KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Kęstutis Malinauskas

RAIŠKIŲJŲ INTERNETO PROGRAMŲ TECHNOLOGIJOS IR JŲ PANAUDOJIMAS KURIANT INTERAKTYVIAS PASKIRSTYTAS SISTEMAS

Magistro baigiamasis darbas

Darbo vadovas: dr. Tomas Blažauskas

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Kęstutis Malinauskas

RAIŠKIŲJŲ INTERNETO PROGRAMŲ TECHNOLOGIJOS IR JŲ PANAUDOJIMAS KURIANT INTERAKTYVIAS PASKIRSTYTAS SISTEMAS

Magistro baigiamasis darbas

Recenzentė

dr. Lina Čeponienė

2009-05-25

Vadovas

dr. Tomas Blažauskas

2009-05-25

Atliko

IFM-3/2 gr. stud.

Kęstutis Malinauskas

2009-05-25

Kaunas, 2009

Turinys

Lentelių sąrašas

Paveikslų sąrašas

1. Įvadas

Raiškiosios interneto programos (iš anglų kalbos - Rich Internet Application, sutrumpintai ir toliau – **RIA**) - tai terminas apibrėžiantis gana naują interneto technologijų klasę, leidžiančių apjungti interneto ir darbalaukio stipriausias savybes ir sukurti išskirtinę vartotojo sąsają bei funkcionalumą. Angliškas šio termino variantas buvo sukurtas ir išvystytas paskutiniajame dvidešimto amžiaus dešimtmetyje svarbios interneto istorijai įmonės Macromedia Inc. Nors praėjo jau 10 metų, tačiau terminas nėra pilnai ir visapusiškai priimtas. Viena iš priežasčių yra ta, kad šis terminas apima labai platų technologijų spektrą – Flex, AIR, Silverlight, JavaFX, AJAX bei kitas technologijas.

RIA technologijos įgyja vis didesnę reikšmę programų inžinerijos srityje. Vis daugiau programų bei sistemų kuriama naudojant šias technologijas. Todėl įvairūs programų inžinerijos mokslo aspektai turi būti pritaikyti tam, kad būtų palengvintas tokios programinės įrangos kūrimo procesas ir tuo pačiu pagerintas jo produktas – RIA programos bei jų sistemos. Darbe analizuojami architektūros projektavimo, našumo bei saugumo aspektai.

Didelis dėmesys darbe atkreipiamas į RIA programų architektūrą. Analizuojami skirtumai tarp RIA bei interneto ar darbastalio programų. Aprašomos svarbiausios RIA architektūros charakteristikos. Analizuojant bei tiriant bendras RIA charakteristikas remiamasi konkrečia technologija – Flex.

Magistro studijų metu buvo įvykdytas magistro projektinis darbas – Incidentų Valdymo Sistema. Tai ypatingų situacijų valdymo programa, skirta ekstremalių situacijų valdymo tarnybų darbo organizavimui. Aprašomi projekto metu priimti technologijos taikymo sprendimai ir pagrindžiamas technologijos pasirinkimas.

Technologijos pasirinkimui pagrįsti buvo atlikti greitaveikos, apkrovimo bei modulinės architektūros eksperimentai.

Galiausiai darbe pateikiamos technologijos analizės bei taikymo rezultatai ir išvados.

2. RIA analizė

2.1. RIA apžvalga

Šiame skyriuje aprašoma trumpa RIA technologijų istorija, termino apibrėžimas ir bendri reikalavimai RIA programoms.

2.1.1. Trumpa RIA Istorija

Atsiradus ir paplitus internetui bei HTML standartui vis daugiau informacijos ir įmonių veiklos buvo perkeliama į internetą. Labai greitai vartotojams ir kūrėjams ėmė nebepakakti standartinių HTML standarto galimybių. Trūko animacijos, interaktyvumo, dinamiškumo – viso to kas buvo įprasta standartinėse darbastalio programose. Tuomet, 1996 metais, lyderiaujanti programinės įrangos kompanija Microsoft išleido savo naršyklės Internet Explorer praplėtimą, kuris leido vykdyti specialias programas – ActiveX įskiepius. Įskiepiai – tai programos, kurios yra paleidžiamos iš naršyklės, tačiau veikia tiesiogiai vartotojo kompiuteryje arba virtualias mašinų viduje, todėl yra nepriklausomi nuo naršyklės ir interneto standartų. Ši įskiepių sistema paplito, nes leido tinklalapyje vykdyti programas, turinčias vartotojo sąsają panašią į naudojamą darbastalio programose, ar atlikti veiksmus, kurie reikalauja tiesioginio priėjimo prie vartotojo kompiuterio.

Ilgainiui dėl gero grafikos palaikymo išpopuliarėjo viena įskiepių rūšis – Flash, kuri veikė Flash grotuvo virtualioje mašinoje. Šio įskiepio kūrėjas – kompanija Macromedia – išvystė ir įtvirtino „Rich Internet Application“ terminą. Vienu iš svarbiausių RIA istorijos įvykių galima laikyti 2002 metų kovą išleistą raiškiųjų interneto programų reikalavimų dokumentą[1].

Maždaug tuo pačiu metu tinklalapių kūrėjai pradėjo vis dažniau naudoti iki tol mažai žinomą Internet Explorer komponentą – XMLHttpRequest. Jis leido parsiųsti duomenis naudojant JavaScript kalbą ir neperkraunant esamo puslapio. Ši nauja ir galinga galimybė atvėrė duris AJAX technologijų raidai. XMLHttpRequest ir JavaScript naudojimas leido atskirti vaizdavimo dalį ir duomenis. Dėl šių ypatumų AJAX taip pat laikomas RIA technologija.

Flash įskiepio ir AJAX technologijų sėkmė išpopuliarino RIA programas ir pritraukė labai daug kitų technologijas kuriančių kompanijų. Dėl tos priežasties buvo sukurta nemažai alternatyvių aplinkų kurti RIA programas [7]. Populiariausios iš jų yra Flex, Silverlight, JavaFX.

Šiuo metu (2009 metais) RIA tapo vienu pagrindinių terminu apibūdinančiu šį platų technologijų spektrą.

2.1.2. Termino apibrėžimas

RIA yra termino „Rich Internet Application“ sutrumpinimas. Lietuviškas šio termino vertimas dar nėra oficialiai patvirtintas ar įsišaknijęs. Šiame magistro darbe siūlomas šio termino vertimas - Raiškioji interneto programa. Parenkant vertimą buvo konsultuotasi su Valstybine lietuvių kalbos komisija.

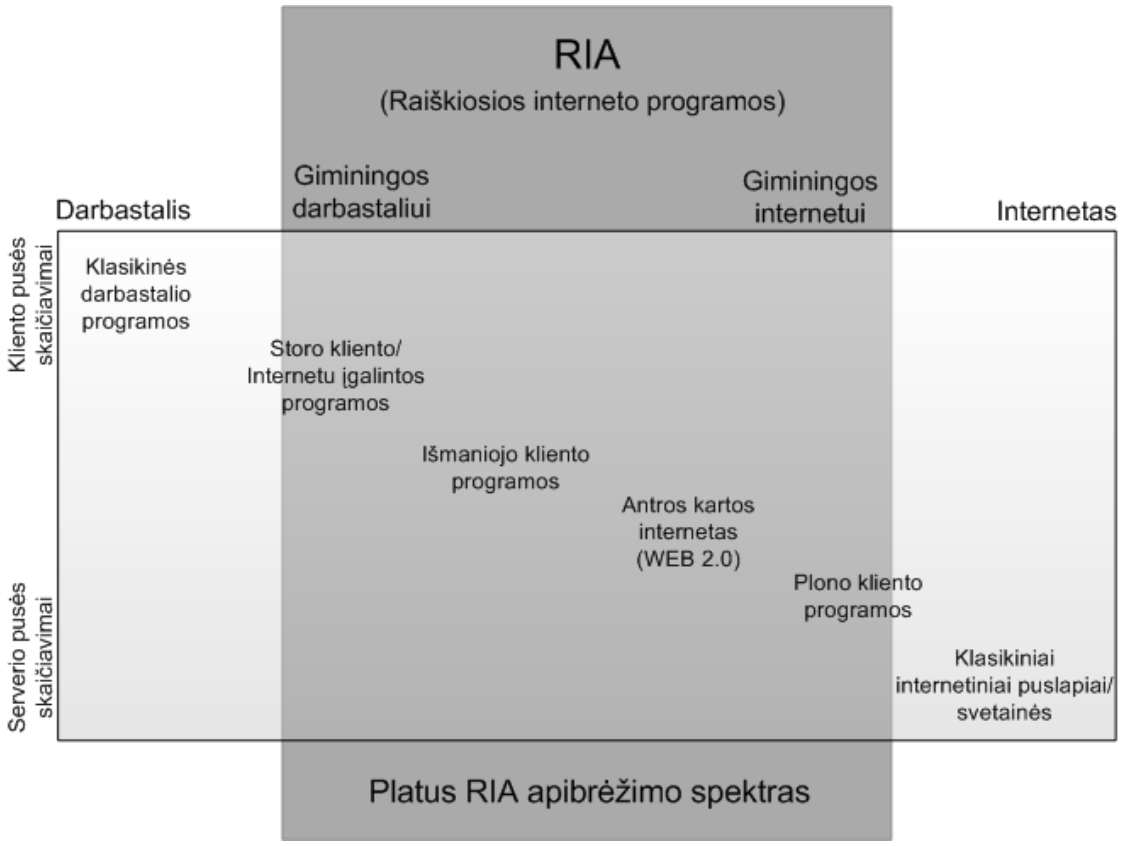
RIA terminas iš esmės apima visas su internetu susijusias programas, kurios nėra grynai darbastalio programos ir nėra gryni tinklalapiai. Apibrėžimas toli gražu nėra aiškus ir paprastai suprantamas.

1 paveiksle pateikiamas įvairių kompiuterinių terminų pasiskirstymas pagal giminystę su darbastaliu ar internetu (horizontali ašis) bei pagal skaičiavimų dalį serveryje ar kliento kompiuteryje (vertikali ašis). Pagal šią schemą į RIA terminą įeina:

- Plono kliento programos;
- Antros kartos internetas (Web 2.0);
- Išmaniojo (angl. - Smart) kliento programos;
- Storo kliento, internetu įgalintos programos.

Iš paveikslo matome, kad technologijos išsidėsčiusios viena ašimi – arba programos yra gimininges darbastaliui (kartu daugiau skaičiavimų atliekama kliento kompiuteryje) arba gimininges internetui (ir daugiau skaičiavimų atliekama serveryje). Gimininges darbastaliui technologijos yra Flex, Silverlight, JavaFX, turinčios savo virtualias mašinas, o gimininges internetui – AJAX, bei kitos naršykle paremtos programos[2].

Siaurąja prasme RIA technologijos apima būtent gimininges darbastaliui technologijas (Flex, Silverlight, JavaFX ir k.t.), kurios pasižymi tuo, kad programos yra iš anksto sukompiliuojamos ir vykdomos specialios virtualios mašinos aplinkoje. Būtent šios technologijos bus daugiausiai nagrinėjamos šiame darbe. Šiame darbe RIA terminas reikš šią specifinę technologijų klasę.



Pav. 1 RIA technologijų terminų spektras

2.1.3. Reikalavimai RIA aplinkoms

2002 kovą Macromedia kompanija paskelbė tuo metu naujo savo produkto Flash MX vizijos dokumentą [1]. Jame buvo aprašyta naujos interneto programų kartos vizija ir kartu reikalavimai šioms programos. Šiuos reikalavimus galima laikyti etalonu RIA programoms. RIA aplinkos ir technologijos turi:

- Sukurti efektyvią ir našią aplinką vykdomam kodui, turiniui ir komunikacijoms;
- Integruoti turinį, ryšius ir programos sąsają į vieną visumą;
- Įgalinti galingus ir išplečiamus objektų modelius interaktyvumui užtikrinti;
- Leisti greitą programų kūrimą naudojant komponentus ir pakartotinį panaudojimą;
- Leisti naudoti interneto ir duomenų paslaugas teikiamas tarnybinių stočių;
- Spręsti prisijungusių ir neprisijungusių klientų problemą;
- Leisti lengvą įdiegimą skirtingose platformose ir įtaisuose.

2.2. Technologijų apžvalga

Toliau pateikiamos populiariausios RIA technologijos (2009 m.). Technologijų sąrašą pradeda Flex karkasas, kuris buvo pasirinktas vykdyti analizei bei atlikti magistro projektą.

2.2.1. Flex/AIR

Flex technologija buvo išvystyta Macromedia kompanijoje apie 2004 metus sukūrus papildomas bibliotekas Flash įskiepiui. 2005 metų pabaigoje Macromedia kompanija ir visi jos produktai buvo perimti Adobe kompanijos. Nuo to laiko ši technologija pradėta vadinti Adobe Flex. Nuo 2008 metų didžioji dalis šio karkaso bibliotekų buvo išleistos su atviro kodo licencija. Paskutinė karkaso versija – 3.3.

Flex technologijos pagrindas yra Flash platforma, kuri pati savaime gali būti laikoma RIA technologija. Tačiau Flash aplinka yra labiau pritaikyta nedidelių įskiepių, atliekančių specifines funkcijas kūrimui. Taip pat ši platforma turi „laiko juostos“ (angl. - timeline) sąvoką, kuri yra labiau suprantama dizaineriams, o ne programuotojams. Papildomos Flex bibliotekos iš esmės pakeitė programų kūrimą Flash platformoje.

Flex karkaso programavimo kalbos: MXML, ActionScript.

Integruotos programavimo aplinkos: Flex Builder.

Reikalingi įskiepiai: Flash Player 9 ir aukštesnė versija.

2007 metais Adobe kompanija išleido naują virtualią mašiną – AIR. Ji leidžia vykdyti Flex karkaso programas tiesiai iš vartotojo kompiuterio. Ši aplinka veikia visose pagrindinėse operacinėse sistemose (Windows, Mac OS ir Unix). Šiuo žingsniu buvo užtikrinta, kad Flex karkaso pagrindu sukurtos programos be didelių pakeitimų gali veikti visose naršyklėse ir visose operacinėse sistemose.

2.2.2. Silverlight

Silverlight yra 2007 metais Microsoft kompanijos išleistas naršyklės įskiepis. Iš pradžių jis palaikė tik vektorinę grafiką ir video grojimą, o su vėlesnėmis versijomis buvo įvestas ir .NET karkaso programinis palaikymas. Dabar Silverlight palaiko dalį kitos Microsoft technologijos - WPF – funkcionalumo. Bendrumas su WPF leidžia su minimaliomis pastangomis kurti tą pačią programą kaip darbastalio ir interneto programą. Naujausia karkaso versija – 3.0.

Silverlight karkaso programavimo kalbos: XAML, C# ir kitos .NET kalbos.

Integruotos programavimo aplinkos: Microsoft Expression Blend ir Microsoft Visual Studio.

Reikalingi įskiepiai: Silverlight vykdymo aplinka.

Silverlight veikia didžiojoje dalyje naršyklių ir Windows bei Mac OS operacinėse sistemose. Paskutinėje versijoje numatyta galimybė naršyklėje veikiančią Silverlight programą išsikeltį į darbastalį ir leisti kaip darbastalio programą (panašiai kaip Adobe AIR).

2.2.3. JavaFX

Oficialiai paskelbta 2007 metais ši Sun kompanijos technologija be naršyklės dar orientuojasi į mobiliųjų įrenginių rinką. Šiais metais buvo išleista JavaFX 1.1 versija. Nors technologija yra labai nauja, bet ji turi gerą integraciją su Java technologijomis ir stiprų palaikymą iš Java programuotojų pusės.

Programavimo kalbos: JavaFX Script, Java.

Integruotos programavimo aplinkos: JavaFXPad, NetBeans, JFXBuilder.

Reikalingi įskiepiai: Java vykdymo aplinka (JRE) 1.5 ar aukštesnė versija.

JavaFX veikia visose operacinėse sistemose ir nepriklauso nuo naršyklės. Stiprioji šios technologijos pusė yra mobiliųjų įrenginių palaikymas. Pastaruoju metu palaikomos Android ir Windows Mobile operacinės sistemos.

2.2.4. AJAX

AJAX - tai naršyklės technologijų (HTML, XML, CSS, SVG, Canvas) rinkinys, skirtas pagerinti paprastų tinklalapių sklandumui ir funkcionalumui. Jis išsivystė kartu su interneto standartais, o pirmą kartą AJAX terminas paminėtas 2005 metais.

Pagrindinė programavimo kalba AJAX technologijoje yra JavaScript, bet serverio dalies kalba priklauso nuo pasirinktos technologijos. AJAX programoms nereikia specialių naršyklės įskiepių, todėl jos veikia visose naršyklėse ir visose operacinėse sistemose.

AJAX technologija yra labai priklausoma nuo kiekvienos naršyklės HTML ir CSS vaizdavimo variklio. Todėl labai svarbu, kad visos naršyklės laikytųsi bendrų HTML standartų, apibrėžiamų W3C organizacijos. Tačiau dažniausiai naršyklės nesilaiko bendrų standartų, todėl programų kūrimas su AJAX technologija tampa daug sudėtingesnis. Dažnai skirtingoms naršyklėms reikia kurti skirtingas kodo šakas, ar net programos versijas.

Kita AJAX ir HTML problema yra prastas grafikos palaikymas, bei vietinės duomenų saugyklos neturėjimas. Dėl šių standartų trūkumo kaip tik ir buvo sukurti įskiepiai, bei išvystyta RIA technologija. Naujai rengiamame HTML5 standarte numatomas geresnis grafikos palaikymas, bei vietinės saugyklos, tačiau šis standartas yra dar tik vystymo stadijoje ir kai kuriomis prognozėmis gali pasirodyti tik už 10 metų.

