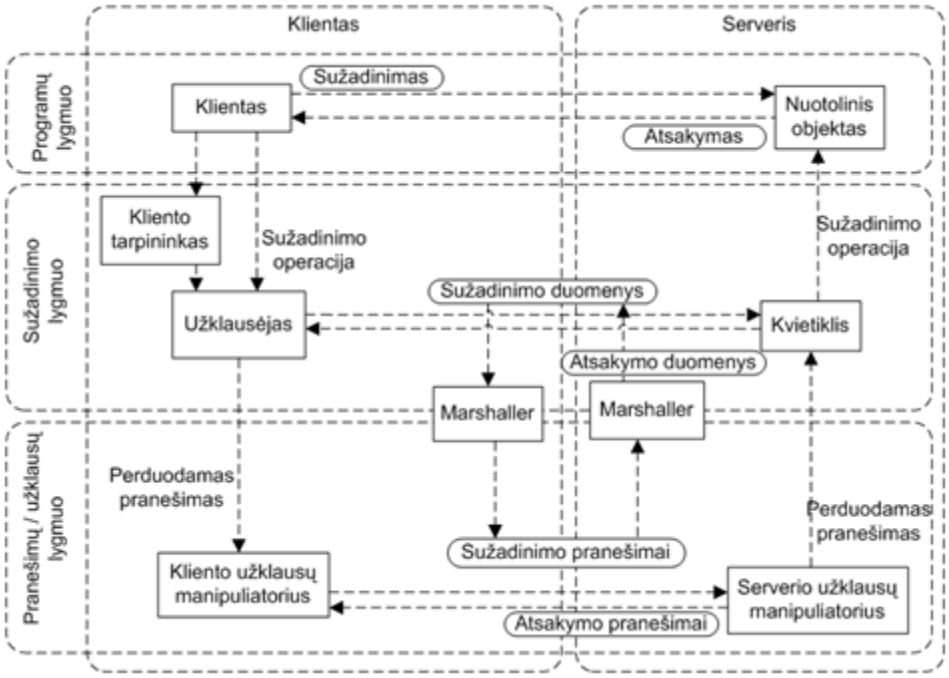
- Sąsajos aprašas.
- Kliento tarpininkas (angl. „Client Proxy“).
- Marshaller.
- Užklausėjas (angl. „Requestor“).
- Nuotolinės klaidos.
- Kliento užklausų manipuliatorius (angl. „Client Request Handler“).
- Serverio užklausų manipuliatorius (angl. „Server Request Handler“).
- Kvietiklis (angl. „Invoker“).
- Nuotolinis objektas.



2.2 pav. Tarpinės PĮ struktūra

Pagrindinis tikslas yra pasiekti nuotolinį objektą, esantį serveryje. Kliento kompiuteryje sužadinama operacija ir peradresuojama serveryje esančiam nutolusiam objektui. Visa tai atlieka „užklausėjas“. Jis suformuoja nuotolinio metodo iškvietimą iš kliento pusės panaudojus parametrus, tokius kaip nuotolinio objekto vieta, tipas, operacijos pavadinimas ir argumentai. Klientas tiesiogiai kreipiasi į „užklausėją“ arba „kliento tarpininką“. „Kliento tarpininkas“ yra

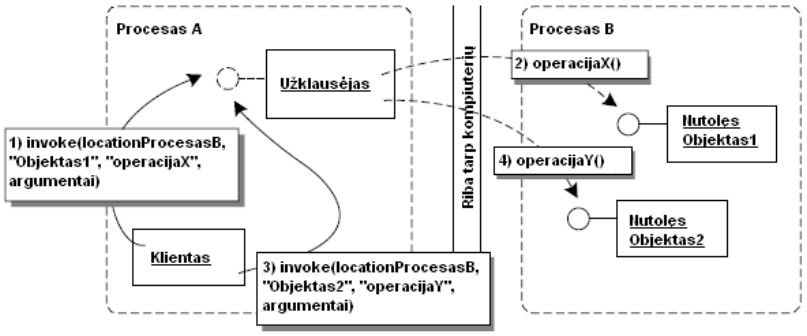
kliento vykdomajame procese ir turi tokią pačią sąsają kaip ir nutolęs objektas. Sąsaja apibrėžiama naudojant „sąsajos aprašu“. „Kliento tarpininkas“ naudoja „užklausėją“ nuotolinio objekto sužadinimui. „Užklausėjas“ naudojasi „kliento užklausų manipuliatoriumi“ tarptinklinio ryšio palaikymui. Serverio pusėje kliento iškvietimus priima „serverio užklausų manipuliatorius“ ir kai užklausa pilnai priimta, perduoda ją „kvietikliui“. Tarp kliento ir serverio apsikeičiama informacija turi būti serializuojama / deserializuojama. Tai atlieka „marshaller“. Jei kyla problemų su ryšiu arba jų iškyla pačiame serveryje, apie tai informuojamas „kliento užklausų manipuliatorius“ ir įspėjamas klientas naudojant „nuotolines klaidas“. Išsamesnė veikimo schema:



2.3 pav. Tarpinės PĮ detali struktūra

2.1.1. Užklausėjas

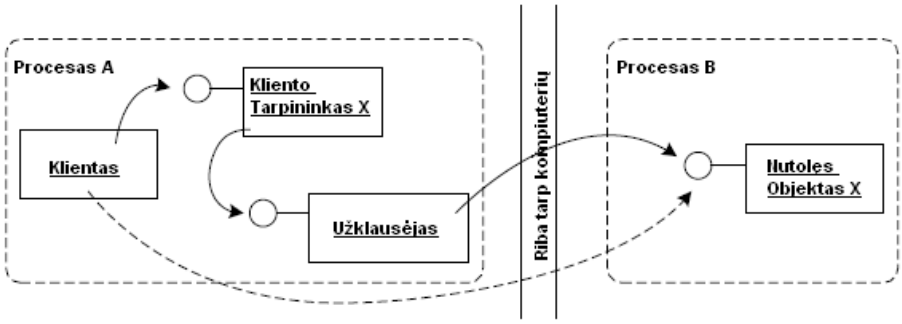
Kliento programoje „užklausėjas“ naudojamas susisiekti su nuotoliniu objektu. Jis turi tiesioginę nuorodą į nuotolinį objektą, operacijos pavadinimą ir argumentus. Pagal šiuos parametrus sukuriamas nuotolinis sužadinimas (angl. „invoke“), kuris nusiunčiamas nutolusiam objektui. „Užklausėjas“ paslepia detales apie kliento pusėje esantį susijungimą [SSRB00].



2.4 pav. Užklausėjo schema

2.1.2. Kliento tarpininkas

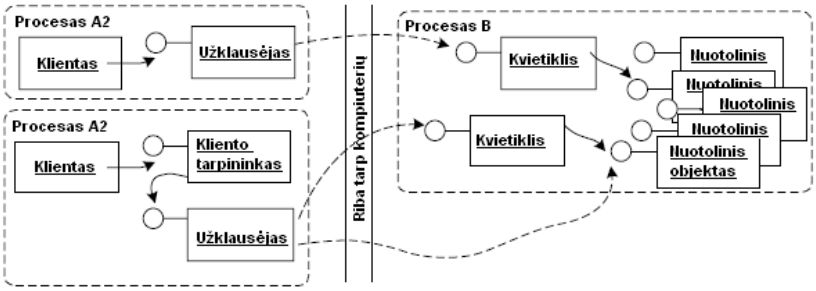
„Kliento tarpininkas“ palaiko tokią pačią sąsają kaip ir nutolęs objektas. Susisiejimui su nutolusiu objektu klientui pakanka kreiptis į „kliento tarpininką“. „Kliento tarpininkas suformuluoja sužadinimo užklausą, kuri keliauja „užklausėjui“. „Kliento tarpininkas“ gauna atsakymą iš „užklausėjo“ ir taip palaiko ryšį su nutolusiu objektu. Programuotojui naudoti „kliento tarpininką“ yra daug paprasčiau nei „užklausėją“, tačiau toks sprendimas veikia lėčiau ir reikalauja daugiau atminties.



2.5 pav. Kliento tarpininko schema

2.1.3. Kvietiklis

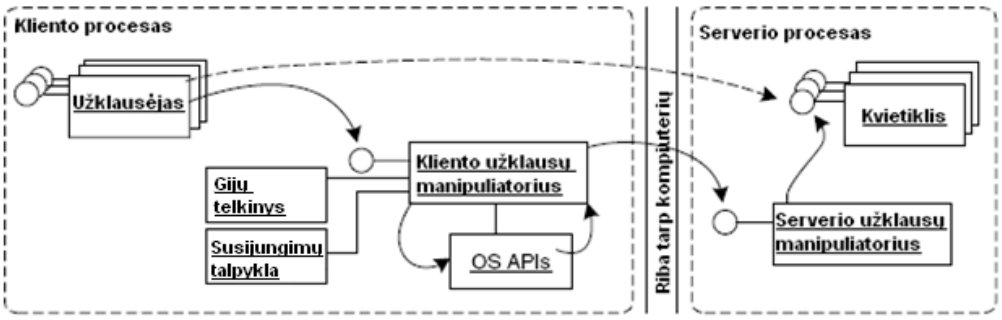
„Kvietiklis“ yra serveryje ir priima užklausas iš „užklausėjo“. „Užklausėjas“ siunčia nuotolinio objekto ID, operacijos pavadinimą ir jos parametrus. „Kvietiklis“ gavęs užklausą ją dešifruoja (angl. „demarshal“), nuskaito užklausto nutolusio objekto ID, operacijos pavadinimą su parametrais. Tada iškviečiamas užklaustas metodas su parametrais. „Kvietiklis“ pranešimus iš „užklausėjo“ gauna per „serverio užklausų manipuliatorių“. Taip pat ir siunčia atsakymą „užklausėjui“. „Kvietiklio“ naudojimas vietoj tiesioginio ryšio padeda sumažinti tinklo adresų aibes. Taip pat įgalina „serverio užklausų manipuliatoriui“ efektyviau dalinti ryšį esant keliems nutolusiems objektams.



2.6 pav. Kvietiklio schema

2.1.4. Kliento užklausų manipuliatorius

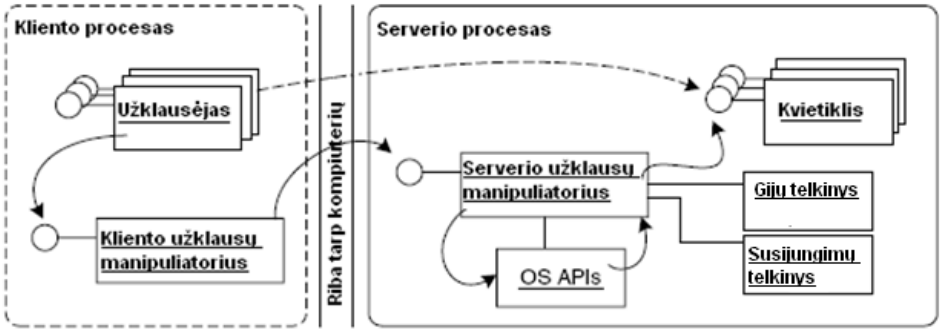
„Kliento užklausų manipuliatoriaus“ tikslas palaikyti ryšį su serveriu siunčiant jam užklausas ir priimant atsakymus iš serverio. Rezultatai perduodami „užklausėjui“. „Kliento užklausų manipuliatorius“ taip pat susitvarko su skirtuoju laiku (angl. „timeouts“), gijomis (angl. „threading“) ir klaidomis [SSRB00]. Susijungimų talpykla (angl. „Connection cache“) skirta pakartotiniam susijungimo su to paties serverio programa panaudojimui. Gijų telkinys (angl. „Thread pooling“) skirtas pakartotinai kuriant gijas [KJ04].



2.7 pav. Kliento užklausų manipuliatoriaus schema

2.1.5. Serverio užklausų manipuliatorius

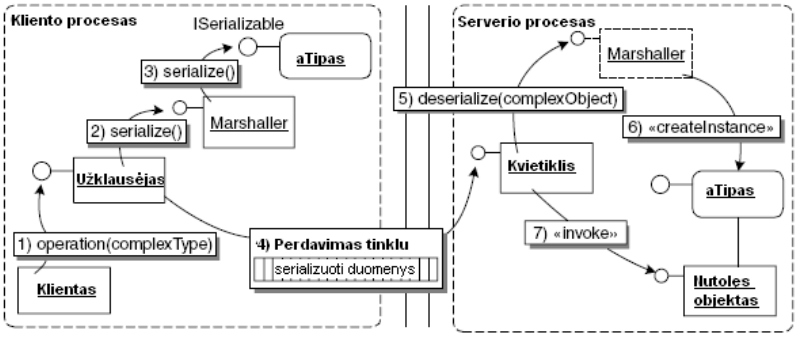
„Serverio užklausų manipuliatorius“ turi jam skirtą prievadą (angl. „port“) per kurį priimami pranešimai tinkle, pranešimų fragmentai sukombinuojami į vieną siųstą pranešimą, kuris perduodamas atitinkamam „kvietikliui“ tolimesniam vykdymui. Panašiai kaip ir „kliento užklausų manipuliatorius“ susijungimų resursų paskirstymo optimizavimui naudoja „gijų telkinį“ ir „susijungimų telkinį“ [KJ04].