2.2.5. Kitos technologijos

Kitos RIA technologijos yra mažiau populiarios, nes jos neturi didžiųjų programinės įrangos technologijų kūrėjų palaikymo. Šių technologijų sąrašas:

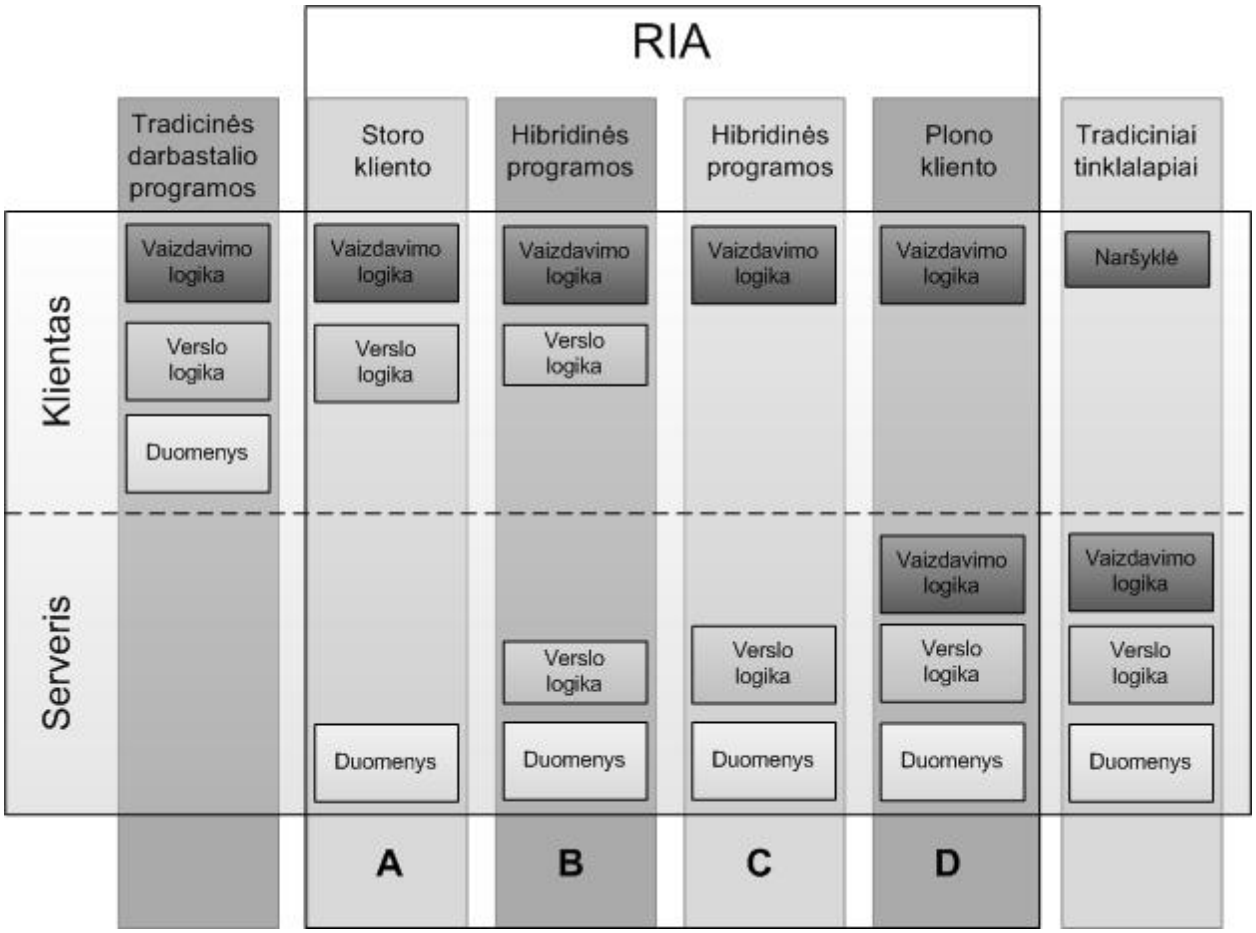
- OpenLaszlo[1];
- uniPaaS[2];
- Curl[3];
- XUL[4].

2.3. RIA architektūros analizė

RIA architektūra yra pagrįsta kliento – serverio architektūros paradigma. Dalis skaičiavimų atliekama serveryje, dalis kliente. Skaidant sistemos architektūrą į tris lygius (vaizdavimo logika, verslo logika, duomenys), galima pamatyti visą šių lygių pasiskirstymo tarp serverio ir kliento spektrą [2]. Šis pasiskirstymas pateiktas 2 paveiksle.

Iš paveikslo matome, kad tradicinėse darbastalio programose visi logikos lygmenys yra kliente, o tradicinių puslapių atveju, visi lygmenys yra serveryje. Visi tarpiniai pasiskirstymo būdai priklauso RIA architektūrai. Iš kitos pusės, RIA programos gali būti įgyvendintos visais keturiais tarpiniais tipais. Logikos pasiskirstymo būdai pažymėti raidėmis A, B, C, D.

[1] http://www.openlaszlo.org/
[2] http://www.magicsoftware.com/2559-en/uniPaaS.aspx
[3] http://www.curl.com/
[4] http://www.mozilla.org/projects/xul/



Pav. 2 Programos lygmenų pasiskirstymas RIA architektūroje

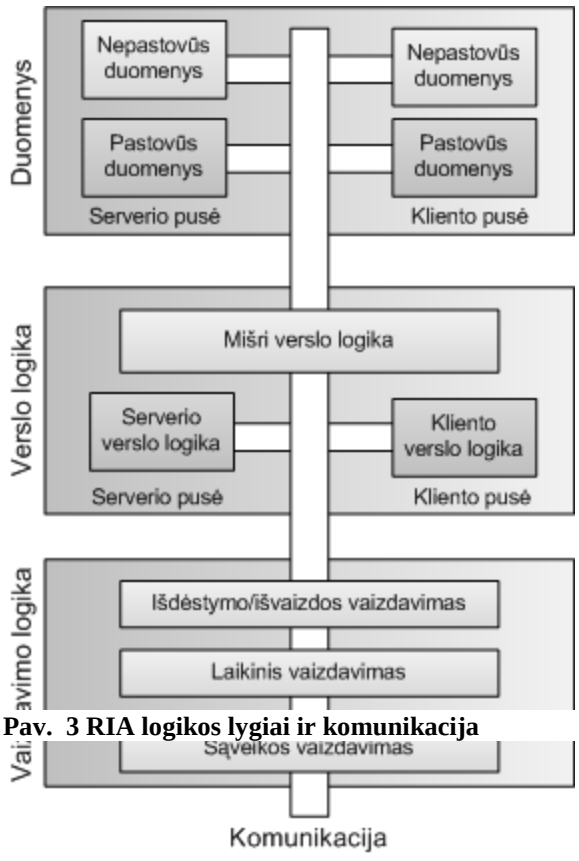
A tipo programos yra storo kliento programos. Jos pasižymi tuo, kad visi logikos lygmenys veikia kliente, tik duomenys saugomi serveryje. Komunikacija tarp serverio ir kliento vyksta duomenų bazės lygio servisų pagalba. Kadangi verslo logika veikia kliento pusėje, tai gali sukurti saugumo problemų.

B tipo programos dalijasi verslo logiką serveryje ir kliente. Tai vienas iš geriausių architektūros pasirinkimų, jei teisingai parenkama, kurią verslo logiką iškelti į klientą, ir kurią į serverį. Kliente geriausia palikti pagalbinius skaičiavimus, bei pirminį duomenų apdorojimą (pvz. duomenų teisingumo tikrinimą). Verslo logika serverio pusėje, bet kokiu atveju turi atlikti duomenų tikrinimą. Pasirinkus šį architektūros tipą iškyla verslo logikos kodo dubliavimo problema.

C tipo programos visą verslo logiką atlieka serveryje. Tai yra rekomenduojamas architektūros pasirinkimas [8]. Šiuo atveju išvengiama dubliavimo, ir visos saugumui jautrios operacijos atliekamos serveryje.

D tipo, arba plono kliento architektūra dažniausiai būdinga AJAX tipo RIA programoms. Jos pasižymi tuo, kad serveryje atlieka dalį vaizdavimo logikos, pvz. generuoja specifinį HTML kodą. Kompiliuojamuose RIA karkasuose šis būdas yra sunkiai įmanomas.

Kartais išeinama iš šių grynų tipų ribų, ypač norint pasiekti sklandesnį darbo pojūtį ar greitesnį veikimą. Tai yra įmanoma dėl padidėjusių RIA technologijų galimybių. Pavyzdžiui, kliento pusėje galima saugoti pastovius duomenis, taip pat vykdymo aplinka yra pakankamai naši, tam kad būtų galima verslo logiką atlikti kliento kompiuteryje. Šis sudėtingesnis logikos projektavimas pavaizduotas 3 paveiksle. Projektuojant sistemą į šiuos aspektus reikia atsižvelgti, kai kuriama detali sistemos architektūra [4].



Pav. 3 RIA logikos lygiai ir komunikacija

Duomenų lygmenyje išskiriami šie aspektai:

- Laikini duomenys kliento pusėje. Tai duomenys, kurie yra užkraunami į operatyviąją atmintį programos vykdymo metu. Kai programa išjungiama, jie dingsta.
- Pastovūs duomenys kliento pusėje. RIA palaiko duomenų saugojimą kliento kietajame diske. Šie duomenys tam tikrais atvejais gali būti labai svarbūs sistemos veikimui, pavyzdžiui gali leisti programai veikti be interneto ryšio.
- Laikini duomenys serverio pusėje. Dalis su klientu susijusių duomenų laikoma serveryje, kai klientas atsijungia, jie dingsta (pvz. sesijos kintamieji).
- Pastovūs duomenys serverio pusėje. Tai duomenų bazės ir kitos duomenų saugyklos.

Tam kad užtikrinti sklandų RIA sistemos veiklą reikia detaliai suprojektuoti komunikaciją tarp programos ir šių duomenų saugyklų. Daugiausia dėmesio reiktų atkreipti projektuojant į pastovių duomenų kliente saugyklą, nes tai gali sutaupyti daugiausiai duomenų komunikacijos kaštų.

Verslo logika gali būti paskirstyta įvairiai: kliente, serveryje arba mišriai. Projektuojant verslo logiką abstrakčiame lygyje galima neatsižvelgti į naudojamą technologiją ir darbų pasiskirstymą. Tačiau toliau detalizuojant projektą reikia išskirti, kuri dalis darbų atliekama serveryje, ir kuri dalis atliekama kliente. Verslo logikos skaidymo kriterijai yra šie: jautrumas saugumui ir darbo eigos sklandumas.

Projektuojant vaizdavimo logiką reikia išskirti šias vaizdavimo logikos rūšis:

- Išdėstymas/išvaizdos vaizdavimas. Apibrėžia vartotojo sąsajos pozicionavimą, dydžius bei išvaizdą.
- Laikinis vaizdavimas. Apibrėžia sąsajos komponentų grafinius efektus, periodinius veiksmus.
- Sąveikos vaizdavimas. Apibrėžia galimus vartotojo sąveikos veiksmus ir komponentų reagavimą į juos.

2.4. RIA sistemų projektavimo metodologija

Sparčiai didėjanti RIA svarba reikalauja sistematinių projektavimo metodų [3]. Tačiau šie metodai dar nėra gerai išvystyti, o didžioji dalis RIA programų projektuojamos intuityviai. Kita problema yra ta, kad RIA technologijos yra nesuderinamos, todėl dažnai kuriami projektavimo įrankiai, skirti tik tai technologijai.

RIA programose dažniausiai modeliuojama verslo ir vaizdavimo logika. Nors šioms programoms modeliuoti tinka ir standartinis UML modelis, tačiau yra sukurti specifiniai metodai, kurie atsižvelgia į internetinių sistemų specifiką. Abstrakčiame projektavimo lygyje siūlomi tokie projektavimo metodai: UWE (UML based Web Engineering, liet. - UML paremtas interneto programų projektavimas), RUX (Rich User eXperience Model, liet. – Turtingos vartotojų patirties modelis), WebML (Web Modeling Language, liet. - Interneto modeliavimo kalba). Taip pat aprašomi įvairūs kodo generavimo įrankiai.

2.4.1. UWE metodologija

UWE yra sistemiškas modeliavimu paremtas internetinių programų projektavimo metodas. UWE metodika remiasi „atsakomybių atskyrimu“ (angl. “separation of concerns“) ir modeliuoja turinį, navigacijos struktūrą, verslo procesus bei vaizdavimo logiką atskirai. UWE įgyvendina modeliavimu pagrįstą programų įrangos kūrimą, ir leidžia transformacijas iš modelių į programos kodą. UWE yra UML praplėtimas, todėl jis gali būti įgyvendintas daugelyje UML įrankių [5].

2.4.2. RUX metodologija

RUX metodas yra interaktyvių interneto sąsajų modeliavimo metodas. RUX metodas išskiria tris skirtingo detalumo sąsajos lygmenis: abstrakti sąsaja, konkreti sąsaja ir galutinė sąsaja. Abstrakti sąsaja leidžia pavaizduoti komponentus, kurie yra nepriklausomi nuo RIA platformos, be jokių erdvinių, išvaizdos ar elgsenos priklausomybių. Konkreti sąsaja išskiria erdvines, laikines ir elgsenos savybes. Galutinė sąsaja talpina informaciją, reikalingą programos kodo generavimui. Perėjimui tarp skirtingų sąsajų naudojamos modelio transformacijos, kurios vis konkretizuoja modelį, o galiausiai sugeneruoja programos kodą. RUX metodas gali būti integruojamas su kitais metodais, kurie modeliuoja verslo logiką ir duomenis [5].

2.4.3. WebML metodologija

WebML metodologija leidžia konceptualiame lygyje projektuoti informacinės sistemos duomenų struktūras, verslo logiką, navigaciją bei išvaizdą [4]. Ši metodologija yra pritaikyta įprastų interneto programų projektavimui, tačiau tinka ir RIA projektavimui.

2.4.4. UML projektavimo ir kodo generavimo metodologijos

Kitos metodologijos remiasi UML modeliavimu ir kodo generavimu. Įrankiai, palaikantys UML projektavimo metodologijas ir RIA kalbų kodo generavimą:

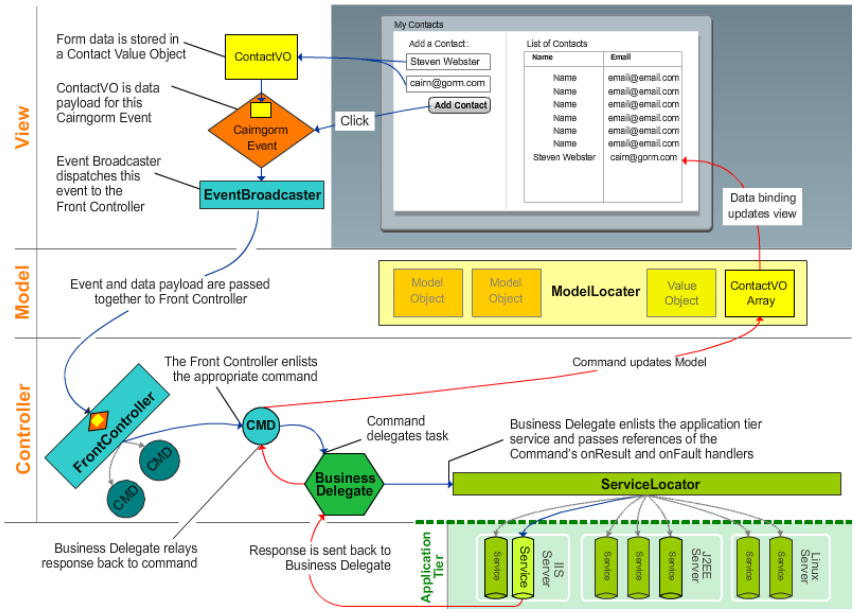
- Enterprise Architect[5]
- UMLet[6]
- Cairngen[7]

[5] http://www.sparxsystems.com/products/ea/index.html
[6] http://www.umlet.com/
[7] http://www.ericfeminella.com/blog/cairngen/

2.5. RIA mikroarchitektūros

Visos RIA technologijos turi gausias klasių ir komponentų bibliotekas. Šios bibliotekos iš dalies apibrėžia tai, kaip bus suformuota programos architektūra. Tačiau kai kuriama programa tampa pernelyg sudėtinga, tai technologijos siūlomos architektūros gali neužtekti, tam kad būtų išlaikoma programos struktūra. Tokiu atveju gali būti naudojami mikroarchitektūrų karkasai. Šie karkasai apibrėžia projekto katalogų struktūrą, bei kiekvienos klasės paskirtį. Populiariausi mikroarchitektūrų karkasai remiasi Modelio-Vaizdo-Valdiklio (angl. Model-View-Controler, toliau - MVC) principu. Toliau pateikiami populiariausi Flex technologijos MVC karkasai.

Cairngorm – atviro kodo Flex technologijos mikroarchitektūra, rekomenduojama Adobe kompanijos. Sukurta palengvinti sudėtingą duomenų ir būsenos komunikaciją tarp serverio ir kliento, išlaikant nesudėtingą vaizdavimo komponentų logiką [10]. Klasės skirstomos į tokius stereotipus – reikšmių objektus (angl. - value object), įvykius, komandas, verslo delegatus (angl. - business delegate). Cairngorm architektūros schema pavaizduota 4 paveiksle.