2.8 pav. Serverio užklausų manipuliatoriaus schema

2.1.6. Marshaller

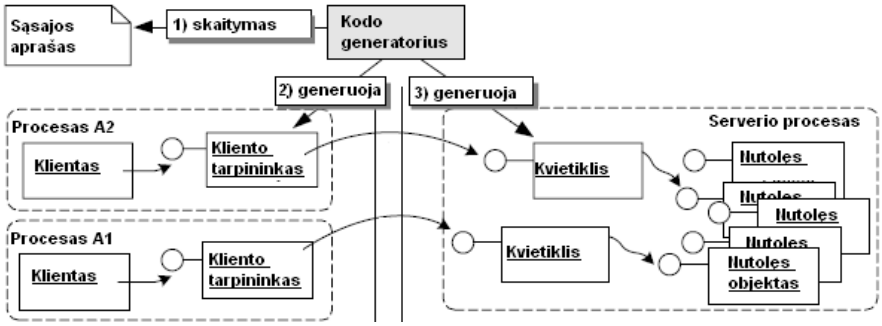
Kad objektus būtų galima persiųsti per tinklą kaip baitų srautą, jie visų pirma turi būti serializuojami. Kliento ir serverio kompiuteriuose panaudojus suderintus „marshaller“ objektus galima šifruoti ir dešifruoti (serializuoti ir deserializuoti) duomenis.



2.9 pav. Marshaller schema

2.1.7. Sąsajos aprašas

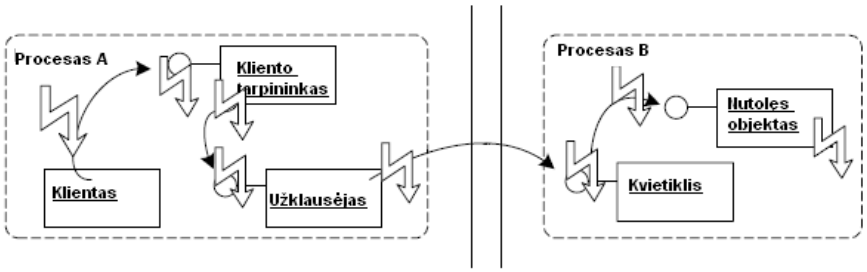
Klientas privalo žinoti kokius metodus su kokiais parametrais galima iškviesti serveryje. Tam naudojami sąsajų aprašai, kurie yra kliente ir serveryje. Pagal aprašą „kliento tarpininkas“ žino kokius metodus galima iškviesti, o „kvietiklis“ gavęs to metodo užklausą sužadina atitinkamą metodą nutolusiame objekte. Kadangi tarpinė PĮ palaiko skirtingas programavimo kalbas, tai reikalingas kodo generatorius, kuris iš sąsajos aprašo atitinkama kalba generuoja sąsają priimtina programavimo kalba. Sąsajos aprašas realizuojamas nepriklausoma kalbos sintakse.



2.10 pav. Sąsajos aprašo schema

2.1.8. Nuotolinės klaidos

„Nuotolinės klaidos“ reikalingos klaidų valdymui kliento pusėje. Nepriklausomai nuo to, kurioje sistemos dalyje įvyko klaida, klientas ją gali identifikuoti. Žinoma, ne visas klaidas reikia pranešti klientui. Pagrindinės jų yra susijusios su komunikavimo ar duomenų perdavimo problemomis.

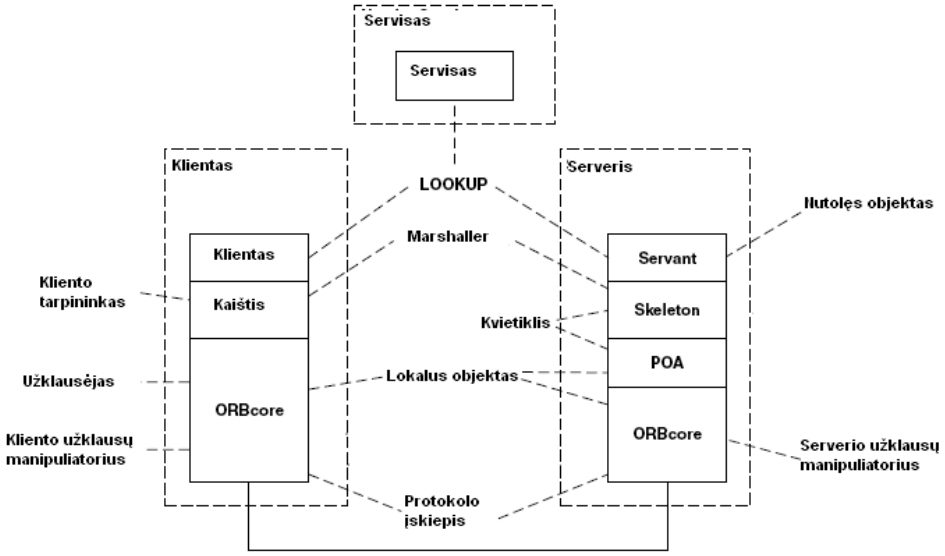


2.11 pav. Nuotolinių klaidų schema

2.2. CORBA

CORBA yra viena iš pirmųjų tarpinės PĮ technologijų. Pirmoji galutinė versija pasirodė 1991 metais. Iki šių dienų jau yra išleista nemažai versijų. Oficialių produktų sąrašą galima rasti OMG grupės tinklalapyje: http://www.omg.org/technology/corba/ . Tyrimo metu bus naudojama IIOP.NET distribucija, kuri naudoja savo IIOP protokolą ir palaikoma Microsoft .NET Framework.

Sekanti diagrama atvaizduoja kaip panaudojami minėti šablonai CORBA architektūroje:



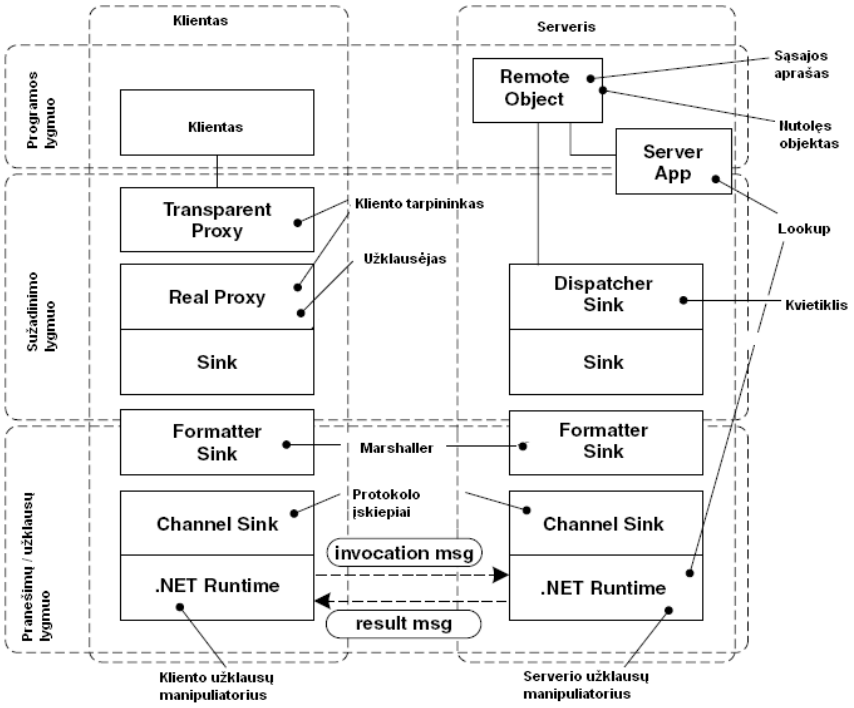
2.12 pav. CORBA diagrama

Pagrindiniai CORBA komponentai:

- Object Request Broker branduolys (ORBcore) – atsakingas už susijungimo tarp serverio ir kliento mechanizmą.
- Portable Object Adapter (POA) – atsakingas už užklausų perdavimą nutolusiam objektui.
- Servisas – atsakingas už sąryšį su nutolusiu objektu.

2.3. Microsoft Remote Objects

Microsoft .NET tapo pakaitalu Windows 32-bit API programavimo kalbai, o paskirstytoms sistemoms – DCOM, kuris yra COM+ dalis, pakaitalu [Mic08]. Reikia atkreipti dėmesį, kad .NET platforma nepakeitė senesnių programavimo kalbų, tiesiog ji sukurta jų pagrindu. Sekanti diagrama atvaizduoja kaip panaudojami minėti šablonai Microsoft .NET Remoting architektūroje:



2.13 pav. Microsoft Remote Objects diagram

Pagrindiniai Remote Objects komponentai:

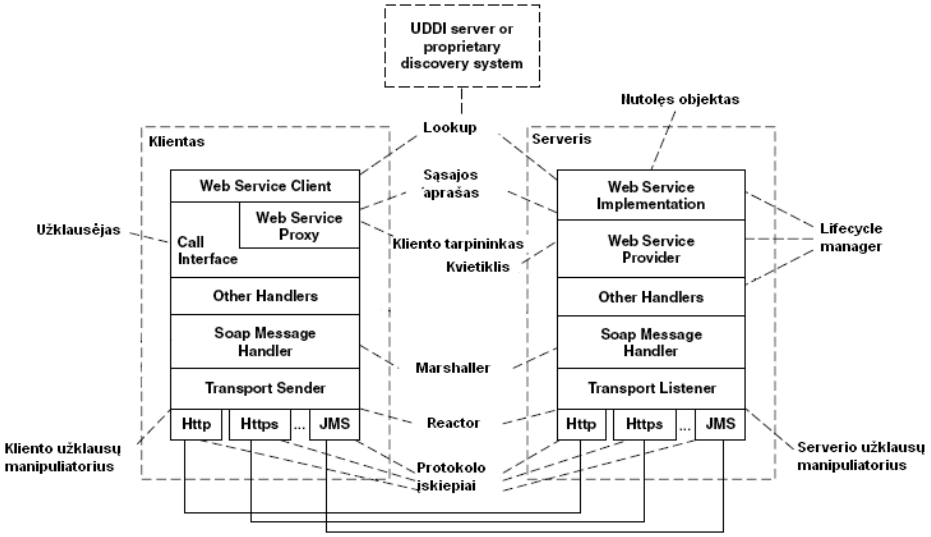
- Server App – serveryje veikianti programa, kuri padaro nutolusį objektą prieinamą tinkle.
- Transparent Proxy – komponentas, turintis tokią pačią sąsają, kaip ir nutolęs objektas.
- Real Proxy – tiesiogiai kliento programai neprieinamas komponentas, kuris priima sužadinimo komandas iš „Transparent Proxy“ ir perduoda juos tinklu.
- Dispatcher sink – sukombinuotų pranešimų talpykla kartu su „kvietikliu“.
- Sink – siunčiamų ir priimamų pranešimų talpykla.
- Formatter Sink – serializuoja pranešimus, kurie bus perduodami tinkle.
- Channel Sink – atsakingas už serializuotų pranešimų perdavimą tinkle.
- NET Runtime – Microsoft .NET technologijos platforma.

2.4. Microsoft Web Services

Web Service yra standartizuotas ir servisais paremtas komunikavimas tarp paskirstytos sistemos programinės įrangos. Programos gali veikti įvairiose aplinkose, būti suprogramuotos skirtingomis programavimo kalbomis. Web Service turi sąsają, kuri apibrėžta „mašinos vykdomuoju“ formatu (paprastai WSDL). Kitos sistemos sąveikauja pranešimais (paprastai

SOAP pranešimai). Pranešimai dažniausiai perduodami HTTP protokolu naudojant XML serializaciją [BHM+03].