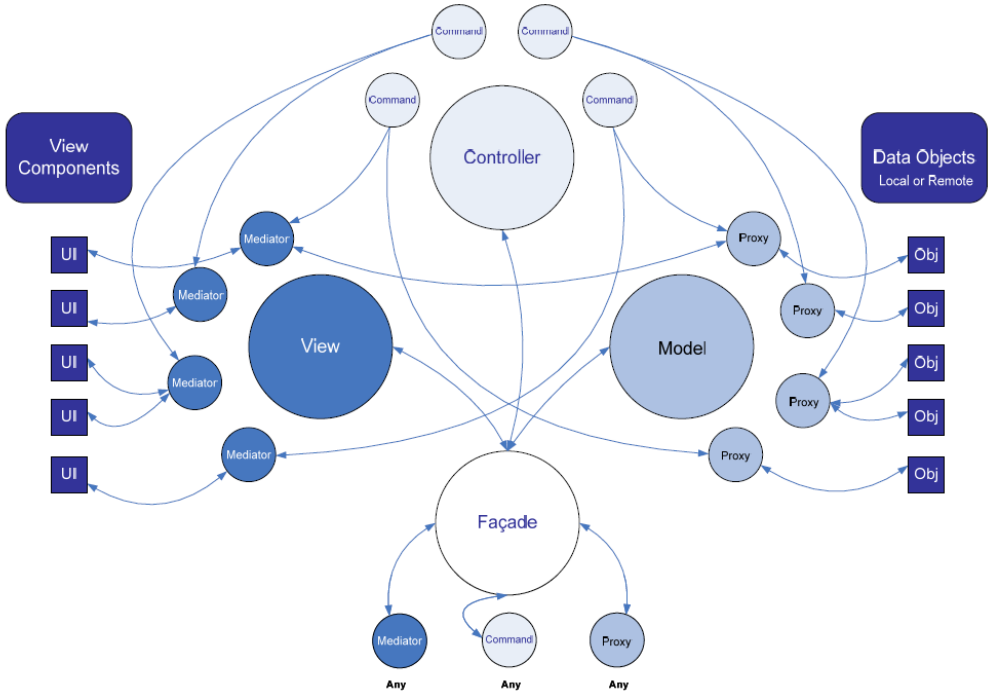


Pav. 4 Cairngorm mikroarchitektūros schema

PureMVC – tai alternatyvi atviro kodo Flex mikroarchitektūra [11]. Ši mikroarchitektūra pasižymi tuo, kad yra nepriklausoma nuo pasirinktos įgyvendinimo technologijos ir įgyvendina grynus MVC principus. Programos klasės pagal stereotipus skaidomos į tris tipus:

- Delegatus – vaizdavimo klases, kurios yra atsakingos už vaizdavimo komponentų įvykių apdorojimą ir būsenos atnaujinimą.
- Komandas – valdiklio klases, kurios yra atsakingos už verslo logiką.
- „Įgaliotas“ (angl. - proxy) klases –modelio klases, kurios yra atsakingos už duomenų saugojimą ir komunikaciją su išorinėmis duomenų talpyklomis, bei paslaugų tarnybomis.

Ši mikroarchitektūra pasižymi tuo, kad palaiko modulinę programos struktūrą. Kiekvienas modulis turi savo delegatus, komandas ir įgaliotas klases. Dėl šios priežasties PureMVC mikroarchitektūra buvo pasirinkta kuriant Incidentų Valdymo Sistemą (žr. 3.2.2 skyrių). PureMVC schema pavaizduota 5 paveiksle.



Pav. 5 PureMVC mikroarchitektūros schema

2.6. Darbo pasidalijimas tarp dizainerių ir programuotojų

RIA dažnai yra naudojamos vaizdžiam informacijos pateikimui, todėl programos dažnai yra kuriamos dviejų tipų kūrėjų – dizainerių ir programuotojų. Tokiu atveju iškyla programos išvaizdos ir funkcijų atskyrimo problema. Dizaineriai nupiešia programos lango vaizdą, o

programuotojai naudodamiesi tuo vaizdu turi sudaryti programos komponentų stilių schemą ir pritaikyti reikiamus komponentus. Šis darbas yra monotoniškas ir varginantis, jei nenaudojami automatiniai įrankiai. Taip pat papildomos problemos iškyla, kai reikia keisti programos išvaizdą.

Ši problema RIA technologijose sprendžiama naudojantis deklaratyvios komponentų aprašymo kalbos galimybėmis (Flex atveju – MXML, SilverLight atveju – XAML). Deklaratyvia kalba aprašomas komponentų išdėstymas ir išvaizda. Programos funkcionalumas prijungiamas pridedant įvykių apdorojimo funkcijas. Tokiu būdu komponentas gali būti vienas, bet jis gali būti kuriamas skirtingais įrankiais. Flex technologija turi specialius įrankius programuotojams (Adobe Flex Builder) ir dizaineriams (Adobe Flash Catalyst). Tokiu pačiu principu sukurti ir Silverlight įrankiai programuotojams (Microsoft Visual Studio) bei dizaineriams (Microsoft Expression Blend).

2.7. RIA architektūriniai ir technologiniai aspektai

Šiame skyriuje pateikiami svarbiausi RIA architektūriniai aspektai bei technologinės naujovės, kurias reikia išnaudoti projektuojant RIA tipo programas. Skyriaus pabaigoje pateikiamas kiekvieno aspekto palaikymo įvertinimas trijose iš populiariausių RIA technologijų (Flex, Silverlight, JavaFX) [14, 15, 16, 17].

2.7.1. Komponentų aprašymas deklaratyvia kalba

Viena iš RIA ypatybių yra komponentų aprašymas deklaratyvia kalba. Deklaratyvi kalba - tai kalba, kuri leidžia aprašyti sistemos komponentų struktūras ir sąveikas, kuria nors iš struktūrizuotų žymėjimo kalbų (pvz. XML). Flex atveju tai yra MXML, SilverLight atveju – XAML. JavaFX programos struktūrą aprašo sava scenarijų kalba, kuri leidžia struktūrizuotas duomenų struktūras.

2.7.2. Modulinės architektūros palaikymas

Didėjant programos apimčiai dažnai iškyla poreikis išskaidyti programą į dalis. Taip sutaupomi tinklo resursai, nes nereikia atsiųsti programos dalių, kurios galbūt niekada nebus naudojamos. Modulių naudojimas leidžia sukurti laisvesnę programos architektūrą.

2.7.3. Vektorinė grafika

Visos pagrindinės RIA technologijos leidžia prigimtinį (angl. - native) vektorinės grafikos palaikymą. Tai reiškia, kad visus grafinius objektus ir komponentus galima didinti ar mažinti neprarandant kokybės ir su mažais procesoriaus skaičiavimų kaštais.

2.7.4. Vartotojo sąsajos komponentai

RIA technologijos išsiskiria vartotojo sąsajos komponentų gausa. Į komponentų sąrašą paprastai įeina tokios komponentų klasės:

- Duomenų įvedimo komponentai;
- Duomenų teisingumo tikrinimo komponentai;
- Išdėstymo komponentai;
- Grafinių duomenų vaizdavimo komponentai;
- Vaizdo ir garso atkūrimo komponentai;
- Vartotojo sąveikos komponentai (pvz. komponentų tempimas pele).

2.7.5. Sąsajos stilių ir apipavidalinimo palaikymas

Tam, kad programos išvaizdos aprašymas būtų paprastesnis, dažniausiai naudojamos CSS tipo stilių aprašymo sistemos. Taip pat kartais leidžiama stilius sukaupti į stilių paketus – programos apipavidalinimus (angl. - skins) – kuriuos pakeitus iškarto pasikeičia visos programos stiliai.

2.7.6. Įvykiais paremta sistema

RIA technologijos dažniausiai veikia komponentų įvykių pagrindu. Dažniausiai komponentai turi gausią įvykių aibę, su reikiamu abstrakcijos lygmeniu.

2.7.7. Daugiagijiškumo palaikymas

Dažnai vykdant ilgus skaičiavimus nenorima laukti kol jie pasibaigs. Taip pat nenorima, kad vartotojo sąsaja taptų nereaguojančia. Šių problemų padeda išvengti kelių gijų veikimo vienu metu palaikymas. Kartu tai sukelia saugumo problemas, nes kelių gijų vykdymas vienu metu gali padaryti kliento kompiuterį nestabiliu.

2.7.8. Duomenų saistymo palaikymas

Tam, kad būtų galima atskirti struktūrinį išvaizdos aprašymą ir vaizdavimo logiką, naudojamas duomenų susaistymas. Ši technologija leidžia pereiti per duomenų sąrašus, pasiekti objektų savybes ir perduoti jų reikšmes vaizdavimo komponentams.

2.7.9. Daugiakalbystės palaikymas

Dažnai programas reikia pritaikyti skirtingomis kalbomis kalbantiems vartotojams. Kuriant RIA programas su daugiakalbystės palaikymu naudojamos dvi strategijos: programos kompiliuojamos kiekvienai kalbai atskirai, arba sukompiliuojamos visos kalbos ir leidžiama pasirinkti norimą kalbą programos vykdymo metu.

2.7.10. Vietinė duomenų saugykla

RIA technologijos leidžia kliento pusėje turėti pastovią duomenų saugyklą. Flex atveju saugykla tipiškai yra apribota 100KB dydžiu. SilverLight atveju – 1024KB.

2.7.11. Sinchroninė ir asinchroninė komunikacija

RIA technologijos leidžia asinchroninę komunikaciją su serveriu, tai yra klientas neturi laukti, kol grįš atsakymas iš serverio. Sinchroninė komunikacija galima, bet nerekomenduojama. RIA technologijos komunikacijai su serveriu dažnai palaiko įvairius protokolus ir formatus. Jų pasirinkimas priklauso nuo sistemos architektūros, našumo, saugumo ir pakartotinio panaudojimo reikalavimų. Šie metodai plačiau aprašomi 2.8 skyriuje.

2.7.12. Integracijos su serveriu palaikymas

Kuriant sistemą norisi turėti vieningą programavimo aplinką – bendrą programavimo kalbą ir vieną teksto redagavimo įrankį. Geriausia integracija yra palaikoma SilverLight technologijoje – kūrimas vyksta viena iš pasirinktų .NET programavimo kalbų, o visas darbas gali būti atliekamas su vienu Visual Studio įrankiu. Flex technologijos atveju kliento dalis kuriama atskirai (naudojant Flex Builder įrankį), naudojant specializuotą kalbą – ActionScript. JavaFX atveju integracija taip pat gera, nes išnaudojamos Java kalbos bibliotekos.

Gera integracija su serverio technologija gali būti dviprasmiškai vertinama. Iš vienos pusės tai yra gerai, nes gaunamas sklandesnis programos kūrimo procesas. Iš kitos pusės integruotos technologijos apribojamos tik savo šeimos technologijų naudojimu.

2.7.13. Multimedijos įvedimo šaltinių palaikymas

Flex ir JavaFX technologijos leidžia be klaviatūros ir pelės įvedimo taip pat įrašyti vaizdą bei garsą. Silverlight technologija taip pat greitu laiku turėti šią galimybę.

2.7.14. Architektūrinių aspektų palyginimas

Aukščiau paminėti aspektai yra apibendrinti ir palyginti pagal technologijas 1 lentelėje.

Lentelė 1 Architektūrinių aspektų palyginimas pagal technologijas

	Flex	SilverLight	JavaFX
Komponentų aprašymas deklaratyvia kalba	Palaikomas	Palaikomas	Nestandartinis palaikymas
Modulinės sistemos palaikymas	Yra	Yra	Nėra
Vektorinė grafika	Labai geras palaikymas	Labai geras palaikymas	Geras palaikymas
Vartotojo sąsajos komponentai	Labai gausus komponentų spektras	Gausus komponentų spektras	Mažai komponentų
Sąsajos stilių ir apipavidalinimo palaikymas	Geras palaikymas	Yra palaikoma	Nepalaikoma
Įvykiais paremta sistema	Palaikoma	Palaikoma	Palaikoma
Daugiagijiškumo palaikymas	Nepalaikoma	Palaikoma	Nepalaikoma
Duomenų saistymo palaikymas	Palaikoma	Palaikoma	Palaikoma
Daugiakalbystės palaikymas	Palaikoma	Palaikoma	Nepalaikoma
Vietinė duomenų saugykla	Palaikoma	Palaikoma	Nepalaikoma
Duomenų komunikavimo protokolai ir formatai	XML, SOAP, JSON, AMF, RTMP, TCP jungtys	XML, SOAP, REST, JSON, TCP jungtys	XML, SOAP, REST, JSON, RMI, TCP jungtys
Integracijos su serveriu palaikymas	Kliento dalis atskirta	Integruota	Integruota
Multimedijos įvedimo šaltinių palaikymas	Yra	Nėra	Yra

Iš lentelės matome, kad technologiškai Flex ir Silverlight yra lygiavertės technologijos, o JavaFX technologija yra mažiau išvystyta.

2.8. Duomenų perdavimo protokolai ir formatai

RIA technologijos palaiko įvairius duomenų perdavimo protokolus. Didžioji dalis protokolų remiasi į standartinį interneto protokolą – HTTP. Tačiau šis protokolas nėra gerai

pritaikytas interaktyvių duomenų siuntimui, todėl buvo sukurti RIA technologijoms pritaikyti protokolai.

2.8.1. RTMP

RTMP (Real Time Messaging Protocol – Realaus laiko žinučių protokolas) yra Adobe kompanijos sukurtas patentuotas protokolas, skirtas garso, vaizdo ir duomenų siuntimui tarp Flash kliento programos ir serverio. Ateityje šis protokolas turėtų būti paskelbtas atviru. Šis protokolas nuo HTTP protokolo skiriasi tuo, kad palaiko ryšio būseną – kliento sesiją. Dažniausiai jis veikia virš TCP protokolo ir naudoja nustatytą serverio prievadą, tačiau kartais (kai serverį ir klientą skiria ugniasienė), jis gali būti veikti naudodamas HTTP (arba HTTPS) protokolą kaip pagrindą.

RTMP protokolas siunčiamiems paketams prideda tik labai mažas antraštes, todėl yra labai našus. Jis gerai tinka vaizdo ir garso perdavimui. Duomenų perdavimui šiuo protokolu žinutės dažniausiai koduojamos AMF formatu. Dėl savo sesijos išlaikymo galimybės jis leidžia įgyvendinti „serverio stūmimo“ (angl. - “server push“) principą.

RTMP protokolą kliento dalyje palaiko Flash programos. Serverio dalyje jis yra palaikomas šių serverių: Adobe Flash Media Server, Adobe LiveCycle Enterprise Suite, Wowza Media Server, WebORB, Red5, FluorineFX (paskutiniai du yra atviro kodo).

2.8.2. Duomenų formatai

Toliau aprašomi dažniausiai RIA programose naudojami duomenų formatai. Visi formatai, išskyrus AMF, veikia tik HTTP protokolo pagrindu.

XML (Extensible Markup Language – Išplečiama žymėjimo kalba) – tai standartizuota bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba. XML aprašo duomenis žymėmis, atributais bei reikšmėmis. Žymės gali turėti žemesnio lygio žymes, taip sudarydamos hierarchinę struktūrą. Šis formatas leidžia kūrėjui naudoti savo suprojektuotą duomenų formatą, kuris skirtųsi naudojamomis žymėmis bei atributais.

SOAP – specifinis XML formatas pateikiantis vieningą schemą duomenų apsikeitimams. Dėl šios savybės jis yra standartinis protokolas duomenų apsikeitimui su trečių šalių paslaugų (duomenų) tiekėjais. Šis protokolas vieningu būdu aprašo nutolusių sistemų metodų iškvietimą,

todėl dažniausiai yra naudojamas interneto tarnybų kvietimui. RIA technologijos dažniausiai apjungia SOAP naudojimą su automatiniu duomenų perdavimo klasių generavimu iš interneto tarnybų aprašų.

JSON (JavaScript Object Notation – JavaScript objektų notacija) – glaustas objektinių duomenų apsikeitimo formatas. Remiasi JavaScript scenarijų kalbos objektų ir masyvų užrašymo būdu. Jis yra dažniausiai naudojamas AJAX programose, kaip alternatyva XML formatui. Taip pat gali būti naudojamas ir standartinėse RIA programose.

AMF (Action Message Format – Veiksmo žinučių formatas) – yra dvejetainis formatas, sukurtas naudojantis SOAP formato principais, skirtas duomenų apsikeitimui tarp Flash kliento ir serverio. Šis formatas yra sukurtas Adobe kompanijos, jo specifikacija yra atvira [9]. AMF formatas pasižymi dideliu siunčiamos informacijos suspaudimu.

Visi šie formatai yra naudojami RIA technologijose, o kai kurie netgi turi standartinių klasių generavimo palaikymą. Kiekvienas iš šių formatų turi specifinę taikymo sritį, pvz. SOAP yra dažniausiai naudojamas, kai reikia operacinio suderinamumo. Projektuojant RIA programas reikia numatyti kuriuos formatus pasirinkti. Šių formatų našumas tiriamas 4.4 skyriuje.

2.9. Saugumo problemos

RIA technologijos atneša daug naujų galimybių, bet kartu padaugėja ir vietų, kurios gali sukelti saugumo problemų. Saugumo problemas galima skaidyti į dvi klases - susijusias su programa ir susijusias su platforma. Su programa susijusios problemos yra tokios pačios, kaip ir kitų rūšių programose: duomenų integralumo, pasiekiamumo ir konfidencialumo. Specifinės RIA platformų saugumo problemos yra šios[18]:

- Silpnas kodo saugumo lygių palaikymas;
- Galimybė parsisiųsti ir užkrauti nepatikrintas bibliotekas;
- Spragos programos izoliacijoje nuo iškviečiančiųjų programų;
- Duomenų saugojimas kliento kompiuteryje leidžia vartotojų sekimą ir piktavališką kietojo disko užpildymą;
- Piktavališkų programų suinstaliavimas apgaunant vartotoją

RIA programas galima skirstyti į du tipus pagal skirtingas programų teises: paleidžiamas iš naršyklės (pvz. Flex) ir paleidžiamas iš darbastalio (pvz. AIR). Skirtingas teisių lygis kartu apsprendžia ir galimybę pakenkti vartotojui. Paleidžiamos iš naršyklės programos potencialiai yra mažiau kenksmingos. Paleidžiamas iš darbastalio RIA programas reikia traktuoti kaip ir įprastas darbastalio programas. Technologijas kuriančios kompanijos atranda ir ištaiso saugumo skyles, bet programų naudotojai, administratoriai ir kūrėjai turi atkreipti labai didelį dėmesį į RIA programų saugumą ir vadovautis saugaus kūrimo bei naudojimosi principų[19].

2.10. RIA panaudojimo sritys

Kiekviena technologija turi savo stipriąsias ir silpnąsias savybes. Pagal šias savybes galima nustatyti kokiose srityse technologiją galima taikyti, o kokiose geriau paieškoti alternatyvų. RIA technologijos yra stiprios interaktyvia sąsaja, interneto formatų palaikymu, didele veikimo aplinkų aprėptimi.

Sritys, kuriose rekomenduojama naudoti RIA technologijas:

- Interaktyvių komponentų (angl. „widget“) kūrimas. RIA programos labai dažnai naudojamos kaip įterptiniai priedai tinklalapiuose ir tradicinėse interneto sistemose. RIA komponentai dažniausiai naudojami, kai reikia sudėtingo grafinio duomenų vaizdavimo, garso bei vaizdo atkūrimo komponentų.
- Informacijos kioskų ir mokymo priemonių programinė įranga. RIA technologija puikiai tinka, kai reikia sukurti nesudėtingas informavimo sistemas arba interaktyvias mokymo programas.
- Grafikos ir diagramų kūrimo programinė įranga. Išvystytas RIA technologijų grafikos palaikymas leidžia sukurti sudėtingas grafikos redagavimo ir diagramų kūrimo programas internete.
- Verslo valdymo sistemose. Dėl savo interaktyvumo RIA gerai tinka tada, kai reikia atlikti ilgalaikius sprendimų priėmimo darbus ir kai reikalinga sklandi vartotojo sąsaja. Tokiais atvejais puslapių principas netinka. Dažnai RIA programoms galima pritaikyti „programinės įrangos, kaip paslaugos“ principą (angl. Software as a Service - SaaS).

- Paskirstytos sistemos. Tinka paskirstytoms sistemoms, ypač taupant serverio resursus. Pavyzdys - bevielis sensorių tinklas [6]. Tinka, nes neapkrauna silpnų serverių ir leidžia „serverio stūmimo“ (angl. – server push) principą.
- Kompiuteriniai žaidimai. RIA leidžia sukurti vidutinio sudėtingumo lygio žaidimus.

Yra nemažai sričių, kuriose RIA gali pilnai pakeisti darbastalio programas. Ypač todėl, kad ta pati programa be didelių pakeitimų gali būti paleidžiama naršyklėje ir darbastalyje (Flex ir AIR, bei Silverlight ir WPF technologijų poros). RIA technologinės naujovės bei gausus interneto naudojimas turėtų pakeisti ateities darbastalio programų kūrimo principus.

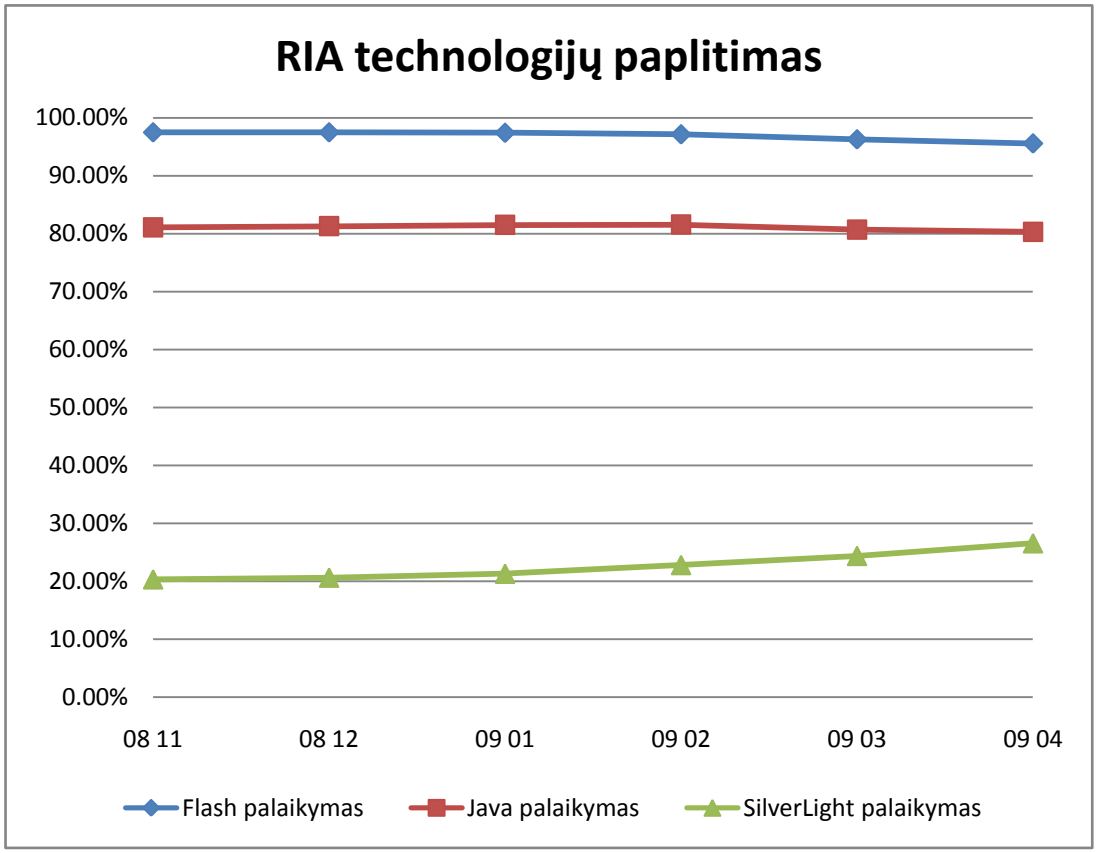
RIA savo įgyvendinimu, atvirumu ir paskirtimi labai skiriasi nuo tinklalapių ir interneto programų. Vienas iš esminių skirtumų yra tai, kad tinklalapiuose duomenys, išvaizda bei funkcijos aprašomos tekstu, o RIA technologijose išvaizda ir funkcijos yra kompiliuojamos. Dėl šios priežasties paieška per RIA programas tampa daug sudėtingesnė nei įprastuose tinklapiuose. Tuo tarpu tekstinė paieška internete yra vienas iš svarbiausių informacijos suradimo būdų. Todėl RIA technologijos ateityje neturėtų pakeisti tradicinių tinklalapių.

2.11. Technologijų palyginimas

Renkantis RIA technologiją pirmiausia reikia išanalizuoti ar sistemos reikalavimus galima patenkinti standartiniu HTML ir AJAX technologija. Jei reikalaujama didelio sistemos interaktyvumo, tai kartais galima sugalvoti priemones kaip tai įgyvendinti, naudojant įvairias naršyklės ar serverio technologijas. Tačiau nestandartinių priemonių kūrimas kainuoja daug žmogiškųjų išteklių resursų. Tokiais atvejais galima naudoti įskiepiu paremtą RIA technologiją.

Šiuo metu labiausiai paplitusi ir brandžiausia RIA technologija yra Flex. Ji pakol kas turi didžiausią programuotojų ir dizainerių bazę. Silverlight technologija šiuo metu stipriai vejasi Flex tiek galimybėmis, tiek susidomėjimu iš programuotojų pusės. Kadangi RIA programos dažniausiai yra didesnės sistemos dalis, į kurią įeina ir serverio paslaugos, tai technologijos pasirinkimas dažniausiai atsiremia į tai kokia technologija naudojama serveryje. Jei serveryje naudojama Java technologija, tai pasirinkimas greičiausiai bus Flex arba JavaFX, jei .NET – tai arba Silverlight, arba Flex.

Jei kuriamos sistemos mobiliesiems įrenginiams, tai galimas technologijos pasirinkimas yra JavaFX. Flex ir Silverlight turi mažesnį mobiliųjų įrenginių palaikymą, nes jų vykdymo aplinkos yra per didelės ir reikalauja per daug resursų [12].



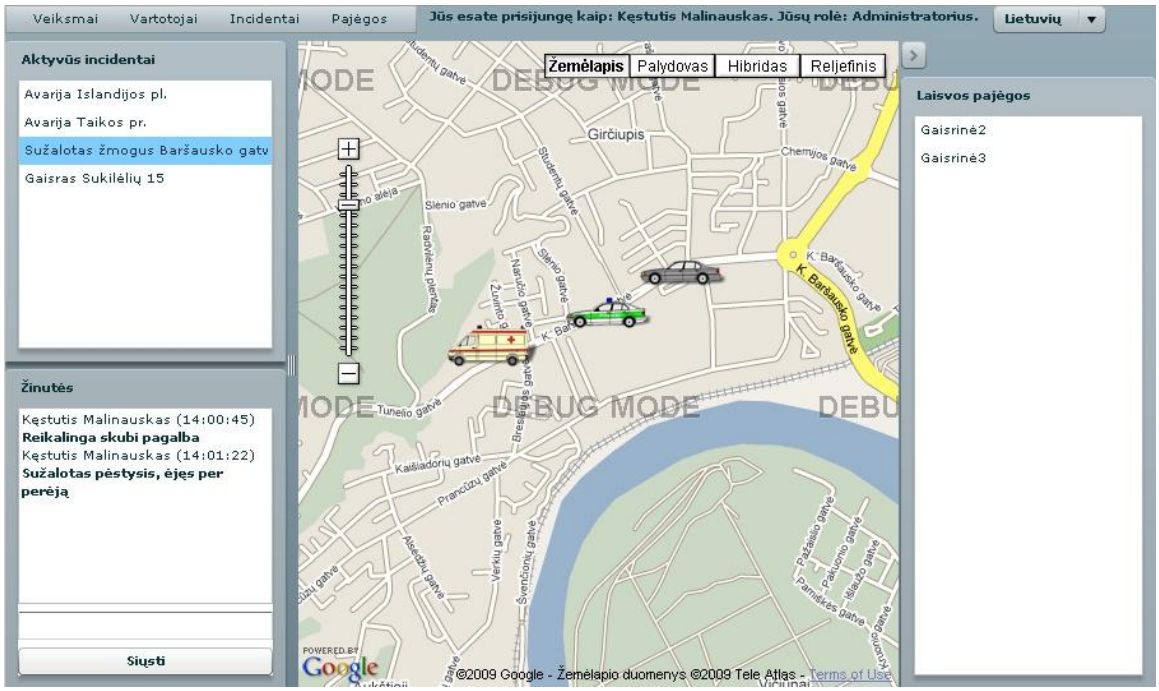
Pav. 6 RIA technologijų paplitimas

6 paveiksle parodytas RIA technologijų paplitimas vartotojų kompiuteriuose[13]. Iš jo matome, kad Flash (o kartu ir Flex) technologija yra palaikoma beveik visuose vartotojų kompiuteriuose, Java – apie 80% kompiuterių. Silverlight palaikymas yra daug mažesnis (apie 20%), tačiau jis ima didėti. Pakol kas, renkantis technologiją, reiktų atsižvelgti į mažą Silverlight paplitimą, tačiau jis pradeda didėti, ir ateityje turėtų susilyginti su kitais.

Šiuo metu Flex yra populiariausia ir geriausiai išvystyta RIA technologija. Tačiau stiprus SilverLight ir JavaFX kūrėjų įsipareigojimas šių technologijų vystymui bei gera integracija su atitinkamai .NET ir Java technologijomis padaro šias technologijas stipriomis Flex konkurentėmis.

3. Incidentų Valdymo Sistemos projektas

Kaip viena iš RIA taikymo sričių pateikiamas magistro studijų metu vykdytas Incidentų Valdymo Sistemos (IVS) projektas. Tai yra interaktyvi paskirstyta sistema gausiai išnaudojanti RIA technologijų galimybes. Galutinis sistemos vaizdas parodytas 7 paveiksle.



Pav. 7 Incidentų Valdymo Sistemos programos langas

3.1. Sistemos reikalavimai

3.1.1. Sistemos aprašymas

Įmonėse ir organizacijose labai dažnai prireikia realiu laiku sekti organizacijos viduje vykstančius procesus. Tai ypač svarbu tokiose organizacijose, kurių nesklandi veikla gali tiesiogiai lemti žmonių aukų skaičių ar didelio mąsto nuostolius. Viena iš tokių sričių yra viešosios tvarkos palaikymas ir gelbėjimo operacijų organizavimas. Organizuojamos gaisrininkų-gelbėtojų, medikų, policininkų, kariškių pajėgos tam, kad būtų neutralizuojami didelio mąsto incidentai, katastrofos ir avarijos. Tokioms užduotims reikia glaudaus visų šių pajėgų darbo, o jį organizuoja pajėgų štabas. Kuriama incidentų valdymo sistema spręs tokio pajėgų štabo organizacines problemas.

Pagrindinės sistemos funkcijos yra:

- Turimų pajėgų užregistravimas ir administravimas;
- Bendros pajėgų veiklos organizavimas įvykus incidentui. Tam panaudojamas žemėlapis, o informacija atnaujinama realiu laiku.
- Incidento metu vykdytų veiksmų ataskaitos generavimas ir aplinkos praėjusių incidentų analizei sukūrimas.

Kuriama sistema turi veikti realiu laiku, nes per didelis laukimas gali neleisti laiku sureaguoti ir gali sukelti neatitaisomus nuostolius. Taip pat sistema turi veikti patikimai ir neužstrigti lemiamu metu. Pradiniais sistemos naudojimo etapais daugelis kompiuterizuojamų funkcijų bus dubliuojamos senais metodais.

Sistema projektuojama kuo lanksčiau, kad ją būtų galima panaudoti įvairiose srityse, specializuojant atitinkamais moduliais. Numatomos sistemos panaudojimo ir išplėtimo sritys:

- Gelbėjimo operacijų ir avarijų likvidavimo organizavimas;
- Masinių renginių organizavimas;
- Įmonės resursų stebėjimas.

3.1.2. Sistemos vartotojai

Išskiriamos trys pagrindinės sistemos vartotojų kategorijos:

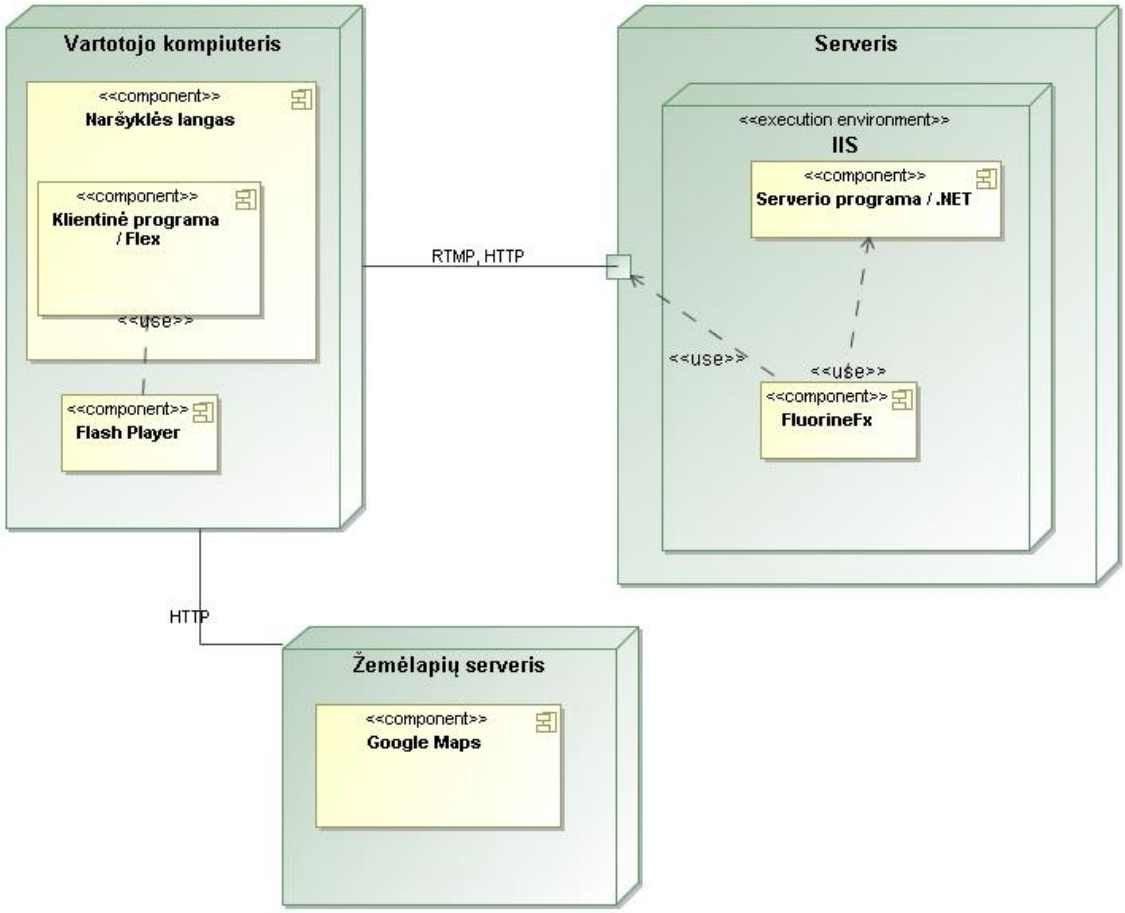
- Operatoriai – jie fiksuoja ir įvedinėja informaciją į sistemą. Jie daugiausiai dirba su sistema.
- Vadovai – jie stebi pateiktą informaciją ir priiminėja sprendimus. Sistema kuriama tam, kad palengvintų štabo vadovų darbą ir komunikaciją su pavaldiniais, todėl vadovai yra pagrindiniai sistemos vartotojai ir į jų pageidavimus yra labiausiai atsižvelgiama.
- Sistemos administratoriai – tvarko sistemos vartotojų sąrašą ir prižiūri sistemos veikimą. Jų darbas su sistema yra epizodinis.