Sekanti diagrama atvaizduoja kaip panaudojami minėti šablonai Web Service architektūroje:



2.14 pav. Microsoft Web Services diagrama

Pagrindiniai Web Service komponentai:

- Web Service Client – pagrindinė sąsaja per kurią klientas bendrauja su nutolusiu objektu.
- Call Interface – užklausėjas.
- Web Service Proxy – kliento tarpininkas su nutolusio objekto sąsajos aprašu.
- Other Handlers – manipuliatorių grandinė, per kurią keliauja sužadinimo komanda, kol ji pasiekia nutolusį objektą.
- SOAP Message Handler – komponentas, kontroliuojantis pranešimų perdavimą.
- Transport Sender – kliento komponentas, stebintis tinklo įvykius ir atitinkamai į juos reaguojantis.
- Web Service Implementation – nutolęs objektas.
- Web Service Provider – kvietiklis.
- Transport Listener – serverio komponentas, stebintis tinklo įvykius ir atitinkamai į juos reaguojantis.

3. Projektas : nuotolinė žinių testavimo sistema “Cool Test Tool”

Sistemos “Cool Test Tool” paskirtis: automatizuoti ir palengvinti žmogaus žinių testavimo darbą. Programa labiausiai tinkama mokymo įstaigoms, pvz., mokykloms, universitetams, taip pat didelėms įmonėms, kurioms svarbu nuolat kelti ir tikrinti darbuotojų kvalifikaciją. Sistema puikiai tinka ir darbuotojų paieška bei atranka užsiimančioms įstaigoms.

Pagrindinės sistemos funkcijos:

- mokymosi medžiagos saugojimas ir pateikimas;
- žinių patikrinimo testų kūrimas ir administravimas;
- testų vykdymas ir žinių vertinimas;
- vartotojų administravimas;
- vartotojų bendravimas tarpusavyje (diskusijos, klausimai-atsakymai ir pan.).

Rinkoje egzistuoja daug panašios paskirties produktų, tačiau dauguma jų turi ne visas aukščiau išvardytas savybes arba jų galimybės yra labai ribotos. Sukurta sistema turi visas pagrindines nuotoliniam mokymuisi bei žinių testavimui reikalingas funkcijas, yra lengvai pritaikoma specifiniams poreikiams.

Projekto užsakovas yra Kauno Technologijos Universiteto Verslo Informatikos Katedra, atstovas: doc. Vytautas Pilkauskas.

Projekto vykdytojai yra Kauno Technologijos Universiteto studentai Tomas Valinčius ir Donatas Beniušis.

Projekto reikalavimai buvo parengti naudojantis jau egzistuojančia sistema, sukurta projekto vykdytojų bakalaurinio darbo metu, bei atlikus esamų produktų bei rinkos analizę. Išnagrinėta keletas egzistuojančių panašių produktų, išsiaiškintos jų problemos bei stipriosios pusės, į tai atsižvelgta projektuojant ir specifikuojant kuriamą sistemą.

3.1. Žinių testavimo sistemos taikymo sritis

3.1.1. Projekto tikslas ir adresatas

Pagrindinis projekto kūrimo tikslas – sukurti universalią nuotolinio mokymosi bei žinių testavimo sistemą, kuri būtų lengvai pritaikoma specifiniams vartotojų poreikiams. Potencialūs sistemos vartotojai: mokymo įstaigos, pvz., mokyklos, universitetai, įmonės, kurios nuolat kelia bei tikrina savo darbuotojų kvalifikaciją, taip pat įmonės, užsiimančios darbuotojų paieška bei atranka.

3.1.2. Problemos sprendimas pasaulyje

Pasaulyje elektroninių žinių pateikimo ir testavimo įrankių yra daug. Pavyzdžiui, Microsoft sertifikatų laikymo testai, kelių eismo taisyklių testai, mokymo įstaigose atliekami testai ir t.t. Šie įrankiai yra specializuoti, nors atlieka panašią funkciją.

Kompiuteriniam raštingumui vertinti naudojama Europoje vieninga sistema "ECDL" (European Computer Driving Licence Foundation).

Komercinių įrankių srityje yra gerai žinomi „eSkill Corporation" produktas „eSkill", Censeo KnowledgeTrack™, „e-validator", „FirstClass".

Darbuotojų žinių patikrinimui yra atskirų įrankių, pavyzdžiui „Hemingway's Testing".

"ECDL" (European Computer Driving Licence Foundation)

[http://www.ecdl.lt/modules/tinycontent/index.php?id=2]

„ECDL" yra ne pelno siekianti įstaiga, kurios tikslas padėti žmonėms įgyti kompiuterinių žinių ir atverti kelią į informacinę visuomenę. „ECDL" standartas remiasi tuo, ką kompiuterio vartotojas turi žinoti apie informacijos technologiją ir asmeninius kompiuterius bei kokius asmeninių kompiuterių ir populiariausios jų taikomosios programinės įrangos panaudojimo įgūdžius jis turi įgyti. „ECDL" standarto numatytos būtinos žinių sritys ir įgūdžių grupės yra aprašytos „ECDL" programoje. „ECDL" programos tikslas – išvardinti faktus, kuriuos reikia žinoti, bei įgūdžius, kuriuos reikia įgyti pagal standarto reikalavimus. „ECDL" programoje nekalbama apie mokymo metodus, technologiją ir programas. Programoje numatyti vieningi visai Europai kompiuterinio raštingumo egzaminų reikalavimai ir tų egzaminų laikymo tvarka.

„ECDL" programa yra plačiai naudojama, tačiau jos pagalba vertinti galima tik konkrečios srities žinias, t.y. tik kompiuterinio raštingumo.

eSkill

[http://www.eskill.com/]

„eSkill" – tai universalus, laisvai platinamas žinių patikrinimo įrankis. Testai vykdomi ir administruojami interneto naršyklėje, kas yra patogu vartotojui. Taip pat yra testų administravimo įrankis. Žinių patikrinimo testai yra skirstomi pagal grupes, sudėtingumą. Šis įrankis neturi mokymosi medžiagos, nekontroliuoja testo vykdymo laiko, taip pat naudojami tik į pačią sistemą integruoti žinių patikrinimo testai, t.y. nėra galimybės susikurti savo testo. Sistemą iš dalies galima naudoti ir kaip pagalbinį mokymosi įrankį – programa gali paštu atsiųsti detalius neatsakytų klausimų aprašymus su paaiškinimais.

Hemingway's Testing

[http://hemingwaysoftware.com/index.htm]

„Hemingway‘s Testing“ – tai įrankis, skirtas darbuotojų kvalifikacijai patikrinti. Jame yra galimybė pasirinkti testų sritį, pavyzdžiui, rašybos, Microsoft Office programų išmanymą. Žinių patikrinimas atliekamas tik vietiniame kompiuteryje, nors yra numatyta galimybė ateityje tai atlikti nuotoliniu būdu. Visgi šis įrankis turi trūkumų: nėra galimybės administruoti testus, teikti mokomąją medžiagą. Sistema nesuteikia galimybės patiems vartotojams susikurti testus – tai gali atlikti tik programos autoriai už papildomą kainą.

Censeo KnowledgeTrack™

[http://www.censeocorp.com/solutions/knowledge.asp]

Censeo KnowledgeTrack™ suteikia galimybę patikrinti darbuotojų žinias. Sistemoje yra testų administravimo įrankis, tačiau nėra galimybės sukurti savo žinių patikrinimo testų. Žinių patikrinimas atliekamas naršyklėje. Šis įrankis neturi mokymosi medžiagos, nekontroliuoja testo vykdymo laiko. Sistemoje naudojami tik tekstiniai klausimų aprašai, todėl jos galimybės yra labai ribotos palyginus su kitais įrankiais. Visgi programa yra patraukli tuo, kad yra paprasta naudoti. Ji yra tinkamas pasirinkimas nedidelių poreikių testams atlikti. Pagrindiniai sistemos privalumai: ataskaitų apie atliktus rezultatus generavimas, analizė pagal vartotojų grupes, rezultatų publikavimas tinkle.

e-validator

[http://www.e-validator.com/evskills.html]

e-validator – tai nuotolinė darbuotojų žinių patikrinimo sistema. Ji gali būti įdiegta įmonės internetinėje svetainėje. Šis įrankis suteikia galimybę organizuoti žinių patikrinimo kalendorių, nustatyti testų laikymų skaičių, organizuoti savo testų komplektus, taip pat automatiškai apskaičiuoja darbuotojo žinių įvertį. e-validator trūkumas yra tas, kad naujų žinių patikrinimo testų sudarymui reikia sistemos kūrėjų įsikišimo. Pats vartotojas negali tiesiogiai sukurti ir publikuoti savo žinių patikrinimo testo.

Be įprastinių klausimų-atsakymų testų sistema gali simuliuoti programinės įrangos darbą, o testų vykdytojas – darbą su simuliuojama programa. Sistema gali skaičiuoti, kaip greitai vartotojas gali spausdinti, surinkti ir įvesti prašomus duomenis. Sistema turi galimybę simuliuoti bendravimą su kitais asmenimis, pvz., gali simuliuoti pokalbį su klientu, kuris prašo tam tikros informacijos arba skundžiasi teikiamų paslaugų kokybe. Programinė įranga vertina, kaip greitai vartotojas randa reikiamą informaciją, ar tiksliai pateikia paieškos rezultatus.

Sistemos archyve yra daugybė specializuotų testų: matematikos, literatūros, gramatikos, programavimo kalbų žinojimo, vadybos ir marketingo, internetinių technologijų ir kt.

FirstClass

[www.firstclass.com]

FirstClass – tai labai plačios paskirties programinis paketas, kurio pagrindinis tikslas – įgalinti individualius vartotojus ir jų grupes dirbti komandoje. FirstClass geriausiai pritaikyta mokslo įstaigų poreikiams bei įmonėms. Šiuo metu Lietuvoje šią sistemą naudoja Vytauto Didžiojo universitetas.

Pagrindinės sistemos galimybės:

- **Nuotolinio mokymosi modulis.** Studentai gali lengvai pasiekti mokymosi medžiagą, kuri gali būti publikuojama pačiais įvairiausiais formatais: vaizdo, garso, tekstinių ir kitų tipų bylų pavidalu. Studentai gali siųsti savo projektus, namų darbus bei kitą medžiagą ir taip atsiskaityti už reikiamus darbus.
- **Patobulintas mokymosi procesas.** Viena iš sistemos svarbiausių galimybių – tobulinti mokymo procesą. Dėstytojai turi galimybę sukurti specialias konferencijas, diskusijų forumus, projektus, kuriuose gali dalyvauti kiekvienas vartotojas. Studentai tarpusavyje gali bendrauti, dalintis darbais, kurti darbų galerijas.
- **Mokymasis realiu laiku.** Studentai, kurie negali dalyvauti paskaitose, dėstomose klasėse tam tikru laiku, gali gauti visą reikiamą medžiagą sistemos pagalba. Sistema leidžia dėstytojui bei studentui bendrauti realiu laiku, dėstytojui tikrinti namų darbus.

Įmonėms šis įrankis suteikia galimybę turėti naujausią informaciją visada šalia, bendrauti su kolegomis realiu laiku nepriklausomai nuo jų fizinės vietos. Darbuotojai gali gauti informaciją apie planuojamas atostogas, apie turimus įmonės produktus ir panašiai.

Sistema sukurta 1990 metais ir nuolat tobulinama. Taigi sistema turi daug galimybių, yra saugi ir patikima naudoti. Saugumui užtikrinti naudojami užkoduoti ryšio kanalai, yra galimybė vidinius sistemos duomenis saugoti bei filtruoti integruota Symantec antivirusine programa. Sistema pritaikyta beveik visoms operacinėms sistemos, taip pat ir delniniams kompiuteriams.

3.1.3. Situacijos Lietuvoje įvertinimas

Surinkti tikslių duomenų apie tai, kokią programinę įrangą naudoja Lietuvos universitetai ir mokyklos, projekto vykdytojai neturėjo galimybių – į prašymus suteikti tokią informaciją nebuvo atsakyta.

Žinoma, kad Vytauto Didžiojo Universitete naudojama FirstClass sistema, Kauno Technologijos Universitete kuriama ir naudojama TestTool sistema. Vilniaus universitetui buvo pristatyta bakalaurinio darbo metu sukurta sistemos versija, sistema buvo puikiai įvertinta, tačiau jai trūko ne mažai savybių, pvz., nuotolinio mokymosi galimybės. Vilniaus Universitetas yra potencialus kuriamos sistemos pirkėjas.

Situacija Lietuvos mokyklose beveik nežinoma, tačiau galima numanyti, kad čia automatizuoti žinių patikrinimo būdai ir nuotolinis mokymasis retai naudojami. Šiuo metu plėtojantis internetinėms technologijoms, augant kompiuterių skaičių klasėms ir dėmesiui informatikai, Lietuvos mokyklos yra potencialiausios kuriamos sistemos pirkėjos.

3.2. Nuotolinės žinių testavimo sistemos apibūdinimas

3.2.1. Programų sistemos funkcijos

Sistemą sudaro keturios dalys: testų vykdytojo programa, testų administratoriaus programa, serverio administratoriaus programa, vartotojų programas aptarnaujančios serverio programos (Windows servisai). Priklausomai nuo vartotojo tipo, sistema atlieka skirtingas funkcijas.