3.1.3. Diegimo aplinka

Sistema įdiegta į štabe esantį serverį, kuris bus vietiniu tinklu prijungtas prie štabo kompiuterių. Vartotojai, norėdami naudotis programa, prisijungia prie serverio. Programa automatiškai parsiunčiama ir paleidžiama vartotojo kompiuteryje. Vartotojo kompiuteryje turi būti įdiegta:

- Interneto naršyklė (Firefox, Internet Explorer, Opera, Safari arba kuri kita).
- Flash Player 9 (ar naujesnė versija).

Sistemos įdiegimo schema pavaizduota 8 paveiksle.



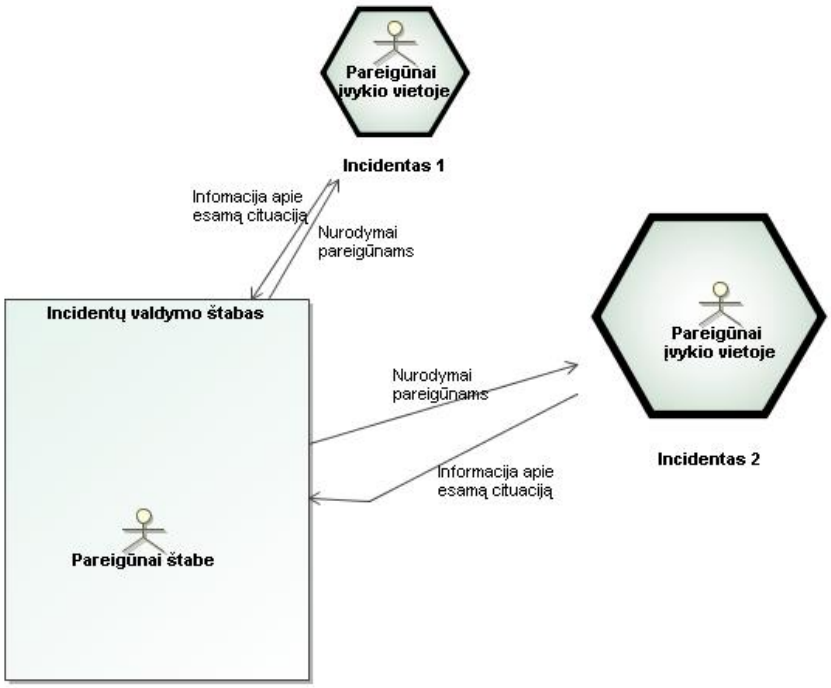
Pav. 8 Sistemos įdiegimo schema

3.1.4. Veiklos kontekstas

Štabo veikla organizuojama vykstančio incidento kontekste. Incidente dalyvaujančios pajėgos praneša apie esamą situaciją valdymo štabui, o šis duoda nurodymus pareigūnams. Veiklos kontekstas pavaizduotas 9 paveiksle.

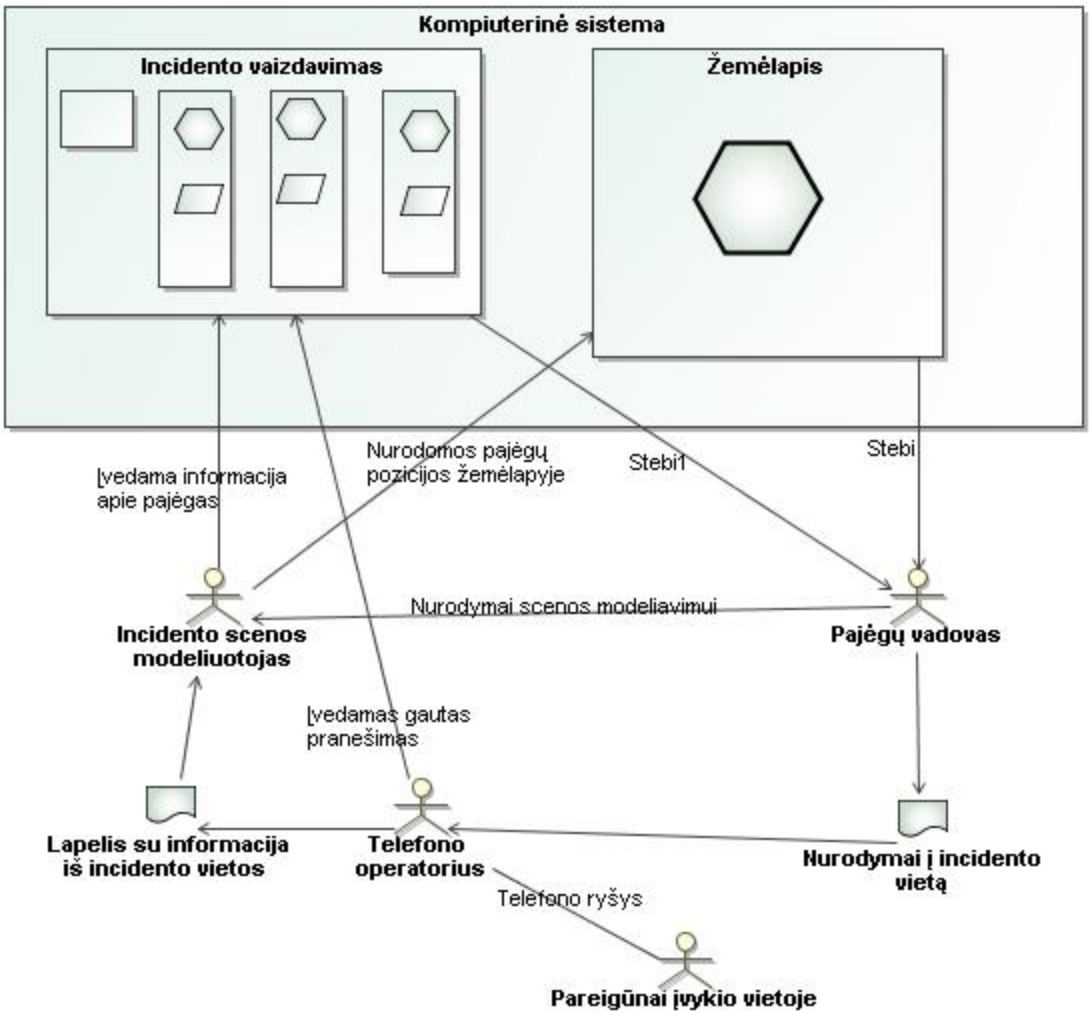
Darbas štabe dabar vyksta tokiu principu. Informaciją, gautą iš įvykio vietoje esančio pareigūno, štabe esantis telefono operatorius užrašo ant lapelio ir nuneša incidento scenos modeliuotojui. Modeliuotojas gautą informaciją pažymi magnetinėje lentoje. Pajėgų vienetai žymimi magnetiniais žymekliais, kurie skiriasi priklausomai nuo vaizduojamo objekto. Ant žymeklių taip pat rašoma papildoma informacija apie žymimą objektą. Incidentas vaizduojamas

stačiakampiu, kuris aprašo incidento situaciją. Šalia jo sudedamos žymeklių grupės, išdėstytos vertikaliai. Stulpeliai grupuojami pagal pajėgų tipus ir pajėgų vienetų būsenas (vykstantis į įvykio vietą, atvykęs, baigęs darbą ir t.t.). Visa tai reglamentuoja incidento žymėjimo taisyklės.



Pav. 9 Sistemos veiklos kontekstas

Kuriama sistema pakeičia dabar naudojamą magnetinę lentą ir sieninius žemėlapius. Naudojamas informacijos perdavimo modelis išlieka beveik toks pats. Incidento scenos modeliuotojas gali lengviau ir patogiau atvaizduoti informaciją sistemoje. Taip pat numatoma, kad telefono operatoriai gali perduoti pranešimus iš karto įvedant į sistemą. Atnaujintas štabo veiklos modelis pavaizduotas 10 paveiksle.



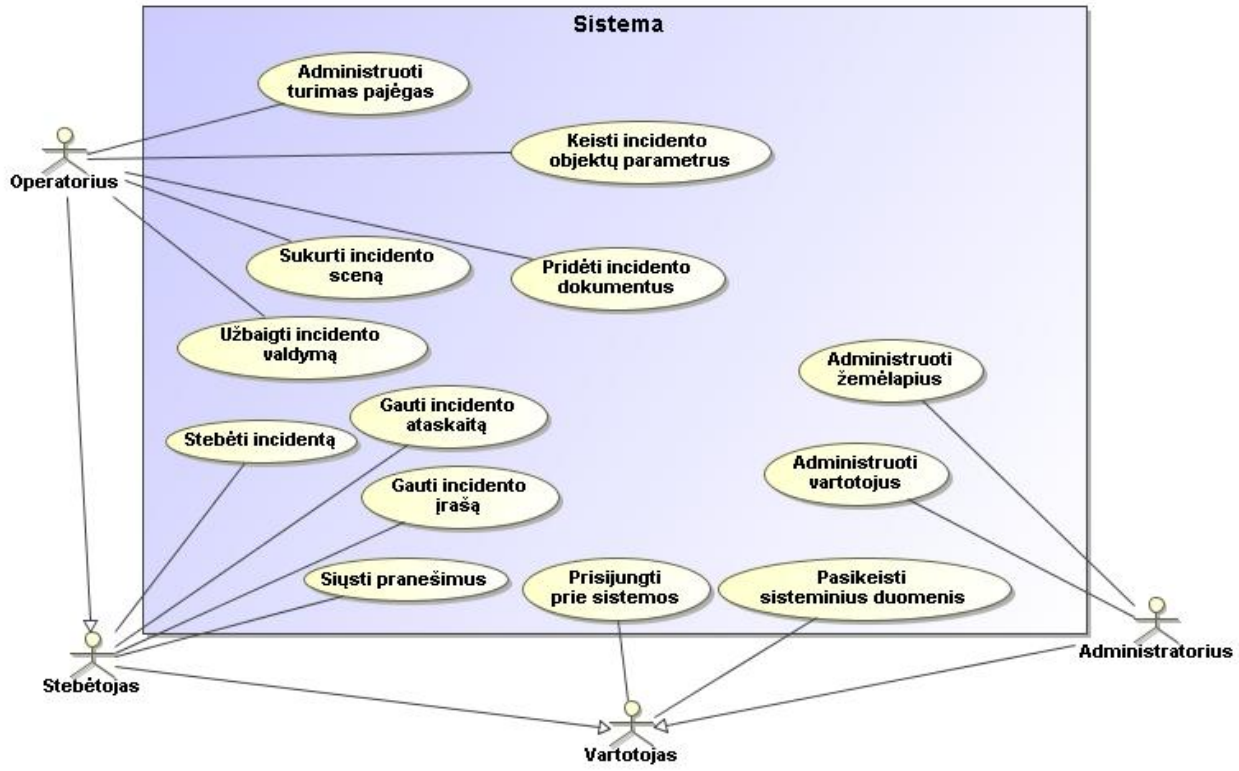
Pav. 10 Atnaujintas štabo veiklos modelis

Ateities darbams taip pat numatoma galimybė pareigūnams, esantiems įvykio vietoje, tiesiogiai teikti informaciją sistemai, kas leistų atsisakyti telefono operatorių. Tam gali būti panaudoti GPS davikliai, įmontuoti į techniką ir pareigūnų įrangą.

3.1.5. Sistemos panaudojimo atvejai

Sistemos vartotojai naudojasi sukurta sistema įsijungę naršyklę ir atsidarę sistemos internetinį puslapį. Detalus vartotojų operacijų vaizdas pavaizduotas 11 paveiksle. Sistemos panaudos atvejai yra šie:

1. Prisijungti prie sistemos. Vartotojas prisijungia prie sistemos, jam suteikiamos nustatytos teisės ir rolės.
2. Pasikeisti sisteminius duomenis. Prisijungęs vartotojas gali pasikeisti savo slaptažodį ir kitus sisteminius duomenis.
3. Administruoti vartotojus. Administratorius tvarko vartotojų sąrašą.
4. Administruoti žemėlapius. Administratorius patenka į žemėlapių administravimo režimą, kuriame gali sukurti naujus ar trinti nereikalingus žemėlapius ir tvarkyti jų sluoksnius.
5. Administruoti turimas pajėgas. Operatorius sukuria, redaguoja bei trina priregistruotus pajėgų objektus.
6. Sukurti incidento sceną. Operatorius sukuria naują incidentą ir paruošia jam žemėlapio sceną.
7. Siųsti pranešimus. Vartotojas išsiunčia su incidentu susijusį pranešimą kitiems sistemos vartotojams.
8. Pridėti incidento dokumentus. Operatorius išsiunčia dokumentus kitiems vartotojams.
9. Stebėti incidentą. Stebėtojas mato visą su incidentu susijusią informaciją.
10. Gauti incidento ataskaitą. Stebėtojas gauna ataskaitą apie incidente dalyvavusias pajėgas.
11. Gauti incidento įrašą. Stebėtojas stebi įrašytą incidento eigą.
12. Keisti incidento objektų parametrus. Operatorius kuria ir keičia incidento objektus.
13. Užbaigti incidento valdymą. Operatorius užbaigia incidentą.



Pav. 11 Sistemos panaudos atvejų diagrama

3.1.6. Funkciniai reikalavimai

Sistemos panaudos atvejus detalizuoja funkciniai reikalavimai. Toliau pateikiami aktualiausi sistemos funkciniai reikalavimai:

- Vartotojo vardas sistemoje turi būti unikalus, todėl sistema turi patikrinti ir neleisti sukurti vartotojo su jau egzistuojančiu vartotojo vardu ir formuoti atitinkamą pranešimą.
- Vartotojo duomenų keitimo metu, vartotojo įvedama informaciją turi būti patikrinama, ar tenkina duomenų turinio ir apimties reikalavimus.
- Sistemos funkcijos turi būti prieinamos tik registruotiems ir prisijungusiems sistemos vartotojams. Joks neprisijungęs vartotojas negali patekti į programos vykdomąjį langą, prieš tai neįvedęs teisingo vartotojo vardo ir slaptažodžio.
- Jei sistemos vartotojas yra užblokuotas, tai , jungiantis prie sistemos, jis turi būti apie tai informuojamas atitinkamu pranešimu.
- Pajėgos skirstomos į padalinius.

- Galima keisti pajėgų priklausymą padaliniams.
- Incidentas sukuriamas pateikiant informaciją apie incidentą ir pradinę jo vietą.
- Incidento stebėjimui pasitelkiamos įvairios perspektyvos.
- Dviejų tipų objektai: pajėgų objektai ir laikini incidento objektai.
- Laikini incidento objektai gyvuoja tik incidento metu.
- Objektai turi tvirtinimo taškus žemėlapyje.
- Laikini objektai gali turėti vieną tvirtinimo tašką arba kelis taškus, kurie sudaro tiesę arba uždarą figūrą.
- Visi žemėlapyje vaizduojami objektai turi ikonas.
- Pažymėjus kelis objektus galima atlikti veiksmus su objektų grupe.
- Kai incidentas užbaigiamas, tai jame dalyvavę objektai turi pasižymėti kaip nebedalyvaujantys.

3.1.7. Nefunkciniai reikalavimai

Nefunkciniai sistemos reikalavimai apima reikalavimus vartotojo sąsajai, sistemos saugumui, panaudojamumui, vykdymo charakteristikoms, duomenims, sistemos palaikymui ir panašiai. Esminiai sistemos nefunkciniai reikalavimai yra šie:

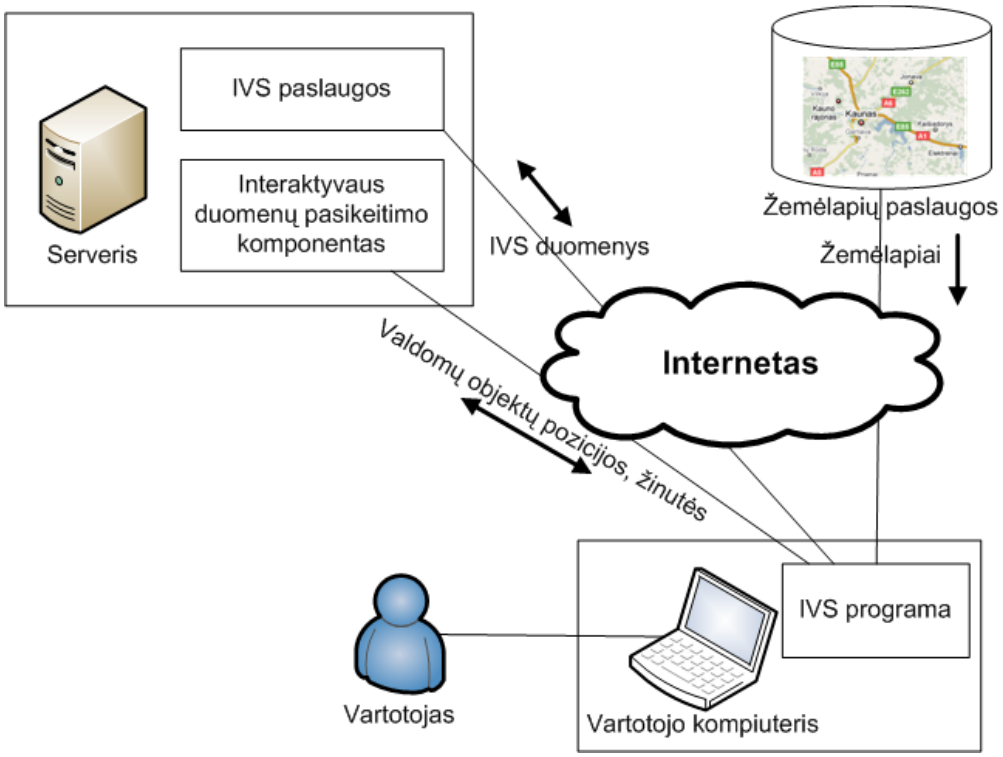
- Sistemos vartotojo sąsaja turi būti patraukli akiai ir sudominti vartotoją.
- Vartotojas turi galėti labai greitai ir efektyviai atlikti norimus veiksmus.
- Vartotojas neturi būti verčiamas prisiminti daug informacijos.
- Sistema turi veikti taip, kad vartotojas galėtų padaryti kuo mažiau klaidų.
- Vartotojas turi būti informuojamas, kai sistema atlieka ilgai užtrunkantį veiksmą.
- Vartotojas turi žinoti ar ryšys su serveriu yra veikiantis.
- Vartotojas turi galėti pasikeisti sąsajos kalbą.
- Vartotojo sąsaja turi priminti šiuo metu naudojamą incidentų valdymo metodiką.
- Štabo operatoriai be didelių apmokymų gali pradėti naudotis sistema.
- Sistema neturi versti išmokti dalykų nesusijusių su dalykine sritimi.
- Informacija tarp vartotojų turi būti perduodama su ne didesniu nei 5 sekundžių vėlinimu.
- Atnaujinimai turi būti iš karto perduoti visiems prisijungusiems vartotojams.
- Vartotojas gali pažymėti objektą ne mažesniu nei 1 metro tikslumu.