Testų vykdytojo funkcijos

Testų vykdytoju gali būti studentas, moksleivis ar bet koks kitas asmuo, kurio žinios bus tikrinamos arba kuris naudosis serveryje patalpinta mokymosi medžiaga.

- **Peržiūrėti mokymosi medžiagą.** Prisijungęs vartotojas gali peržiūrėti mokymosi medžiagą, atsisiųsti reikiamą informaciją įvairių bylų pavidale ar tiesiog naudotis informacija (skaityti, žiūrėti, klausytis).
- **Bendrauti su kitais vartotojais.** Prisijungęs vartotojas gali dalyvauti diskusijose, užduoti klausimus, atsakyti į juos ir pan.
- **Vykdyti testą.** Vartotojas atsakinėja į užduodamus klausimus pasirinkdamas vieną ar kelis teisingus atsakymus arba įvesdamas reikiamą frazę. Testo vykdymui yra skirtas

laikas, per kurį vartotojas turi spėti atsakyti į klausimus. Testo pabaigoje parodomas testo įvertinimas.

- **Keisti savo duomenis.** Vartotojas gali keisti savo prisijungimo prie sistemos duomenis: prisijungimo vardą ir slaptažodį.

Testų administratoriaus funkcijos

Testų administratoriumi laikomas asmuo, administruojantis testus. Jis gali kurti, redaguoti testus, priskirti juos tam tikroms vartotojų grupėms laikymui, kurti, talpinti bei administruoti mokymosi medžiagą.

- **Kurti testą.** Testų administratorius sukuria klausimus, atsakymus, pažymi teisingus variantus. Sukurtą testą priskiria tam tikrai vartotojų grupei ar atskirai konkretiems vartotojams, kurie laikys testą.
- **Redaguoti testą.** Galimybė pakeisti jau sukurto testo duomenis: klausimus, atsakymus, testo parametrus.
- **Peržiūrėti laikytus testus.** Galimybė peržiūrėti informaciją, kaip vartotojas laikė testą, t.y. pateiktų klausimų sąrašą, pasirinktus atsakymų variantus, sistemos apskaičiuotus įvertinimus.
- **Administruoti testų laikymo leidimus.** Galimybė konkrečiam testo vykdytojui arba jų grupei suteikti/atimti teisę laikyti testą(-us). Taip pat galimybė nurodyti/keisti tam tikrus testo laikymo parametrus: laikotarpį, galimų laikymų skaičių.
- **Keisti laikytų testų įvertinimus.**
- **Administruoti mokymosi medžiagą.** Testų administratorius galės kurti/redaguoti mokymosi medžiagą, ją talpinti, valdyti prieinamumo teises.

Serverio administratoriaus funkcijos

Serverio administratorius yra asmuo, atsakingas už serverio programų darbą. Jis taip pat gali atlikti visus testų administratoriaus veiksmus, administruoti sistemos vartotojus: testų vykdytojus, testų administratorius. Šis vartotojas atsakingas už prisijungimo duomenų prie duomenų bazės ir ftp serverio administravimą, atsarginių duomenų kopijų darymą ir pan.

- **Sistemos vartotojų administravimas.** Galimybė kurti, redaguoti bei šalinti sistemos vartotojus.

- **Serverio programų administravimas.** Vartotojus aptarnaujančių Windows servisų valdymas: paleidimas, sustabdymas, šalinimas. Prisijungimo duomenų prie ftp serverio ir duomenų bazės valdymas. Atsarginių duomenų kopijų darymas.

3.2.2. Sistemos kontekstas

Sukurtas produktas skirtas tik Microsoft Windows šeimos operacinėms sistemoms, pradedant MS Windows 2000. Produktas realizuotas Microsoft .NET pagrindu, todėl testų administratorių ir vykdytojų programoms veikti reikalingas Microsoft .NET Framework paketas. Sistemos duomenų serveryje turi veikti duomenų bazių valdymo sistema MySQL.

3.2.3. Vartotojo charakteristikos

Sukurtą sistemą buvo stengiamasi padaryti kuo patogesnę, paprastesnę ir intuityvesnę naudoti. Testų administratoriai ir testų vykdytojai turėtų turėti pagrindines naudojimosi kompiuteriu žinias. Sistemai įdiegti ir serverio programai bei duomenų bazių valdymo sistemai administruoti reikalingos pažengusio kompiuterio vartotojo žinios. Sistemos administratoriumi galėtų būti įstaigos darbuotojas, atsakingas už kompiuterių tinklą.

3.2.4. Vartotojo problemos

Kuriamas produktas įgalins mokymo įstaigos darbuotojus greitai ir automatizuotai įvertinti testų vykdytojų žinias, nebereikės skirti daug laiko atliktų testų taisymui bei vertinimui. Testų vykdytojai galės bet kada ir iš bet kurios vietos pasiekti mokymosi medžiagą, bendrauti virtualioje erdvėje, užduoti svarbius klausimus ir gauti reikiamus atsakymus. Sistema taupys jos vartotojų darbo laiką.

3.2.5. Vartotojo tikslai

Pagrindiniai visų vartotojų tikslai:

- patogus ir suprantamas sistemos naudojimas;
- informacijos pasiekiamumas visur ir visada.

Pagrindiniai mokymo įstaigos ar žinias tikrinančios įstaigos darbuotojų tikslai:

- žinių patikrinimo testų parengimas naudojantis daugialypės terpės priemonėmis (vaizdo ir garso medžiaga, paveikslėliai);
- testų vykdymo teisių administravimas;
- automatinis testų įvertinimas;

- mokymosi medžiagos parengimas ir talpinimas;
- vartotojų bendravimas tarpusavyje.

Žinias tikrinančių asmenų tikslai:

- mokymosi medžiagos pasiekiamumas visur ir visada;
- galimybė laikyti testus bet kurioje interneto ryšį turinčioje vietoje sau patogiu laiku;
- bendravimas tarpusavyje.

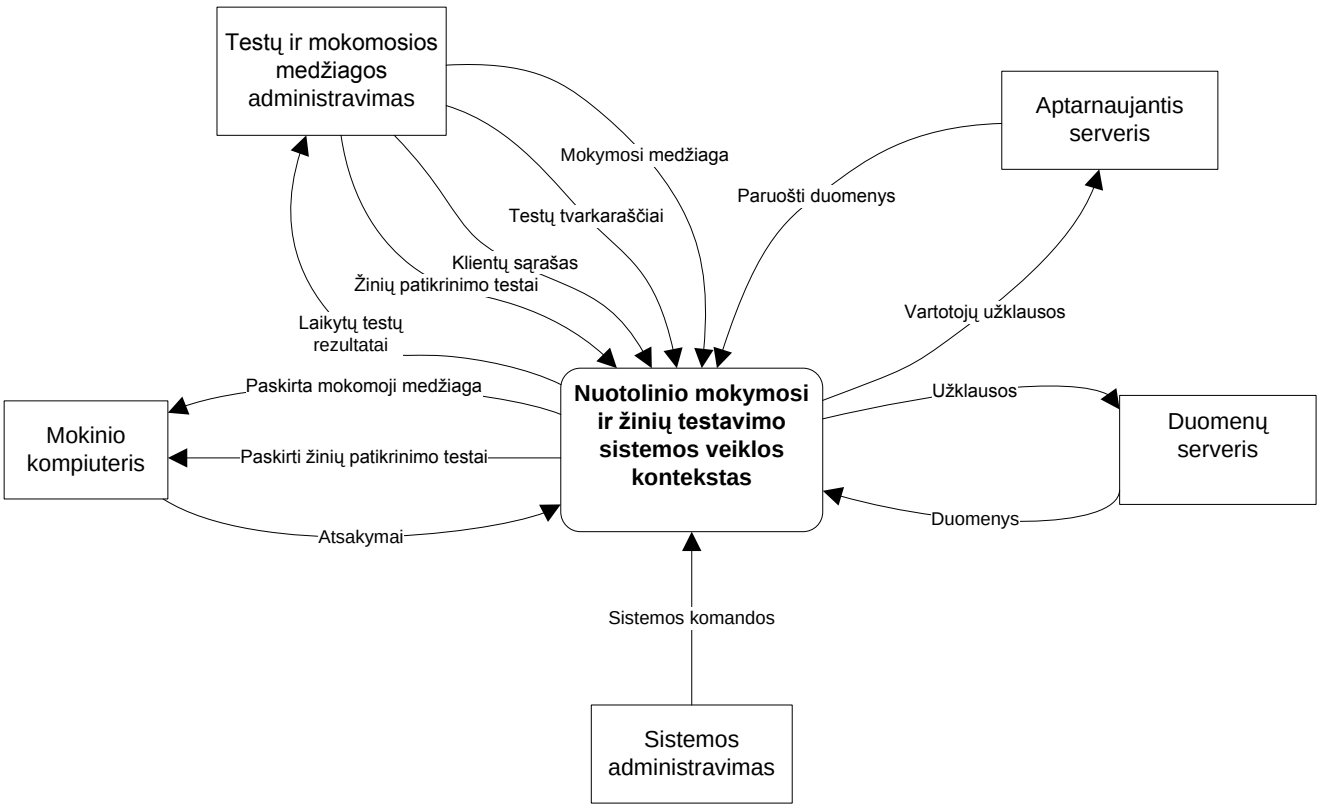
3.2.6. Bendri apribojimai

Sistema turi susidoroti su dideliu vienu metu prisijungusių vartotojų skaičiumi. Sistemos reakcijos laikas turi būti labai mažas. Tai ypač svarbu todėl, kad paprastai testų vykdymo laikas yra ribojamas laike. Be to, sistema turi tausoti kompiuterio resursus, kad ja galėtų naudotis įvairios konfigūracijos kompiuterius turintys asmenys.

3.3. Funkciniai reikalavimai

3.3.1. Veiklos sudėtis

Projekto veiklos kontekste vaizduojamos sistemoje dominuojančios pagrindinės veiklos bei tarp jų perduodami duomenų srautai:



3.1 pav. Veiklos kontekstas

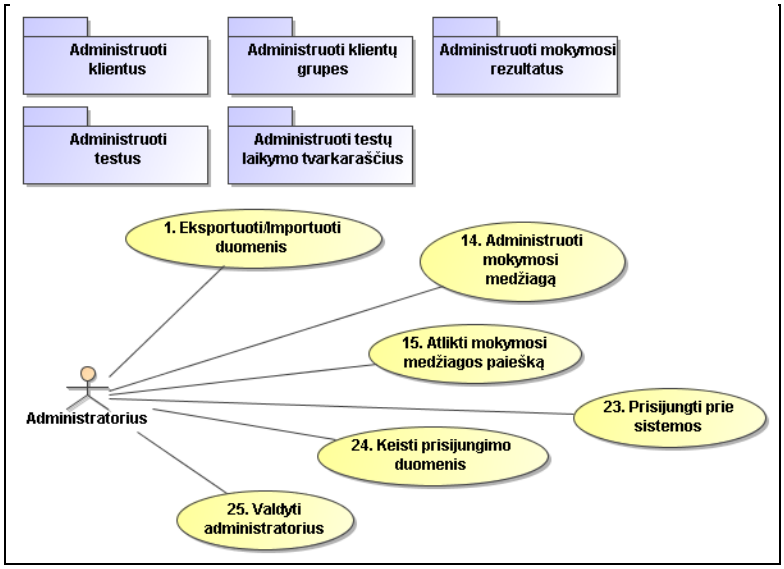
Toliau lentelėje pateikiami pagrindiniai sistemos įvykiai ir su jais susiję informacijos srautai:

3.1 lentelė. Veiklos padalinimas

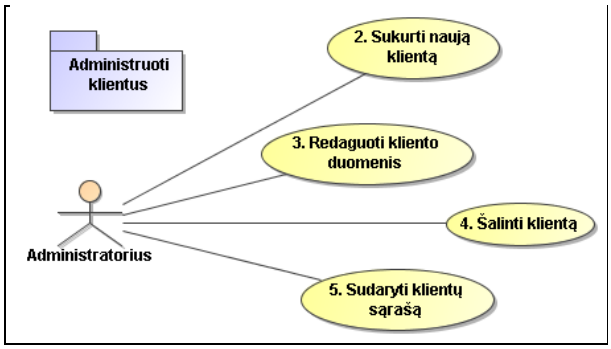
Įvykio pavadinimas	Įeinantys informacijos srautai	Išeinantys informacijos srautai
Administratorius paruošia/modifikuoja mokymosi medžiagą	Mokymosi medžiaga	
Administratorius sudaro/modifikuoja žinių patikrinimo testus	Žinių patikrinimo testai	
Administratorius sudaro/modifikuoja testų laikymo tvarkaraščius	Testų tvarkaraščiai	
Administratorius sudaro/modifikuoja klientų (kliento programos vartotojų) sąrašus	Klientų sąrašas	
Administratorius peržiūri testų laikymo rezultatus		Laikytų testų rezultatai
Administratorius keičia sistemos apskaičiuotą laikyto testo įvertinimą	Laikyto testo įvertinimas	Laikyto testo rezultatai
Aptarnaujančio serverio paruošti duomenys vartotojui	Paruošti duomenys	
Vartotojų užklausos aptarnaujančiam serveriui		Vartotojų užklausos
Užklausos duomenų serveriui		Užklausos
Duomenys aptarnaujančiam serveriui	Duomenys	
Aptarnaujančio serverio ir duomenų serverio valdymo komandos	Sistemos komandos	
Paruošta mokomoji medžiaga mokiniui		Paskirta mokomoji medžiaga
Paruošti testai mokiniui		Paskirti žinių patikrinimo testai
Mokinio testų atsakymai	Atsakymai	

3.3.2. Sistemos sudėtis

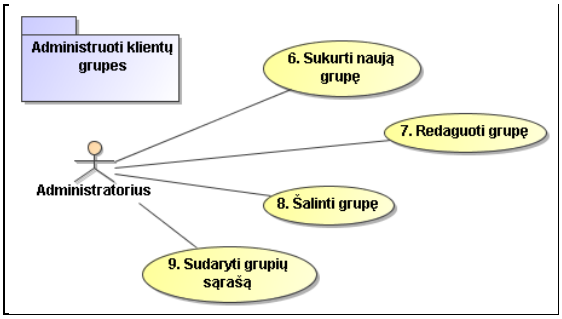
Sistemos sudėtis pateikiama UML diagramomis. Jose atvaizduoti sistemos dalyviai (aktoriai), sistemos komponentai ir atliekami veiksmai.



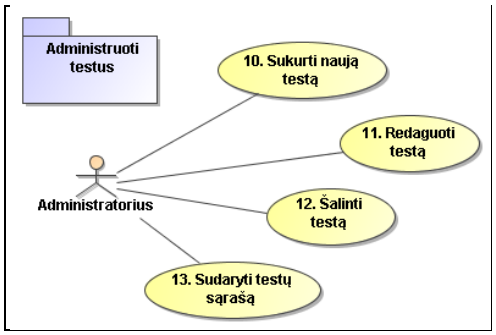
3.2 pav. Administratoriaus funkcijos (1)



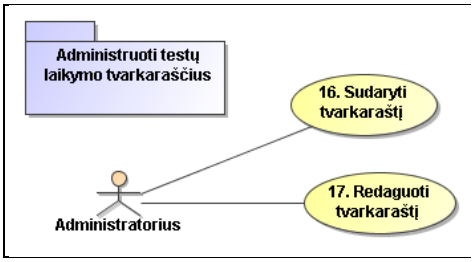
3.3 pav. Administratoriaus funkcijos (2)



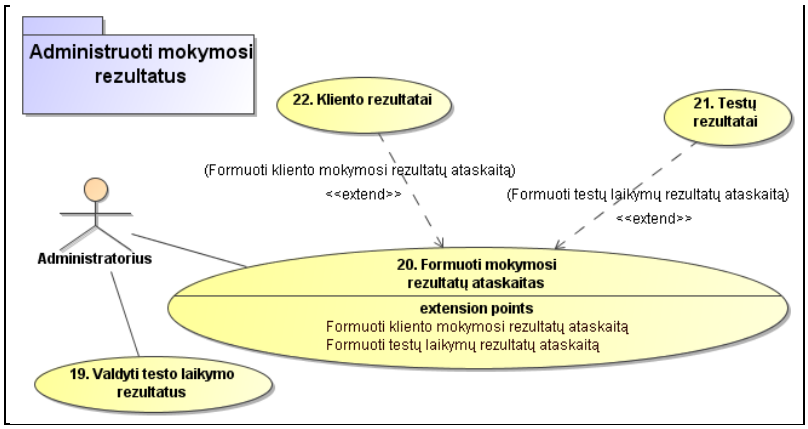
3.4 pav. Administratoriaus funkcijos (3)



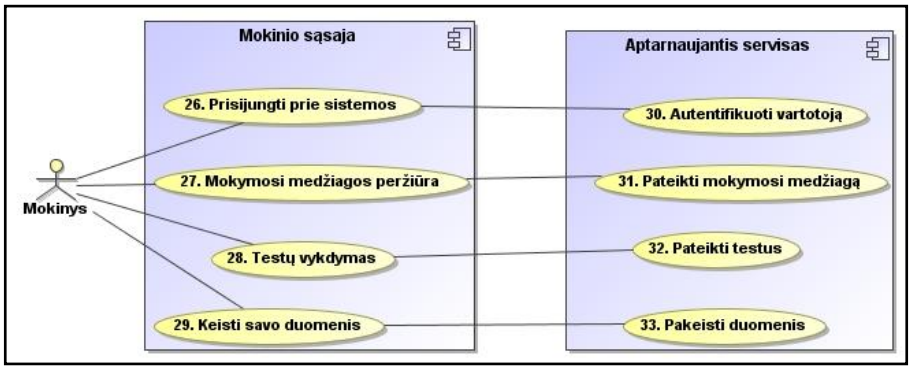
3.5 pav. Administratoriaus funkcijos (4)



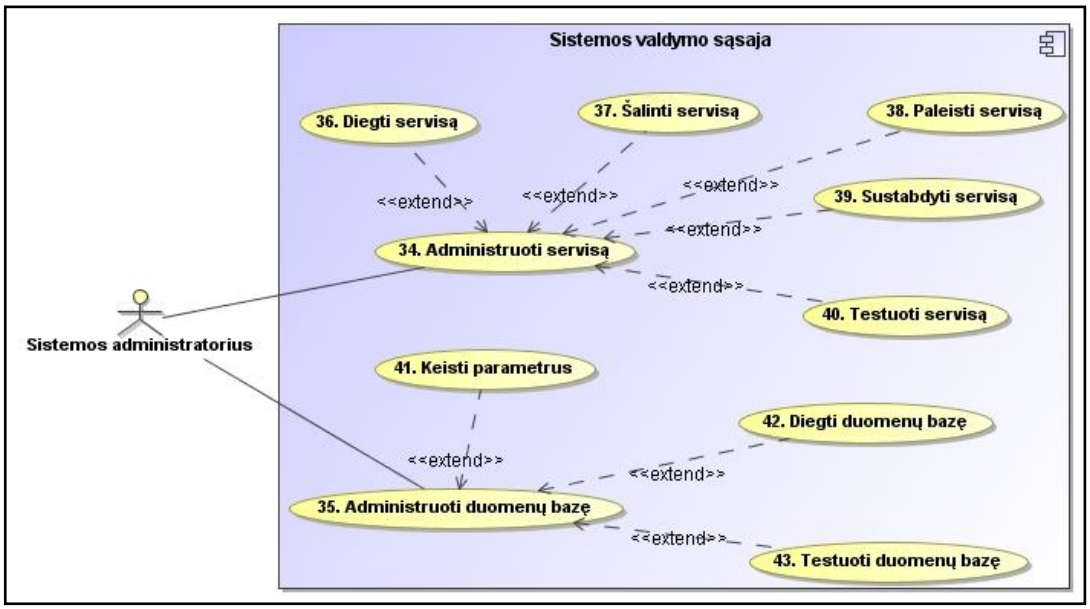
3.6 pav. Administratoriaus funkcijos (5)



3.7 pav. Administratoriaus funkcijos (6)



3.8 pav. Mokinio funkcijos



3.9 pav. Sistemos administratoriaus funkcijos

3.4. Architektūros specifikacija

3.4.1. Architektūros tikslai ir apribojimai

„Nuotolinio mokymosi ir žinių testavimo sistema“ kuriama „nuo nulio“, todėl architektūrinių sprendimų neįtakoja egzistuojančios sistemos versijos, kūrimo įrankiai ar senas (legacy) kodas. Remiantis reikalavimais sistemos priežiūrai, sistema turėtų būti suprojektuota taip, kad pastebėtas klaidas būtų galima kuo greičiau ištaisyti. Taip pat reikalaujama, kad sistema turi būti prieinama daugeliui vartotojų vienu metu, todėl turėtų būti projektuojama remiantis kliento ir serverio architektūra. Be to, kliento ir serverio architektūra leidžia pasiekti tikslą lengvai ir greitai atnaujinti galutinių vartotojų programinę įrangą, nes visa sistemos logika teikiama vieno serverio. Sistemos aprašymui ir realizavimui turi būti naudojami šie pasirinkti įrankiai bei programinės įrangos paketai:

- Microsoft .NET Framework SDK 2.0
- Microsoft Visual Studio 2005 (sistemos realizavimui)
- MySQL Server (duomenų bazių valdymo sistema)
- No Magic MagicDraw UML (sistemos projektavimui ir specifikavimui)
- Microsoft Word (sistemos specifikavimui ir dokumentavimui)

3.4.2. Sistemos statinis vaizdas

Sistema sukurta remiantis kliento ir serverio architektūra. Sistemą sudaro keturi nepriklausomai vienas nuo kito veikiantys komponentai:

- vartotojo sąsajos programa,
- Windows OS servisas, kuriame realizuota sistemos logika,
- duomenų bazė,
- ftp serveris,
- sistemos logikos serviso valdymo ir konfigūravimo programa.

Vartotojo sąsajos programa atsakinga tik už duomenų atvaizdavimą ir funkcijų, reikalingų vartotojo inicijuotiems veiksmams atlikti, iškvietimą iš logikos serviso.

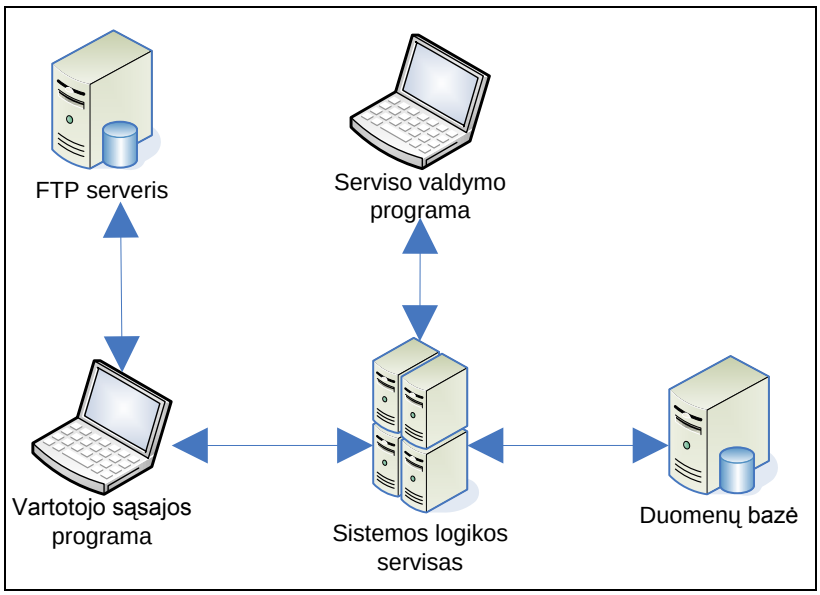
Serviso valdymo programa gali paleisti/sustabdyti sistemos logikos servisą bei nurodyti/keisti prisijungimo prie duomenų bazės ir ftp serverių duomenis.

Duomenų bazės serveryje saugomi sistemos duomenys, veikia duomenų bazių valdymo sistema.

FTP serveryje saugomos daugialypės terpės medžiaga (vaizdo, garso bylos, paveikslėliai).

Sistemos logikos servisas – tai pagrindinė sistemos programa, teikianti sistemai užduotas funkcijas.

Sistemos komponentų bendradarbiavimo schema:



3.10 pav. Sistemos komponentų bendradarbiavimo schema

Sistemos logikos serviso ir vartotojo sąsajos programos atliekamos funkcijos išskaidytos į šiuos modulius:

- klientų ir jų grupių administravimas,
- administratorių administravimas,
- testų ir jų laikymo tvarkaraščių administravimas,
- mokymosi rezultatų administravimas,
- mokomosios medžiagos administravimas,
- darbo su duomenų baze modulis.

4. Eksperimentas: nutolusių programinių objektų sąveikos našumo tyrimas

4.1. Eksperimento tikslas

Tyrimo pagrindinis tikslas – sukurti pavyzdinę programą ir ją realizuoti 4 būdais: naudojantis CORBA technologija, .NET Remote Objects (TCP ir HTTP protokolai), .NET Web Services. Tyrimo metu išmatuoti našumą. Kriterijai, kuriais remiamasi našumo nustatymui: programos vykdymo trukmė (įskaitant susijungimo su tarnybine stotimi trukmę), serveryje naudojama atmintis, tinkle perduodami resursai (matuojama serveryje). Tiriamąją sistemą sudarys vienas serveris ir 4 klientai, kurie palaipsniui iškvies tą patį metodą serveryje. Tyrimo esmė – nustatyti, kuri technologija geriausiai tiktų sukurtam projektui efektyvumo atžvilgiu.

Pastaba: kadangi eksperimentas atliekamas su vienu kompiuteriu, kuriame bus imituojamas klientų ir serverio tinklas, tai su didesniu klientų skaičiumi jis tiesiog nesusidorotų, t.y. neužtektų operatyvinės atminties. O su 4 klientų sistemos rezultatais jau galima daryti atitinkamas išvadas.

4.2. Eksperimento priemonės

Tyrimas bus atliekamas su vienu, tačiau pakankamai galingu kompiuteriu. Kompiuterio parametrai: 4 branduolių procesorius Intel Quad Core Q6600, 4 Gb operatyviosios atminties, Windows XP 64-bit.

Kitos priemonės:

- Virtualizacijos įrankis – VMware Workstation 6
- Virtualioms mašinoms – Windows XP 32-bit
- Programos kūrimo įrankis – Microsoft Visual Studio 2003
- CORBA – IIOPNet 1.9.0 (by ELCA) [http://iiop-net.sourceforge.net/]
- Našumo matavimui – Windows Performance Counter

4.3. Eksperimento eiga

1) Naudojantis Microsoft .NET Framework kompiliatoriumi, sukompiliuojama CORBA biblioteka IIOPNet. Remtasi IIOP.NET dokumentacija [Elc08].
2) Sukuriama sąsaja į serverio metodą (žr. „8.1. Priedas. Sąsaja į serverio metodą.“).
3) Sukuriamas pavyzdinis metodas, kuris atlieka didelius skaičiavimus, t.y. cikle 15000x15000 atliekamos elementarios matematinės funkcijos: sudėtis ir dvi

trigonometrinės funkcijos (žr. „8.2. Priedas. Nuotolinio vykdomojo metodo išeities kodas.“).

4) Sukuriami CORBA, Remote Objects, Web Service serverio ir kliento programos (žr. „8.3. Priedas. Serverio ir kliento programų išeities kodai“). Remtasi literatūra [RS05].
5) Sukuriama viena virtuali mašina. Jai nustatoma kaip vieno procesoriaus branduolio sistema ir paskiriama 512Mb operatyviosios atmintinės. Į ją įdiegiama Windows XP 32-bit operacinė sistema. Remtasi VMware Workstation dokumentacija [Vmw08].
6) Sukurta virtuali mašina klonuojama į atskiras 4 mašinas. Gaunasi 5 identiškos virtualios mašinos. Viena sukonfigūruojama atlikti serverio rolę, kitos 4 – kliento.
7) Serveryje sukonfigūruojamas Windows Performance Counter – išnaudotai atminčiai ir tinklo resursams fiksuoti kas 1 sekundę.
8) Klientų ir serverio virtualiose mašinose įdiegiamos sukurtos programos.
9) Atliekami testai ir fiksuojami rezultatai.

4.4. Eksperimento rezultatai

4.4.1. Skaičiavimo trukmė (įskaitant susijungimo laiką)

Skaičiavimo trukmė skaičiuota kliento pusėje. Imtas laiko tarpas nuo kreipimosi į nutolusį objektą iki rezultato gavimo.

4.1 lentelė. CORBA skaičiavimo trukmė

Klientų skaičius	Vidutinis skaičiavimo laikas (s)
1	32,53125
2	47,546875
3	62,99479167
4	78,34765625

4.2 lentelė. Remote Objects (TCP) skaičiavimo trukmė

Klientų skaičius	Vidutinis skaičiavimo laikas (s)
1	32,515625
2	47,5390625
3	62,41145833
4	78,71875

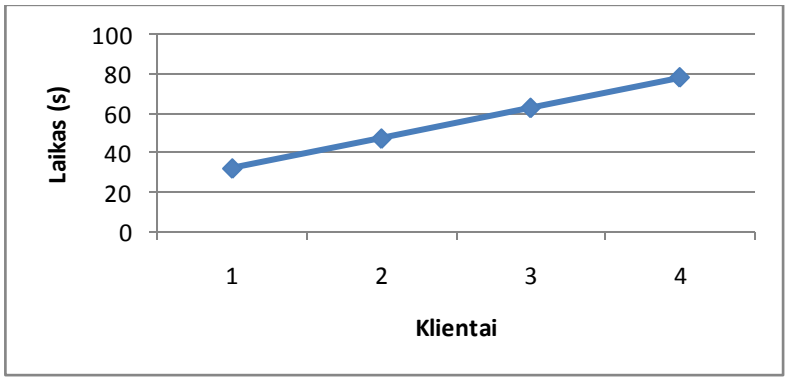
4.3 lentelė. Remote Objects (HTTP) skaičiavimo trukmė

Klientų skaičius	Vidutinis skaičiavimo laikas (s)
1	33,0625
2	47,75
3	62,35416667
4	78,4296875

4.4 lentelė. Web Services skaičiavimo trukmė

Klientų skaičius	Vidutinis skaičiavimo laikas (s)
1	32,578125
2	46,921875
3	62,50520833
4	78,65625

Kaip matyti šių technologijų trukmės beveik vienodos. Todėl galima nubrėžti vieną kreivę (pavyzdys paimtas pagal CORBA duomenis):

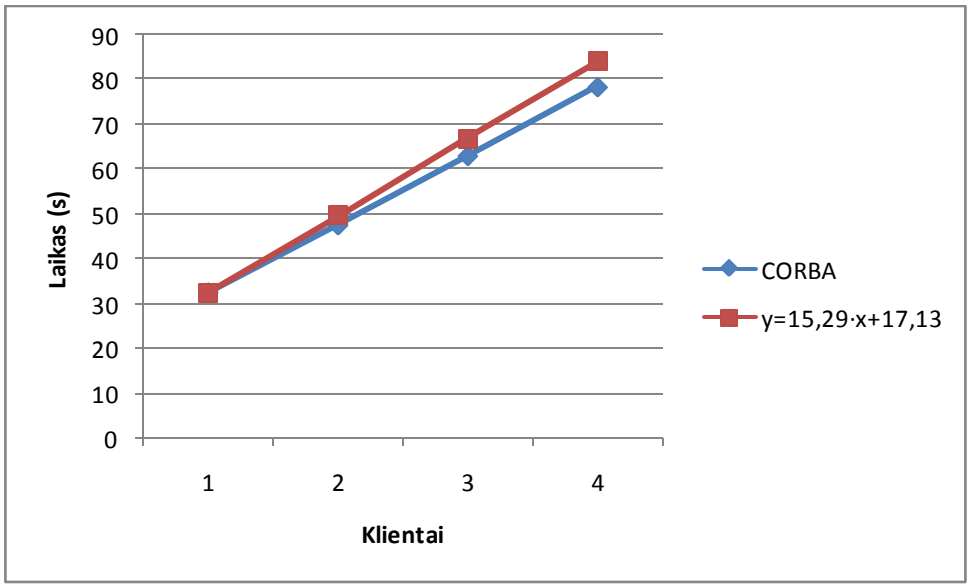


4.1 pav. Skaičiavimo trukmė

Iš diagramos matyti, kad didėjant klientų skaičiui, metodo atlikimo trukmė didėja tiesiškai. Tai galima įrodyti pasinaudojus koreliacinės regresijos analizės teorija (x – klientų skaičius, y – skaičiavimo laikas) [Aks02]:

- Taikome tiesinės regresinės analizės metodą. Apskaičiuojame $SS_{xx} = \sum x^2 - \frac{(\sum x)^2}{4} = 5$, $SS_{yy} = \sum y^2 - \frac{(\sum y)^2}{4} = 1168{,}919$ ir $SS_{xy} = \sum xy - \frac{\sum x \sum y}{4} = 76{,}45$.
- Paskaičiuojame koreliacijos koeficientą: $\rho = \frac{SS_{xy}}{\sqrt{SS_{xx} \cdot SS_{yy}}} = 0{,}99998189$.
- Kadangi koreliacijos koeficientas ρ labai artimas vienetui, tai galima tvirtinti, kad klientų ir skaičiavimo trukmės priklausomybė yra tiesinė [JJ06].

- Apskaičiuojame tiesinės lygties $y = ax + b$ koeficientus: $a = \frac{ss_{xy}}{ss_{xx}} = 15{,}29$, $b = \overline{y} - a\overline{x} = 17{,}13$. Taigi gauname tiesinę lygtį $y = 15{,}29 \cdot x + 17{,}13$. Tai yra skaičiavimo laiko priklausomybė nuo klientų skaičiaus.
- Ši tiesinė priklausomybė skirtųsi esant skirtingai procesoriaus spartai ir procesoriaus bei operatyviosios atmintinės taktiniam dažniui (šiame eksperimente naudojami elementarūs matematiniai skaičiavimai, reikalaujantys procesoriaus ir operatyviosios atmintinės resursų). Vadinasi skirtingose sistemose tiesinės lygties $y = ax + b$ koeficientai a ir b skirsis (kur x – klientų skaičius, y – skaičiavimo laikas).



4.2 pav. Laiko tiesinės priklausomybės kreivė

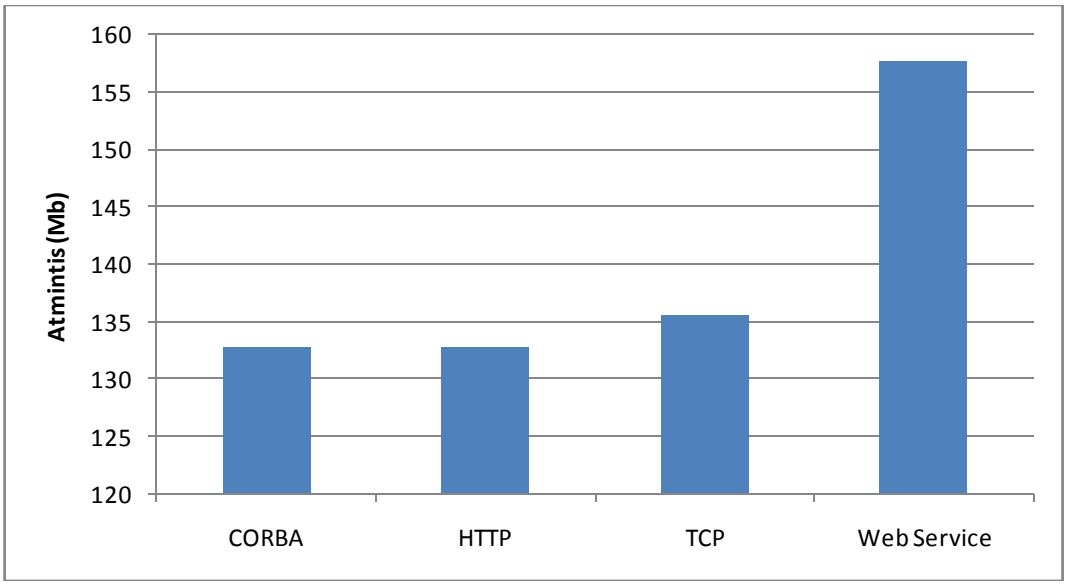
4.4.2. Serverio resursų išnaudojimas

Pateiktoje lentelėje atvaizduojami kiekvienos technologijos vieno kliento operatyviosios atminties sunaudojimas ir vidutinis visų klientų tinkle perduodamų duomenų vidurkis:

4.5 lentelė. Serverio resursų matavimai

Technologija	Sunaudota atmintis (Megabaitai)	Perduota (Baitai)
CORBA	132,8789063	930,9431625
HTTP	132,8789063	930,9431625
TCP	135,7275391	1201,323999
Web Service	157,7636719	2092,731304

Lentelėje matyti, kad CORBA ir Remote Objects HTTP atvejai nesisikiria. Pagal sunaudotos atminties kiekį mažai skiriasi Remote Objects TCP, o pagal bendra išsiųstų tinkle duomenų kiekį vyrauja CORBA ir Remote Objects HTTP.