- Vykstant incidentui sistema turi veikti visą laiką.
- Jei įvyksta paprasta sistemos klaida, tai sistemos darbas negali sustoti.
- Jei įvyksta kritinė sistemos klaida, tai sistema turi veikti iš naujo ją paleidus.
- Sistema turi efektyviai veikti vienu metu prisijungus ne mažiau 20 vartotojų.
- Sistema turi būti atspari piktavalėms atakoms.
- Turi būti išlaikomas duomenų integralumas.

3.2. Sistemos architektūra

3.2.1. Sistemos veikimo principas

Incidentų Valdymo Sistema veikia kliento serverio principu, kaip paskirstyta interaktyvi sistema (12 paveikslas). IVS programa yra RIA tipo programa ir savyje talpina vaizdavimo logiką, bei dalį verslo logikos. Visų pirma IVS programa prisijungia prie IVS paslaugų serverio ir gauna būtinus veikimui duomenis: patvirtinimą, kad prisijungta teisingai, pajėgų bei incidentų sąrašus. Vartotojas priklausomai nuo jo rolės gali atlikti įvairius administravimo veiksmus.



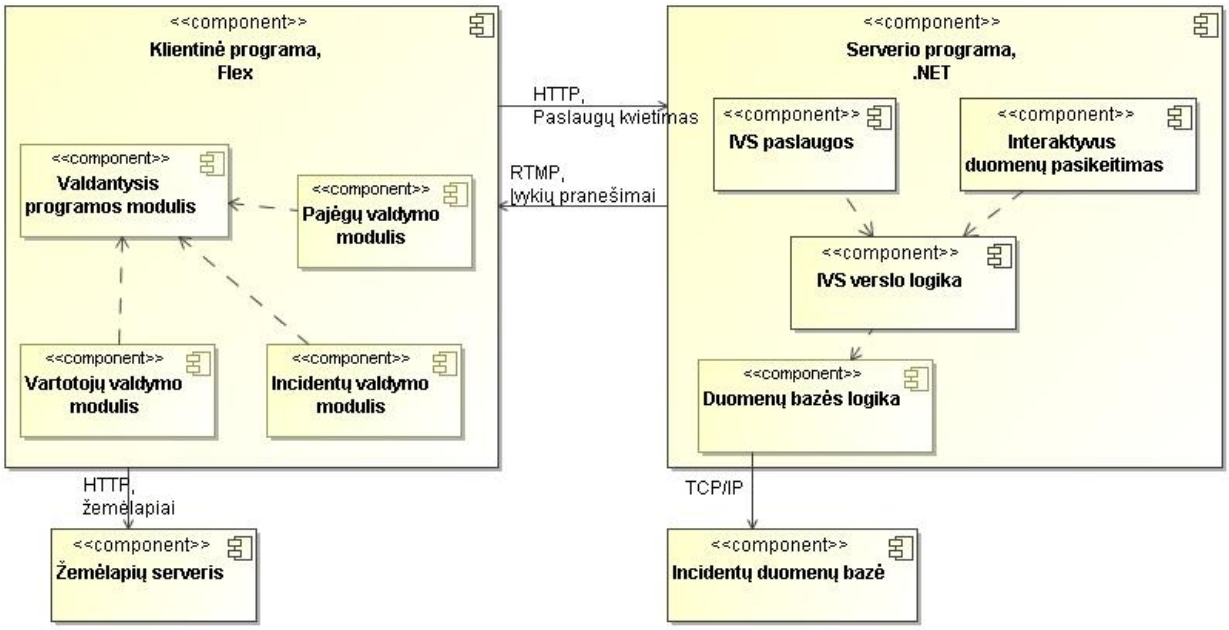
Pav. 12 Sistemos veikimo principas

Kai vartotojas nusprendžia, kad jis nori sukurti naują incidentą ar sekti vykstančio incidento veiklą, jis prisijungia prie interaktyvaus duomenų pasikeitimo komponento, kuris realiu laiku teikia duomenis visiems prisijungusiems klientams. Iš šios paslaugos komponento vartotojas gauna incidento objektų duomenis ir gali juos atnaujinti. Kai vartotojas nusprendžia, jog nori matyti incidento vaizdą žemėlapyje, programa kreipiasi į trečios šalies žemėlapių serverį ir iš jo gauna reikiamus žemėlapio vaizdus. Šie vaizdai yra išsaugomi spartinančioje atmintinėje, kad vėliau būtų greitai pasiekiami. Vartotojas taip pat gali rašyti žinutes kitiems vartotojams. Jos, kaip ir incidentų objektų pozicijų duomenys, gaunamos realiu laiku.

3.2.2. Bendroji architektūra

IVS susideda iš dviejų – serverio ir kliento – dalių. Sistemos modulių schema pavaizduota 13 paveiksle. Serverio dalis yra paremta .NET technologija, joje komunikacijai užtikrinti naudojamas trečios šalies atviro kodo komunikacijos komponentas FluorineFX. Serveryje išskiriami keturi komponentai:

1. IVS paslaugų komponentas. Šiame komponente yra aprašytos klasės ir metodai, kurie gali būti pasiekiami iš kliento. Paslaugos paviešinamos naudojant FluorineFX komponentu, kuris atlieka reikiamą konvertavimą tarp AMF formato žinučių ir klasių metodų.
2. Interaktyvaus duomenų pasikeitimo komponentas. Šiame komponente aprašomos paslaugos, kurios veikia paskelbimo-užsiregistravimo (angl. – publish/subscribe) principu. Šios paslaugos taip pat veikia tarpininkaujant FluorineFX komponentui, kuris užtikrina paslaugų veikimą RTMP protokolu.
3. IVS verslo logikos komponentas. Jame realizuotas duomenų teisingumo tikrinimas, verslo logikos taisyklės, sistemos panaudojimo scenarijai.
4. IVS duomenų bazės logikos komponentas. Jame aprašytos klasės, skirtos komunikuoti su duomenų baze.



Pav. 13 Sistemos modulių diagrama

Kliento dalis yra sukurta naudojant Adobe Flex technologiją. Kliento architektūros projektavimui naudojamas PureMVC karkasas, kuris palaiko skaidymą į modulius. Tokiu būdu visą programą galima suskaidyti į modulius, kurių kiekvienas yra sukomponuotas pagal MVC principą. Kiekvienas modulis turi savo modelio, vaizdo ir valdiklių klases. Komunikacija su serveriu vykdoma modelio klasėse. Kliento programa yra suskaidyta į pagrindinės programos modulį ir tris pagalbinius modulius. Toliau išvardinamas kiekvienas iš jų:

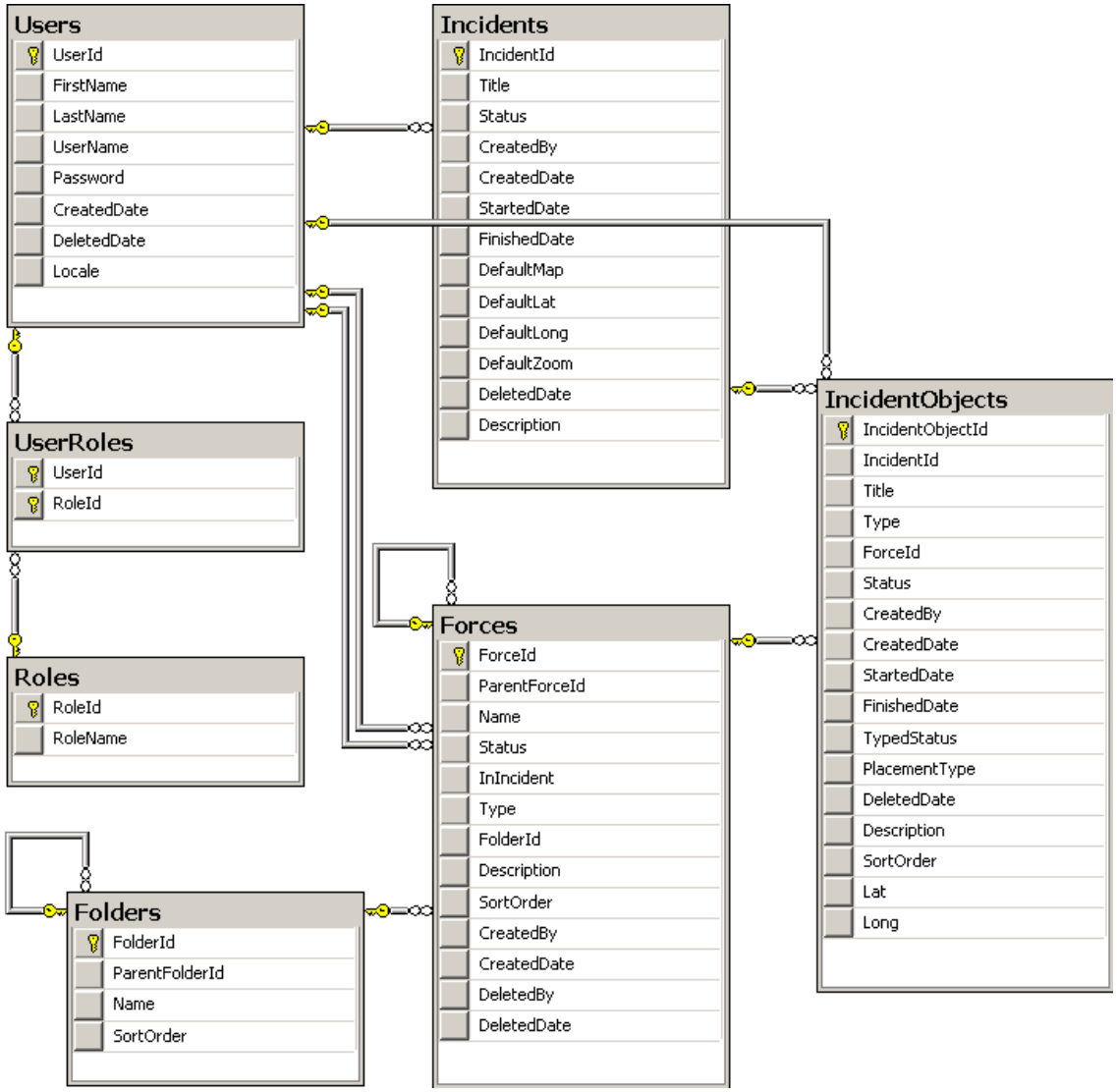
1. Valdantysis programos modulis. Jis yra atsakingas už bendrą sistemos veikimą ir modulių užkrovimą ir valdymą. Jame yra įgyvendintos prisijungimo, teisių nustatymo, kalbos keitimo, meniu ir modulių navigacijos, žinučių siuntimo funkcijos. Įgyvendinta vidinės navigacijos schema, kuri reikiamu metu užkrauna reikiamą modulį.
2. Vartotojų valdymo modulis. Jame įgyvendintos funkcijos, skirtos administruoti vartotojus. Šį modulį pasiekia tik sistemos administratoriaus teises turintys vartotojai.
3. Pajėgų valdymo modulis. Jame įgyvendintos funkcijos, skirtos administruoti štabo pajėgas.

4. Incidentų valdymo modulis. Jame įgyvendintos funkcijos, skirtos valdyti ir sekti incidentus, bei jame dalyvaujančius objektus. Modulyje naudojamas trečios šalies žemėlapių komponentas.

Kliento programos įgyvendinimas modulių pagalba leidžia sukurti mažesnį bazinį modulį, kuris yra greičiau atsiunčiamas ir paleidžiamas. Tai taip pat sąlygoja laisvesnę architektūrą geresnį darbo išskaidymą, tarp sistemą kuriančios komandos narių.

3.2.3. Duomenų bazė

Sistemos duomenų saugojimui naudojama MSSQL duomenų bazė. Jos schema pavaizduota 14 paveiksle.



Pav. 14 IVS duomenų bazės schema

3.3. Sistemos testavimas

3.3.1. Tikslai ir uždaviniai

Testavimo metu siekiama patikrinti sistemos atitikimą specifikacijai, aptikti galimas programavimo bei logikos klaidas, patikrinti atskirų sistemos dalių funkcionavimą bei kliento ir serverio dalių komunikavimą. Kuriant sistemą visų pirma atliekami vienetų testai. Šiems testams atlikti naudojamos dvi strategijos – „baltos dėžės“ ir „juodos dėžės“. Kai įsitikinama vienetų veikimo teisingumu, atliekama jų integracija. Po kiekvieno integracijos etapo įvykdomi integracijos testavimai. Galiausiai atliekami visos sistemos priėmimo, našumo ir apkrovimo testai, vartotojo sąsajos patikra ir įvertinimas.

3.3.2. Vienetų testavimas

Vienetų testavimo metu buvo ištestuotos pagrindinės sistemos funkcijos. Vienetų testavimas buvo atliktas dviem metodais - „baltos dėžės“ ir „juodos dėžės“. Baltos dėžės testavimo principas remiasi tuo, kad žinoma testuojamo komponento struktūra. Todėl baltos dėžės testavimo metu reikia ištestuoti visas galimas komponento vykdymo šakas. Juodos dėžės testavimas remiasi prielaida, kad komponento struktūra nežinoma. Todėl testuojant komponentą reikia išanalizuoti jo sąsają ir pateikti įvairias metodų parametrų reikšmes.

Įvykdytas vienetų testavimas, jo metu buvo pastebėtos ir ištaisytos programos loginės klaidos.

3.3.3. Integravimo testavimas

Integracijos testavimas vykdomas integruojant sukurtus komponentus į bendrą sistemą. Testai atliekami kiekvieną kartą, kai integruojamas naujas modulis. Kadangi sistema yra sukurta moduliniu principu, tai integracija įvyko sklandžiai ir integracijos testai aptiko mažai klaidų.

3.3.4. Priėmimo testavimas

Sukūrus sistemą vykdomas produkto priėmimo testavimas. Jis vykdomas pateikiant produktą užsakovams ir leidžiant jiems įvertinti ar produktas atitinka specifikaciją, ir ar yra tinkamas naudoti. Atliktas testavimas parodė, kad sistema tenkina iškeltus reikalavimus.

3.3.5. Aukšto lygio testavimas

Aukšto lygio testavimo metu tikrinamos visos sistemos charakteristikos: patikimumas, saugumas, greitaveika, atstatomumas, veikimas esant dideliam apkrovimui.

Saugumo testavimo metu buvo patikrintas tapatybės nustatymo ir prieigos teisių užtikrinimo teisingumas. Buvo bandoma prisijungti įvedus neteisingus prisijungimo duomenis, taip pat buvo bandoma pasiekti informaciją, kuri yra neprieinama pagal turimas teises. Saugumo testavimas parodė, kad sistema yra pakankamai saugi.

Buvo atlikti sistemos našumo testai, jie detaliau aprašyti eksperimentinėje dalyje. Našumo testai parodė, kad pasirinktos technologijos ir komunikacijos protokolai yra tinkami.

Taip pat buvo simuliuojamas sistemos veikimas esant dideliam apkrovimui. Testas plačiau aprašytas eksperimentinėje dalyje. Sistemos testas parodė, kad sistema tenkina apkrovimo reikalavimą, t.y. sistema tikrai sklandžiai veiks esant 20 vienu metu prisijungusių klientų.

3.4. Sistemos kokybės analizė

IVS kokybė įvertinama atsižvelgiant į šiuos faktorius:

- Glaustumas – Kuriant sistemą naudojamasi kodo generatoriais, komunikavimo karkasais ir kitais kalbų bei kūrimo aplinkų įrankiais, kurie leidžia rašyti kuo mažiau nuosavo kodo. Programos kodas taip pat sumažinamas naudojant nuosavas kodo bibliotekas.
- Pernešamumas – Sistema naudoja daug specifinių Microsoft technologijų (.NET, LINQ, MSSQL), todėl serverio pusė yra pririšta prie Microsoft Windows operacinės sistemos. Tuo tarpu klientinė programa sukurta Flex technologija, todėl ją galima paleisti iš beveik bet kurios operacinės sistemos ar naršyklės.
- Nuosaikumas – Sistemos programinis kodas parašytas naudojantis C# ir ActionScript kalbų kodo standartais, todėl komponento ribose programinis kodas yra parašytas nuosaikiai. Tačiau atskiri komponentai parašyti naudojantis skirtingais standartais dėl aukščiau paminėtų kalbų sintaksės skirtumų.
  Vartotojo sąsaja yra nuosaiki, nes naudojami standartiniai Flex karkaso komponentai.

- Palaikomumas – Kadangi sistema kuriama naudojantis programavimo ir architektūriniais šablonais, bei yra gerai dokumentuota, todėl ją turėtų būti lengva palaikyti ir įvesti naujus reikalavimus.
- Testuojamumas – Sistema yra suskaidyta į mažai priklausomus komponentus, kas palengvina sistemos testavimą. Sistemos architektūra yra ganėtinai paprasta, todėl komponentų integravimas ir testavimas bus nesudėtingas.
- Panaudojamumas – Sistema turi interaktyvią grafinę vartotojo sąsają (Flex), todėl ji turėtų būti patraukli vartotojui. Sistema turi daug interaktyvių pagalbinių funkcijų, kurios palengvina bei pagreitina sistemos vartotojo darbą.
- Patikimumas – Naudojamų technologijų patikimumas yra geras. Tačiau yra imtasi papildomų priemonių pagerinti sistemos veikimo stabilumą. Naudotos priemonės – ryšio kokybės valdymas, veikimas atsijungus nuo serverio, atsistatymas po klaidos.
- Našumas –Naudojamasi žinomomis geromis programavimo praktikomis, tam kad be reikalo nebūtų švaistomi atminties ir kompiuterio procesoriaus resursai. Tačiau didele dalimi remiamasi naudojamų technologijų našumu.
- Saugumas –Naudojamas tapatumo ir prieigos teisių nustatymas. Vartotojų slaptažodžiai yra laikomi užšifruoti. Kadangi sistema daugiausiai bus naudojama apsaugotame vietiniame tinkle, todėl nėra imtasi papildomų ypatingų priemonių saugumui užtikrinti.

3.5. Technologijos taikymo rezultatai ir išvados

Incidentų Valdymo Sistemos kūrimo metu buvo praktiškai išbandytos Flex technologijos galimybės. Projekte buvo įgyvendinta modulinė architektūra, navigacijos sistema, prisijungimo ir teisių valdymas, realaus laiko komunikacija žinučių ir incidentų objektų koordinačių perdavimui, kalbų palaikymas, duomenų saistymas. Šis taikymas parodė, kad Flex technologija yra tinkama kurti interaktyvias paskirstytas sistemas.

PureMVC mikroarchitektūros taikymas pasiteisino, nes leido sukurti vieningą programos ir jos modulių struktūrą, o tai labai pagerina sistemos palaikomumą ir išplečiamumą.

4. Eksperimentinis tyrimas

4.1. Tikslas

Eksperimento tikslas - įvertinti RIA technologijos efektyvumą ir jos panaudojimo kuriant interaktyvias paskirstytas sistemas tikslingumą.

4.2. Tyrimo metodika

Tyrimas susideda iš trijų dalių:

1. Duomenų perdavimo formatų apimties ir greitaveikos tyrimas;
2. Apkrovimo tyrimas;
3. Modulinės architektūros tyrimas.

Pirmiems dviem tyrimams buvo sukurtos specialios programos, kurios formuoja specialias testines užklausas ir siunčia į serverį. Trečio tyrimo metu buvo analizuojami sukompiliuotų programos failų dydžiai.

4.3. Priemonės

Eksperimentų metu buvo naudojamos tokios priemonės:

- Serveris:
  - Toshiba Satellite
  - Operacinė sistema: Windows XP
  - Procesorius: Intel T2250 1,73GHz x2.
  - Atmintis: 3GB
- Kliento kompiuteriai:
  - Toshiba Satellite
  - Operacinė sistema: Windows XP
  - Procesorius: Intel T2250 1,4 GHz x2
  - Atmintis: 512MB

4.4. Duomenų perdavimo formatų apimties ir greitaveikos tyrimas

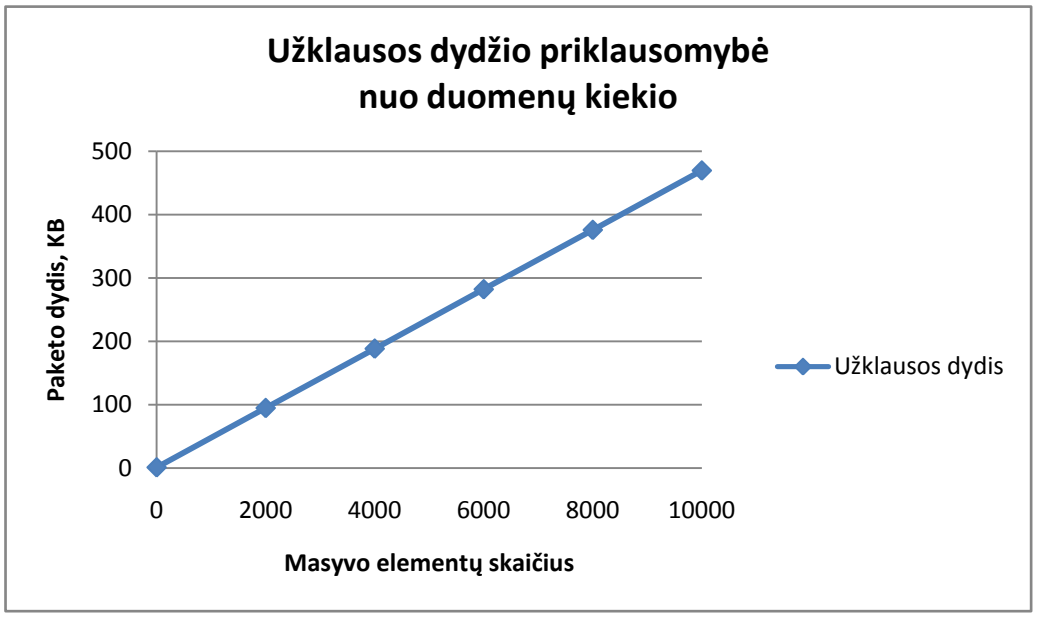
Šio eksperimento metu buvo tiriama AMF formato greitaveika ir užklausų apimtis. Buvo padaryta programa, kuri suformuoja pasirinktų objektų masyvą ir siunčia į serverį, bei gauna

„apverstą“ tokio paties dydžio masyvą atgal. Pilnas ryšio su serveriu ciklas apima šiuos veiksmus:

1. Užklausos vertimas į nuosekliąją formą kliento pusėje;
2. Žinutės siuntimas į serverį;
3. Užklausos atstatymas į objektinę formą serveryje;
4. Užklausos įvykdymas serverio pusėje;
5. Atsakymo vertimas į nuosekliąją formą serverio pusėje;
6. Žinutės parsiuntimas į klientą;
7. Atsakymo atstatymas į objektinę formą kliente.

Šio pilno ryšio ciklo metodika bus remiamasi atliekant eksperimentus.

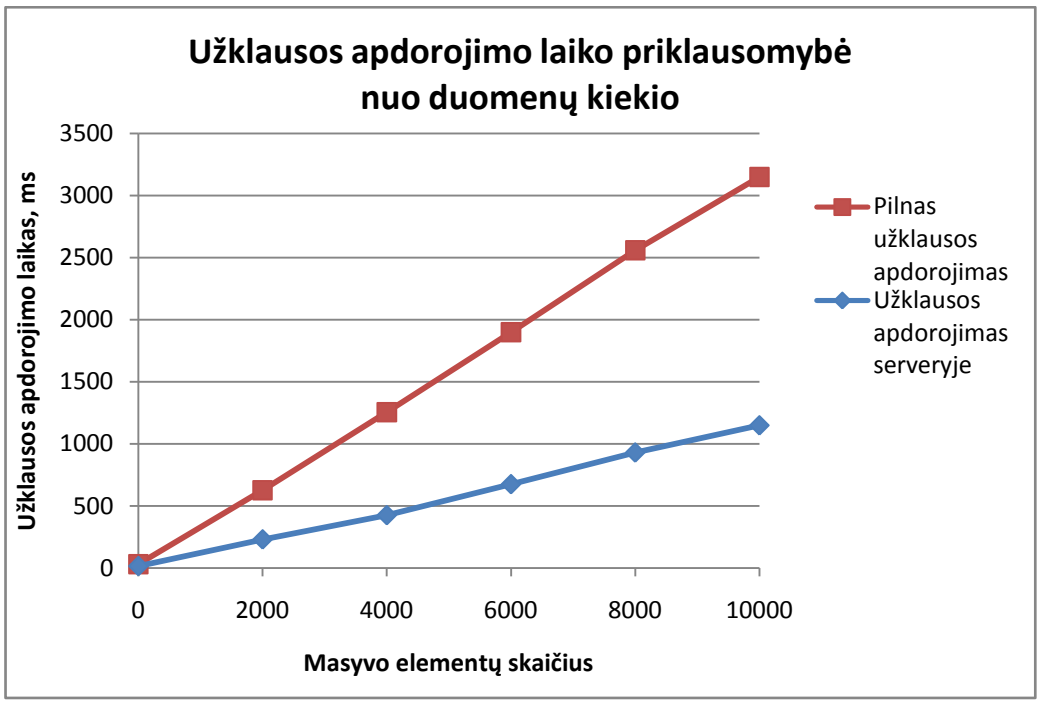
Testui atlikti buvo naudojama įterptinė programinė įranga, skirta analizuoti naršyklės užklausoms,– ServiceCapture2. Visų pirma buvo ištirta užklausos dydžio priklausomybė nuo siunčiamų duomenų kiekio (15 paveikslas). Eksperimento metu, kaip ir buvo tikėtasi, buvo nustatyta, kad užklausos dydis tiesiškai priklauso nuo masyvo elementų skaičiaus.



Pav. 15 Užklausos dydžio priklausomybė nuo duomenų kiekio

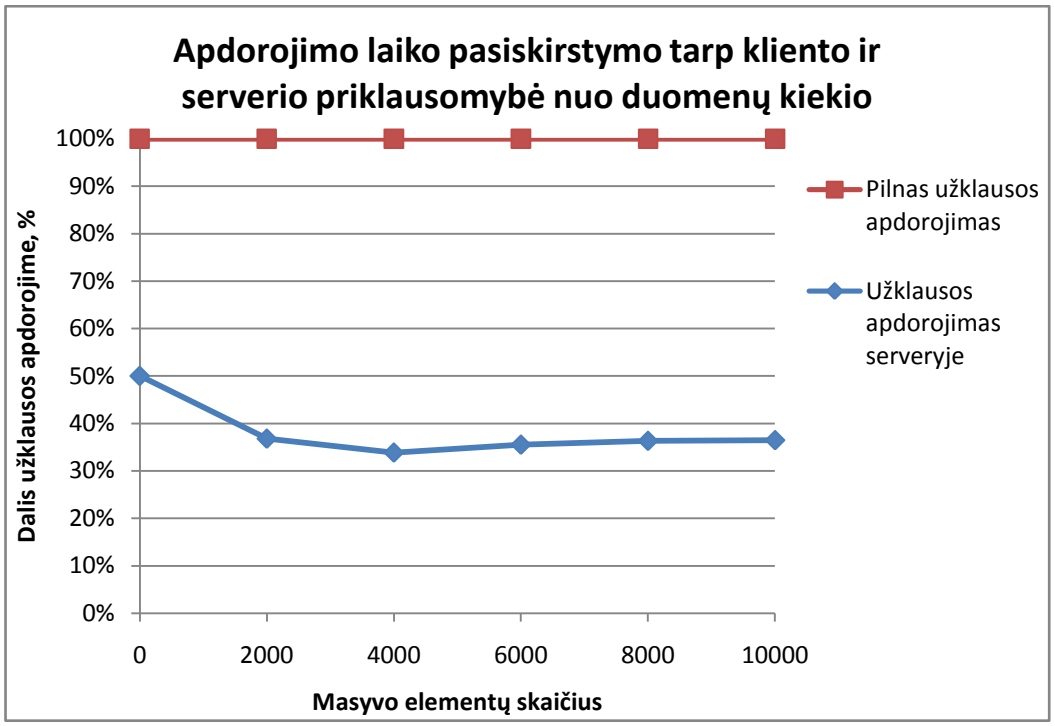
Toliau buvo nagrinėtas užklausos apdorojimo laikas. Su ServiceCapture2 programa buvo gautas iš naršyklės išeinančios užklausos apdorojimo laikas. Tai atitinka 2-6 punktus iš aukščiau paminėto pilno užklausos ciklo. Taip pat buvo gautas pilnas užklausos apdorojimo laikas (1-7

punktai). Šie laikai pavaizduoti 16 paveiksle. Kaip matome užklausos apdorojimo laikas taip pat yra tiesiškai priklausantis nuo masyvo elementų skaičiaus. Kadangi serveryje vykdytos operacijos sudėtingumas buvo tiesinis, tai užklausos apdorojimas serveryje taip pat turi tiesinę priklausomybę. Gautas didžiausias apdorojimo laikas – 3 sekundės yra gana didelis, bet neviršija leistinos vėlinimo ribos – 5 sekundžių. Kita vertus, eksperimento metu siunčiamos užklausos buvo neįprastai didelės, paprastai programos yra optimizuojamos, kad vienu metu nesiųstų daugiau nei 100 elementų.



Pav. 16 Užklausos apdorojimo laiko priklausomybė nuo duomenų kiekio

Šie duomenys toliau vaizduojami kitu aspektu – pagal tai kokią dalį jie sudaro nuo bendros užklausos. Šie duomenys pavaizduoti 17 paveiksle. Iš paveikslo matome, kad užklausos apdorojimas serveryje sudaro truputį mažesnę dalį (apie 35% nuo visos užklausos apdorojimo), negu užklausos apdorojimas kliente. Procentinės dalies dydis yra beveik pastovus, t.y. abi dalys kinta tiesiškai, kintant masyvo elementų skaičiui. Esant mažam (iki 2000) elementų skaičiui serverio dalis yra šiek tiek didesnė. Tai paaiškinama didesne perdavimo tinklo įtaka matavimų rezultatams.



Pav. 17 Apdorojimo laiko pasiskirstymo tarp kliento ir serverio priklausomybė nuo duomenų kiekio

Iš eksperimentų rezultatų matome, kad AMF duomenų formatas yra gana efektyvus. Koduojant žinutes šiuo protokolu prireikė tik 50 baitų vienam įrašui. Žinutės apdorojimo laikas taip pat yra geras – 35 milisekundės šimtui įrašų. Kaip ir buvo galima tikėtis, buvo nustatyta, kad serverio pusėje žinutės kodavimas užtrunka beveik 2 kartus greičiau, nei kliente. Iš to galima daryti išvadą, kad programos NET aplinkoje veikia beveik dvigubai greičiau, nei Flash Player aplinkoje.

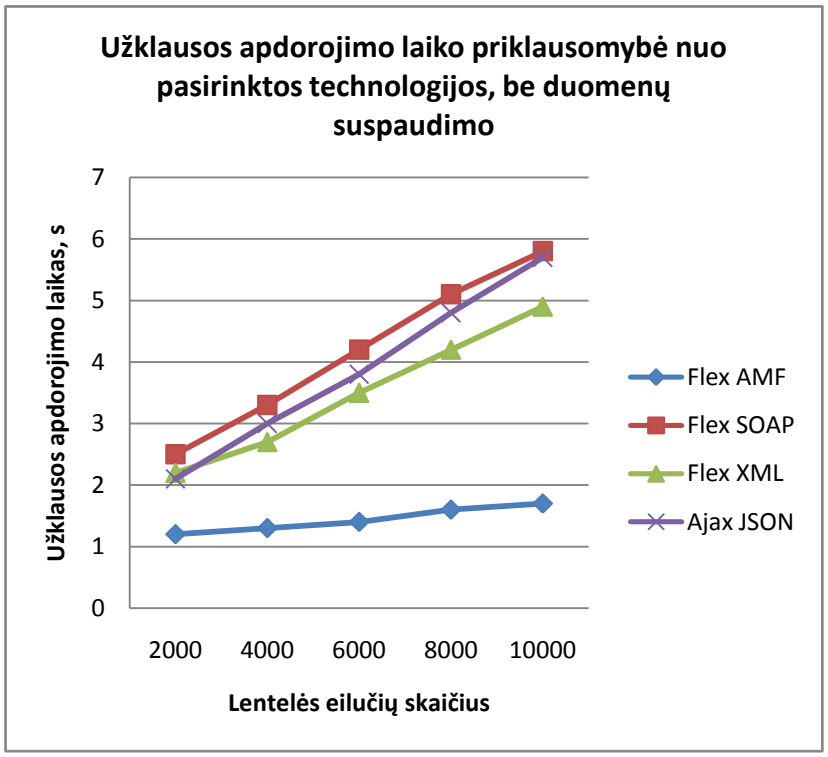
Vykdytas greitaveikos eksperimentas neparodė AMF veikimo santykinio našumo palyginus su kitais perdavimo formatais. Dėl laiko stokos nebuvo suspėta įgyvendinti visų galimų žinučių formatų testų programų. Naudojamas FluorineFX komponentas žinutes koduoja tik AMF formatu, todėl būtų reikėję didelių pastangų perprojektuoti programą. Todėl buvo pasinaudota trečios šalies viešai internete skelbiamu duomenų apdorojimo greitaveikos palyginimo įrankiu – „Census – RIA Data Loading Benchmarks“[8].