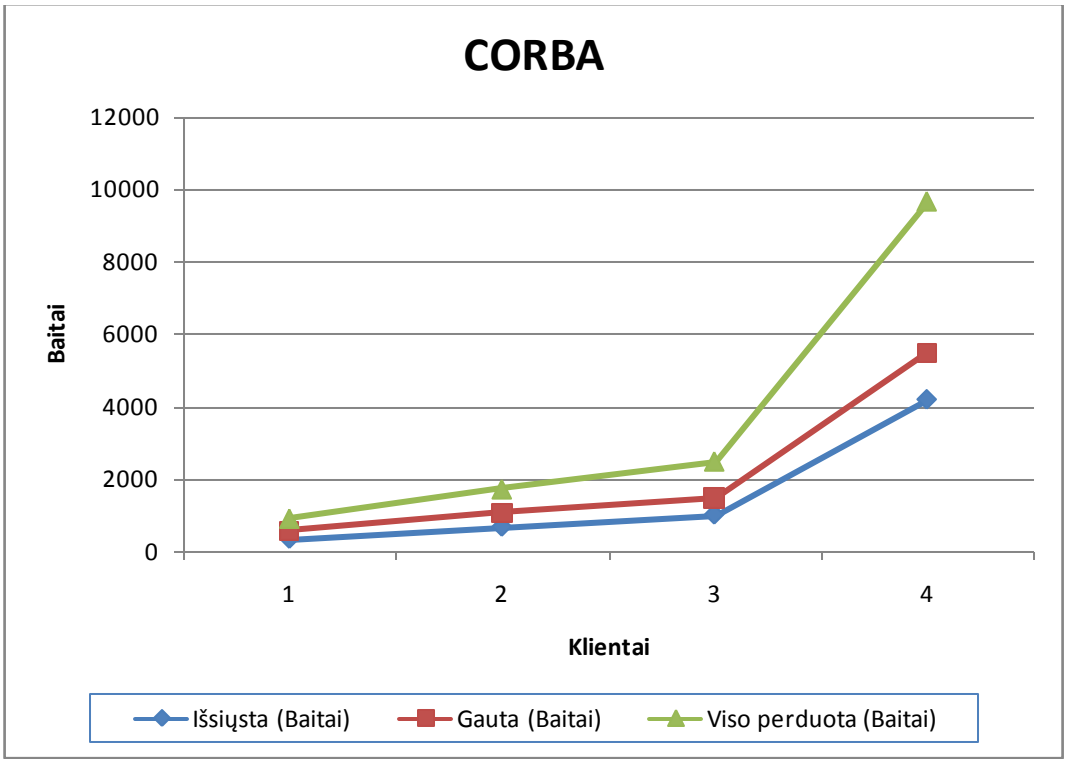


4.3 pav. Atminties sunaudojimas

Atminties sunaudojimo diagramoje matyti, kad CORBA ir Remote Objects HTTP rezultatai nesiskiria. Remote Objects TCP nedaug atsilieka nuo pastarųjų technologijų, o Web Services gerokai neefektyviai išnaudoja atmintį palyginus su kitomis tiriamomis technologijomis.

4.6 lentelė. CORBA tinklo resursai

Klientų skaičius	Išsiųsta (Baitai)	Gauta (Baitai)	Viso perduota (Baitai)
1	334,8754422	596,0677203	930,9431625
2	669,6646977	1071,631204	1741,295902
3	1004,540148	1505,352814	2509,892963
4	4181,116967	5503,713724	9684,830691

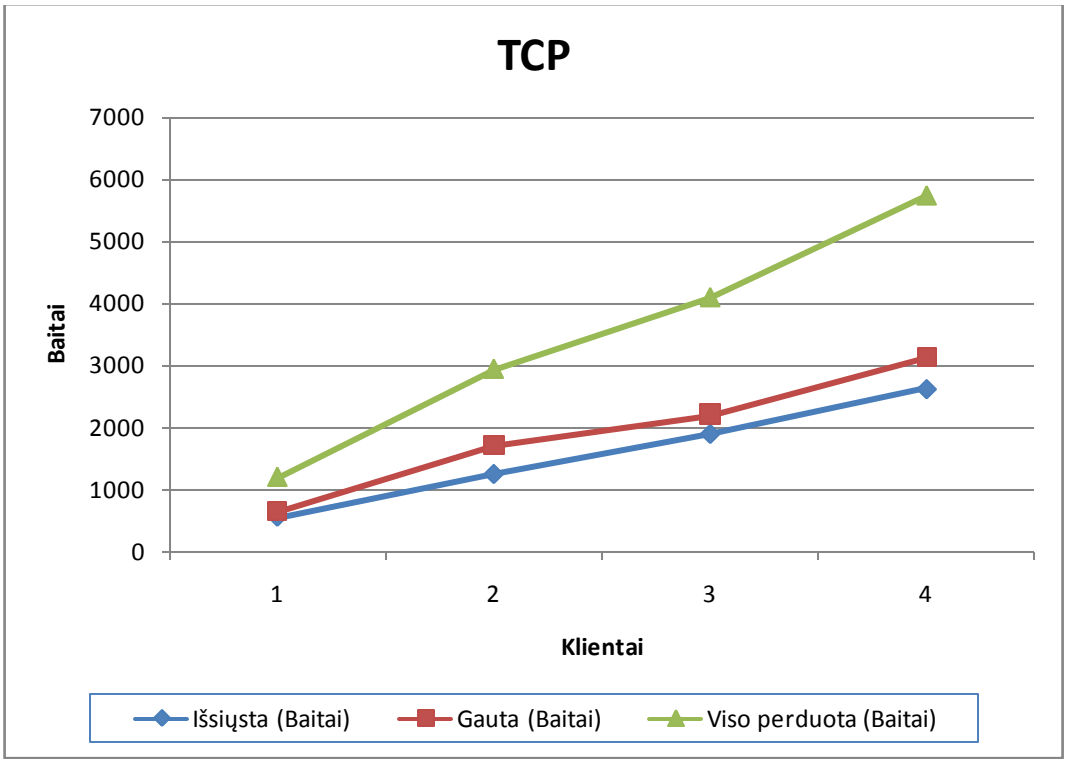


4.4 pav. CORBA tinklo resursai

Paskaičiuojame CORBA klientų skaičiaus ir visų perduotų duomenų tinkle sąryšio koreliaciją: $\rho = 0{,}865770967$. Kadangi $0{,}5 < \rho < 1$, galima sakyti, kad reikšmės išsidėsčiusios gana arti tiesės. Tačiau diagramoje matyti, kad perduotų duomenų kiekis tinkle linkęs augti eksponentiškai nuo klientų skaičiaus.

4.7 lentelė. Remote Objects TCP tinkle resursai

Klientų skaičius	Išsiųsta (Baitai)	Gauta (Baitai)	Viso perduota (Baitai)
1	555,8672921	645,4567074	1201,323999
2	1248,698285	1695,380677	2944,078962
3	1899,355583	2198,767395	4098,122978
4	2624,935837	3114,65608	5739,591916

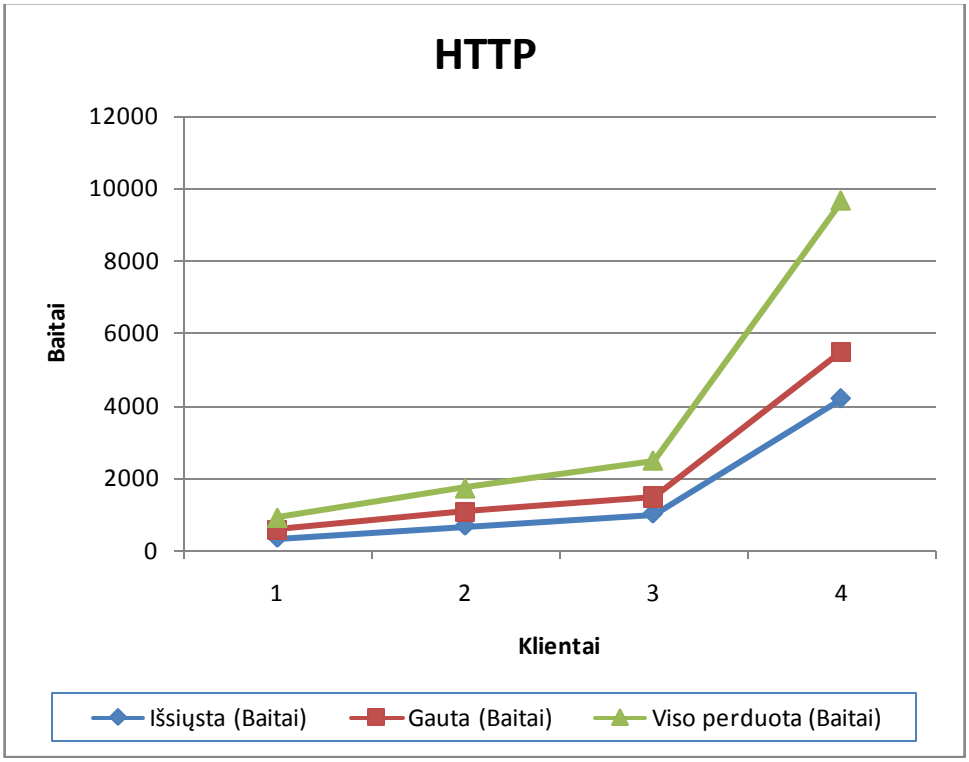


4.5 pav. Remote Objects TCP tinkle resursai

Paskaičiuojame Remote Objects TCP klientų skaičiaus ir visų perduotų duomenų tinkle sąryšio koreliaciją: $\rho = 0{,}997239219$. Kadangi ρ reikšmė labai artima 1, galima sakyti, kad reikšmės išsidėsčiusios labai arti tiesės. Todėl pagal koreliacijos koeficientą ir diagramą galima spręsti, kad perduotų duomenų kiekis tinkle tiesiškai priklauso nuo klientų skaičiaus.

4.8 lentelė. Remote Objects HTTP tinkle resursai

Klientų skaičius	Išsiųsta (Baitai)	Gauta (Baitai)	Viso perduota (Baitai)
1	334,8754422	596,0677203	930,9431625
2	669,6646977	1071,631204	1741,295902
3	1004,540148	1505,352814	2509,892963
4	4181,116967	5503,713724	9684,830691

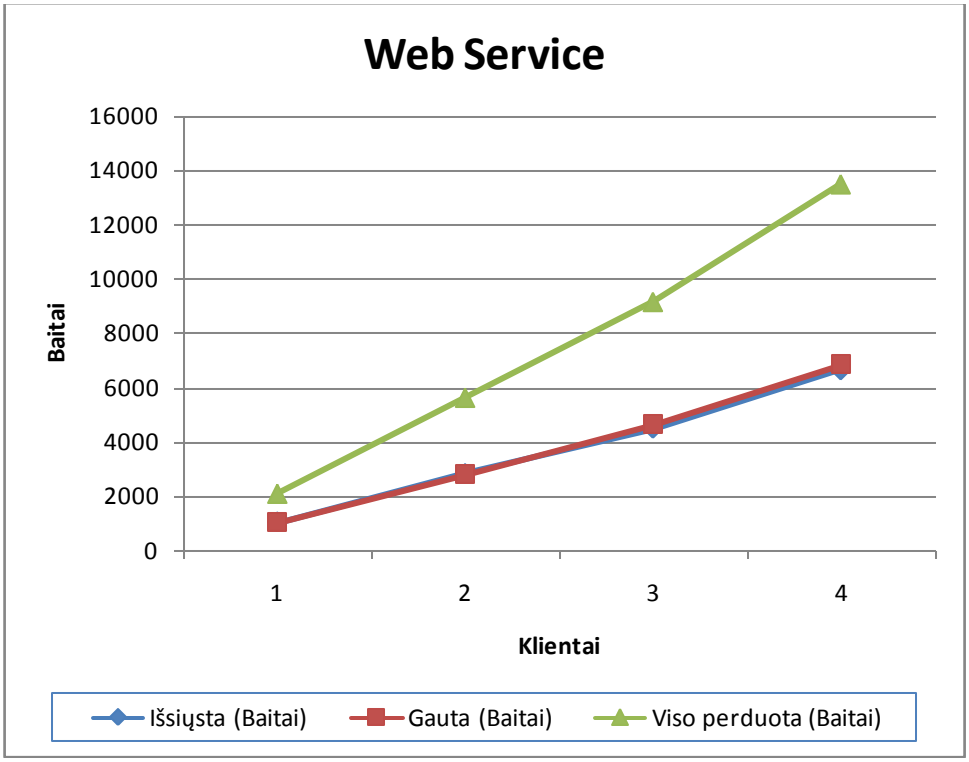


4.6 pav. Remote Objects HTTP tinkle resursai

Paskaičiuojame Remote Objects HTTP klientų skaičiaus ir visų perduotų duomenų tinkle sąryšio koreliaciją: $\rho = 0{,}865770967$. Kadangi $0{,}5 < \rho < 1$, galima sakyti, kad reikšmės išsidėsčiusios gana arti tiesės. Tačiau diagramoje matyti, kad perduotų duomenų kiekis tinkle linkęs augti eksponentiškai nuo klientų skaičiaus.

4.9 lentelė. Web Services tinkle resursai

Klientų skaičius	Išsiųsta (Baitai)	Gauta (Baitai)	Viso perduota (Baitai)
1	1051,225613	1041,505691	2092,731304
2	2845,560683	2789,490521	5635,051204
3	4507,363175	4663,361696	9170,72487
4	6670,682537	6842,642313	13513,32485



4.7 pav. Web Service tinkle resursai

Paskaičiuojame Web Services klientų skaičiaus ir visų perduotų duomenų tinkle sąryšio koreliaciją: $\rho = 0{,}99865$. Kadangi ρ reikšmė labai artima 1, galima sakyti, kad reikšmės išsidėsčiusios labai arti tiesės. Todėl pagal koreliacijos koeficientą ir diagramą galima spręsti, kad perduotų duomenų kiekis tinkle tiesiškai priklauso nuo klientų skaičiaus.

5. Išvados

- Tyrimo rezultatai parodė, kad mažiausiai atminties sunaudoja CORBA ir Remote Objects HTTP technologijos. Ne daug atsilieka (~2%) Remote Objects TCP. Daugiausiai atminties naudoja Web Services.
- Atsižvelgiant į tinklo išnaudojamus resursus, mažiausiai naudoja CORBA ir Remote Objects, tačiau jų išnaudojimas linkęs augti eksponentiškai didėjant klientų skaičiui. Remote Objects TCP tinklo resursų išnaudojimas didėja tiesiškai didėjant klientų skaičiui. Vadinasi esant dideliam klientų skaičiui Remote Objects TCP efektyvesnė nei CORBA ir Remote Objects HTTP technologijos. Web Services palyginus su pastarosiomis technologijomis išnaudoja daugiausiai resursų.
- Tyrimo metu nustatyta, kad „Cool Test Tool“ pasirinkta teisinga technologija, t.y. Remote Objects su TCP protokolu.

6. Literatūra

[Aks02] **A. Aksomaitis**. Tikimybių teorija ir statistika, *Kaunas Technologija*, 2002

[BHM+03] **D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard**. Web Services Architecture, W3C Working Draft 8, August 2003 http://www.w3.org/TR/2003/WD-ws-arch-20030808/

[Elc08] **ELCA.** IIOP.NET documentation, 2008 http://iiop-net.sourceforge.net/documentation.html

[JJ06] **R. Januškevičius, O. Januškevičienė**. Elementarusis tikimybių ir statistikos kursas informatikams. 1 dalis. TIKIMYBIŲ TEORIJOS ELEMENTAI, 2006 http://www.mif.vpu.lt/publikacijos/1.tikimybes.pdf

[KJ04] **M. Kircher and P. Jain.** Pattern-Oriented Software Architecture – Patterns for Resource Management, *John Wiley & Sons*, 2004

[KS08] **Krakowiak, Sacha.** What's middleware?, 2008 http://middleware.objectweb.org/

[Mic08] **Microsoft.** Information on Microsoft COM+ technologies, 2008 http://www.microsoft.com/com/tech/COMPlus.asp

[Nic05] **G. Nick**. Origin of the term middleware. 2005 http://ironick.typepad.com/ironick/2005/07/update_on_the_o.html

[RS05] **I. Rammer, M. Szpuszta.** Advanced .NET Remoting, Second Edition. *Apress*, 2005

[SSRB00] **D. C. Schmidt, M. Stal, H. Rohnert, F. Buschmann**. Pattern-Oriented Software Architecture – Patterns for Concurrent and Networked Objects, *John Wiley & Sons*, 2000

[Ste98] **R. Stevens.** UNIX Network Programming, *Prentice Hall*, 1998

[TS02] **A. Tanenbaum, M. van Steen**. Distributed Systems: Principles and Paradigms. *Prentice Hall*, 2002

[VKZ05] **M. Voelter, M. Kircher, U. Zdun**. Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware. *John Wiley & Sons*, 2005

[Vmw08] **VMware**. Guest Operating System Installation Guide, 2008 http://www.vmware.com/pdf/GuestOS_guide.pdf

[VSW02] **M. Völter, A. Schmid, E. Wolff.** Server Component Patterns. *John Wiley & Sons*, 2002

7. Santrumpų ir terminų žodynas

Application Server – programinė įranga, kurios dėka klientas gali naudoti kitoje sistemoje esančius programinius objektus, neturėdamas jų savo kompiuteryje.

CORBA – Common Object Request Broker Architecture.

Marshaller – programinė dalis, kuri įgalina objektus paversti į bitų seką ir transportuoti tinklu.

OMG – Object Management Group.

Tarpinė programinė įranga (Middleware) – programinė įranga, kuri suriša tarpusavyje atskiras programas, nepriklausomai nuo to, kokioje aplinkoj jos veikia.

Web Server – programa, kuri priima HTTP užklausas ir pateikia atitinkamus rezultatus, kuriuos klientas mato naršyklėje.

8. Priedai

8.1. Priedas. Sąsaja į serverio metodą.

```
public interface IRemoteObject
{
    double RemoteMethod(double x);
}
```

8.2. Priedas. Nuotolinio vykdomojo metodo išeities kodas.

```
public class RemoteObject : MarshalByRefObject, IRemoteObject
{
    public double RemoteMethod(double x)
    {
            int nCyc = RemoteInterface.Utilities.Cycles;
            double result = 0;
            for(int i = 0; i < nCyc; i++)
            {
                    for(int j = 0; j < nCyc; j++)
                    {
                            result = Math.Cos(x) + i + Math.Sin(j);
                    }
            }
            return result;
    }
}
```

8.3. Priedas. Serverio ir kliento programų išeities kodai.

ServerCORBA.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using Ch.Elca.Iiop;
using RemoteLibrary;

namespace ServerCORBA
{
    class Server
    {
        [STAThread]
        static void Main(string[] args)
        {
            int port = RemoteInterface.Utilities.Port;
            IiopChannel chan = new IiopChannel(port);
            ChannelServices.RegisterChannel(chan);

            RemoteObject obj = new RemoteObject();
            string objectURI = RemoteInterface.Utilities.ObjectURI;
            RemotingServices.Marshal(obj, objectURI);

            Console.WriteLine("server running on :" + port.ToString() + "/" + objectURI);
            Console.ReadLine();
        }
    }
}
```

ClientCorba.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using Ch.Elca.Iiop;
using Ch.Elca.Iiop.Services;
using RemoteInterface;

namespace ClientCORBA
{
    class Client
    {
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                string host = RemoteInterface.Utilities.Address;
                int port = RemoteInterface.Utilities.Port;
                string objURI = RemoteInterface.Utilities.ObjectURI;

                Console.WriteLine("Server on: iiop://" + host + ":" + port + "/" + objURI);
                Console.WriteLine("input number: ");
                double x = Double.Parse(Console.ReadLine());

                IiopClientChannel channel = new IiopClientChannel();
                ChannelServices.RegisterChannel(channel);

                IRemoteObject obj =
(IRemoteObject)RemotingServices.Connect(typeof(IRemoteObject), "iiop://" + host + ":" + port +
"/" + objURI);

                long dtStart = DateTime.Now.Ticks;
                double result = obj.RemoteMethod(x);
                long dtStop = DateTime.Now.Ticks;
                TimeSpan duration = new TimeSpan(dtStop - dtStart);
                Console.WriteLine("result: " + result);
                Console.WriteLine("Duration: " + duration.TotalSeconds.ToString() + " s");
            }
            catch (Exception e)
            {
                Console.WriteLine("exception: " + e);
            }
            finally
            {
                Console.ReadLine();
            }
        }
    }
}
```

ServerTCP.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using RemoteLibrary;

namespace ServerTCP
{
    class Server
    {
        [STAThread]
        static void Main(string[] args)
        {
            int port = RemoteInterface.Utilities.Port;
            TcpServerChannel chan = new TcpServerChannel(port);
            ChannelServices.RegisterChannel(chan);

            RemoteObject obj = new RemoteObject();
            string objectURI = RemoteInterface.Utilities.ObjectURI;
            RemotingServices.Marshal(obj, objectURI);

            Console.WriteLine("server running on :" + port.ToString() + "/" + objectURI);
            Console.ReadLine();
        }
    }
}
```

ClientTCP.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using RemoteInterface;

namespace ClientTCP
{
    class Client
    {
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                string host = RemoteInterface.Utilities.Address;
                int port = RemoteInterface.Utilities.Port;
                string objURI = RemoteInterface.Utilities.ObjectURI;

                Console.WriteLine("Server on: tcp://" + host + ":" + port + "/" + objURI);
                Console.WriteLine("input number: ");
                double x = Double.Parse(Console.ReadLine());

                TcpClientChannel channel = new TcpClientChannel();
                ChannelServices.RegisterChannel(channel);

                IRemoteObject obj =
(IRemoteObject)RemotingServices.Connect(typeof(IRemoteObject), "tcp://" + host + ":" + port + "/"
+ objURI);

                long dtStart = DateTime.Now.Ticks;
                double result = obj.RemoteMethod(x);
                long dtStop = DateTime.Now.Ticks;
                TimeSpan duration = new TimeSpan(dtStop - dtStart);
                Console.WriteLine("result: " + result);
                Console.WriteLine("Duration: " + duration.TotalSeconds.ToString() + " s");
            }
            catch (Exception e)
            {
                Console.WriteLine("exception: " + e);
            }
            finally
            {
                Console.ReadLine();
```

```
            }
        }
    }
}
```

ServerHTTP.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
using RemoteLibrary;

namespace ServerHTTP
{
    class Server
    {
        [STAThread]
        static void Main(string[] args)
        {
            int port = RemoteInterface.Utilities.Port;
            HttpServerChannel chan = new HttpServerChannel(port);
            ChannelServices.RegisterChannel(chan);

            RemoteObject obj = new RemoteObject();
            string objectURI = RemoteInterface.Utilities.ObjectURI;
            RemotingServices.Marshal(obj, objectURI);

            Console.WriteLine("server running on :" + port.ToString() + "/" + objectURI);
            Console.ReadLine();
        }
    }
}
```

ClientHTTP.cs

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Http;
using RemoteInterface;

namespace ClientHTTP
{
    class Client
    {
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                string host = RemoteInterface.Utilities.Address;
                int port = RemoteInterface.Utilities.Port;
                string objURI = RemoteInterface.Utilities.ObjectURI;

                Console.WriteLine("Server on: http://" + host + ":" + port + "/" + objURI);
                Console.WriteLine("input number: ");
                double x = Double.Parse(Console.ReadLine());

                HttpClientChannel channel = new HttpClientChannel();
                ChannelServices.RegisterChannel(channel);

                IRemoteObject obj =
(IRemoteObject)RemotingServices.Connect(typeof(IRemoteObject), "http://" + host + ":" + port +
"/" + objURI);

                long dtStart = DateTime.Now.Ticks;
                double result = obj.RemoteMethod(x);
                long dtStop = DateTime.Now.Ticks;
                TimeSpan duration = new TimeSpan(dtStop - dtStart);
                Console.WriteLine("result: " + result);
                Console.WriteLine("Duration: " + duration.TotalSeconds.ToString() + " s");
            }
            catch (Exception e)
            {
                Console.WriteLine("exception: " + e);
            }
            finally
            {
                Console.ReadLine();
            }
        }
    }
}
```

ServerWebService.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using RemoteLibrary;

namespace ServerWEBService
{
    [WebService(Namespace = "http://192.168.0.1")]
    public class Server : System.Web.Services.WebService
    {
        public Server()
        {
            InitializeComponent();
        }


        [WebMethod]
        public double WebMethod(double x)
        {
            RemoteObject ro = new RemoteObject();
            double result = ro.RemoteMethod(x);
            return result;
        }
    }
}
```

ClientWebService.cs

```
using System;

namespace ClientWEBService
{
    class Client
    {
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("input number: ");
                double x = Double.Parse(Console.ReadLine());
                WebReference.Server srv = new ClientWEBService.WebReference.Server();
                srv.Timeout = System.Threading.Timeout.Infinite;

                long dtStart = DateTime.Now.Ticks;
                double result = srv.WebMethod(x);
                long dtStop = DateTime.Now.Ticks;
                TimeSpan duration = new TimeSpan(dtStop - dtStart);
                Console.WriteLine("result: " + result);
                Console.WriteLine("Duration: " + duration.TotalSeconds.ToString() + " s");
            }
            catch (Exception e)
            {
                Console.WriteLine("exception: " + e);
            }
            finally
            {
                Console.ReadLine();
            }
        }
    }
}
```