Eksperimento metu iš serverio gaunama kintamo dydžio duomenų lentelė, kuri paskui atvaizduojama pasirinkta technologija. Buvo atliktas eksperimentas su šiais duomenų formatais:

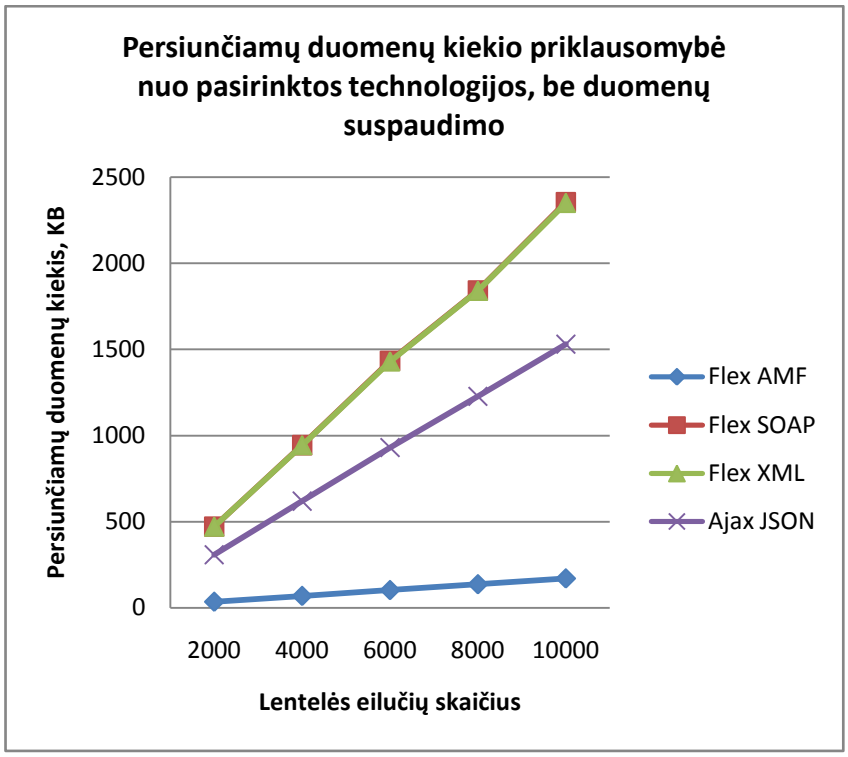
[8] http://www.jamesward.com/census/

1. Flex AMF. Duomenys perduodami AMF formatu, ir vaizduojami Flex technologija.
2. Flex SOAP. Duomenys perduodami SOAP protokolu ir vaizduojami Flex technologija.
3. Flex XML. Duomenys perduodami XML formatu ir vaizduojami Flex technologija.
4. Ajax JSON. Palyginimui buvo pasirinkta viena ne Flex technologija. Prieš darant eksperimentą buvo ištirta, kad koduojant Ajax užklausas JSON protokolu, gaunama didžiausia sparta ir mažiausias perduodamų duomenų kiekis tarp visų Ajax duomenų perdavimo formatų.

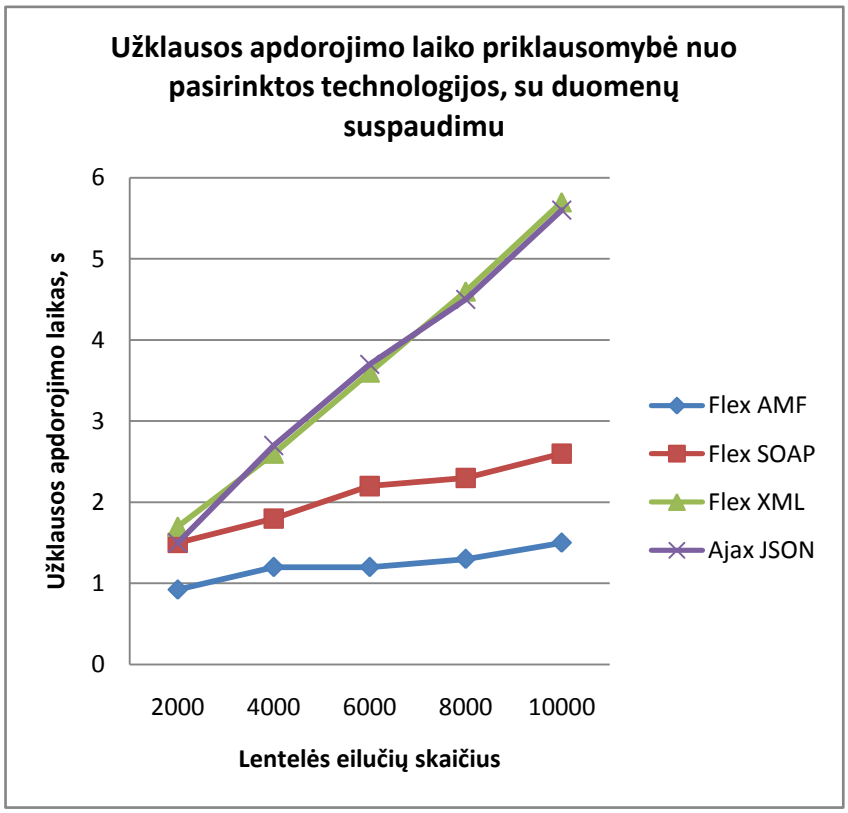
Eksperimentas buvo atliktas su skirtingais lentelės dydžiais, taip pat įjungus duomenų suspaudimą (GZip archyvavimas), arba ne. Buvo matuojamas užklausos apdorojimo laikas ir perduodamų duomenų kiekis. 18, 19, 20 ir 21 grafikuose pavaizduotos atitinkamai užklausos apdorojimo laiko ir persiunčiamų duomenų kiekio priklausomybės nuo pasirinktos technologijos, nenaudojant persiunčiamų suspaudimo, arba naudojant jį.



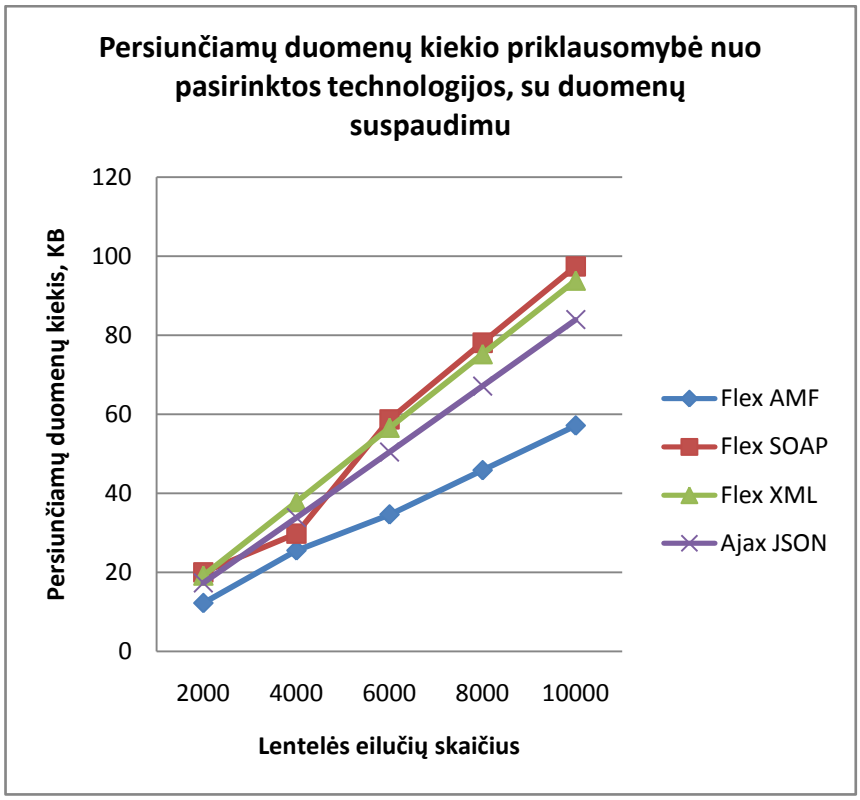
Pav. 18 Užklausos apdorojimo laiko priklausomybė nuo pasirinktos technologijos, be duomenų suspaudimo



Pav. 19 Persiunčiamų duomenų kiekio priklausomybė nuo pasirinktos technologijos, be duomenų suspaudimo



Pav. 20 Užklausos apdorojimo laiko priklausomybė nuo pasirinktos technologijos, su duomenų suspaudimu



Pav. 21 Persiunčiamų duomenų kiekio priklausomybė nuo pasirinktos technologijos, su duomenų suspaudimu

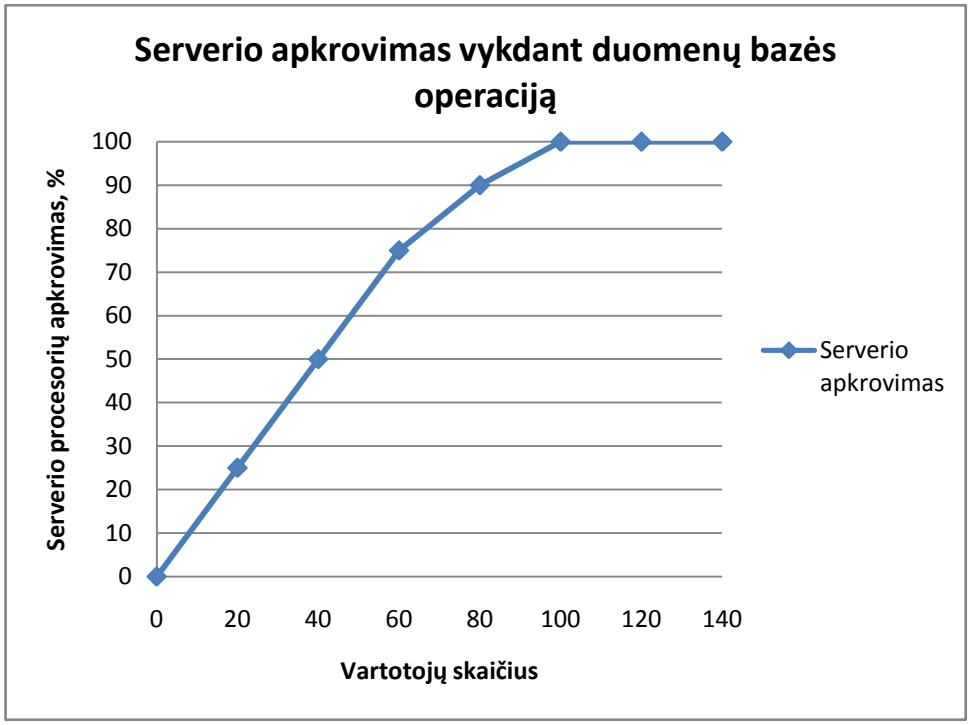
Iš grafikų matome, kad naudojant Flex technologiją ir AMF formatą, užklausos buvo įvykdomos greičiausiai ir persiunčiamų duomenų kiekis buvo mažiausias. Kai duomenų suspaudimas nebuvo naudojamas, tai SOAP, XML ir JSON formatų atveju padidinus lentelės eilučių skaičių būdavo persiunčiamas didelis duomenų kiekis (iki 2.5MB). Todėl bendrą apdorojimo laiką labiausiai įtakodavo duomenų persiuntimo tinklu laikas. Įjungus duomenų suspaudimą, perduodamų duomenų kiekis sumažėjo drąstiškai (iki 20 kartų). Tačiau XML ir JSON formatų atveju labai padidėjo duomenų suspaudimo trukmė serveryje (20 paveikslas). Iš grafikų taip pat matome, kad Ajax technologija ir JSON formatas pagal nagrinėtus parametrus mažai nusileidžia Flex technologijai, naudojant SOAP ir XML formatus. Iš eksperimento taip pat matome, kad duomenų suspaudimo įtaka Flex AMF atveju yra maža, ir galima jo nenaudoti.

Eksperimentas buvo atliekamas siunčiant didelius duomenų kiekius. Tačiau tokie dideli duomenų kiekiai realiose sistemose dažniausiai nėra siunčiami. Arba, jei reikia vaizduoti dideles duomenų lenteles, tai įgyvendinamas serverio pusės duomenų puslapiavimas ir informacijos išsaugojimas spartinančiojoje atmintinėje. Vis dėlto, sistemose, kuriose reikia didelio našumo,

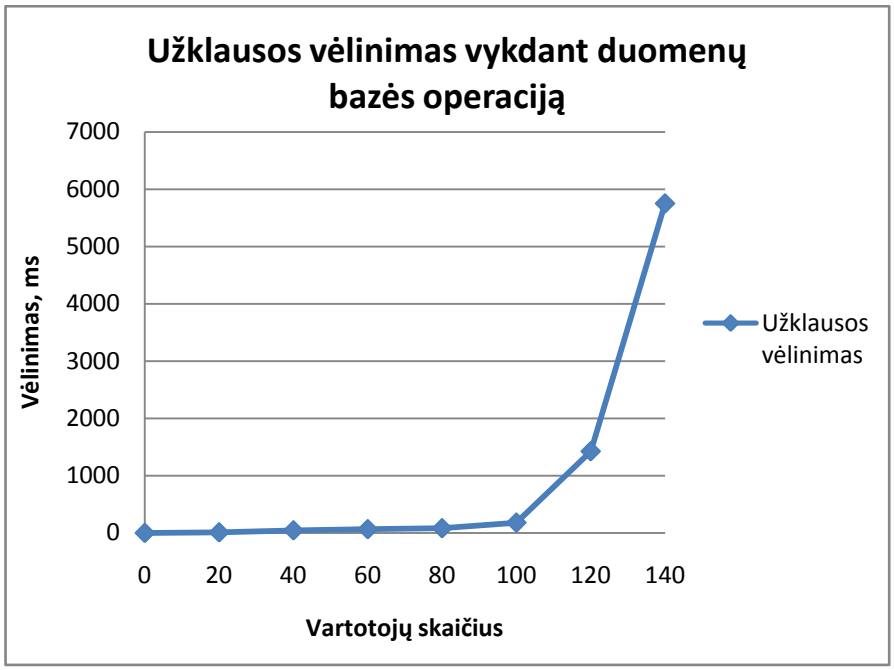
geriausia naudoti tam pritaikytus formatus (pvz. AMF). Ir atitinkamai, kai reikia didesnio suderinamumo, galima naudoti SOAP ar kitus standartizuotus formatus.

4.5. Apkrovimo tyrimas

Šio eksperimento metu buvo tiriamas sistemos apkrovimas vienu metu prisijungus dideliam klientų kiekiui. Buvo sukurta speciali programa, kuri labai dažnai siunčia užklausas į serverį, taip imituodama daugelio klientų darbą. Ši programa buvo paleista keliuose kompiuteriuose, kad eksperimentas būtų artimesnis realybei. Buvo laikoma, kad vieno vartotojo prisijungimas yra tolygus 5 užklausoms per sekundę. Užklausos buvo vykdomos RTMP protokolu, AMF formato žinutėmis. Žinutės buvo nedidelės ir talpino vieną incidento objektą. Toks žinučių pobūdis buvo pasirinktas dėl to, kad labiau atitiktų realių sistemoje naudojamų žinučių charakteristikas. Serveryje buvo kviečiamos dviejų tipų komandos: į pirmą komandą įėjo duomenų saugojimas duomenų bazėje, į antrą – ciklas su matematiniais skaičiavimais, be kreipinių į išorinę atmintį. Buvo matuotas serverio procesorių apkrovimas ir užklausos atsakymo vėlavimas. 22 paveiksle pavaizduotas serverio procesorių apkrovimas, priklausomai nuo prisijungusių vartotojų skaičiaus, vykdant duomenų bazės operaciją. 23 paveiksle pavaizduotas to paties eksperimento užklausos vėlinimas.



Pav. 22 Serverio apkrovimas priklausomai nuo vartotojų skaičiaus



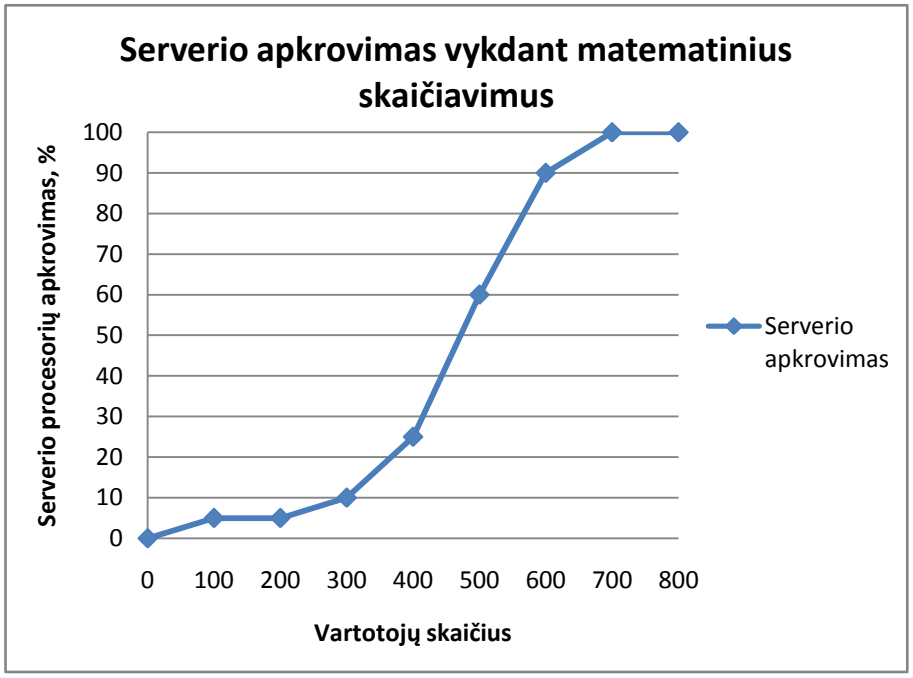
Pav. 23 Užklausos vėlinimas vykdant duomenų bazės operaciją

Iš grafikų matosi, kad vienas serveris gali aptarnauti iki 140 vienu metu prisijungusių vartotojų (serveris turi 2 procesorius, todėl vienam procesoriui tenka 70 vartotojų). Dar padidinus vartotojų skaičių, užklausos pradeda kauptis ir užklausos vėlinimas staigiai padidėja, bei pradeda viršyti leistiną 5 sekundžių ribą. Eksperimento metu siųstos žinutės buvo mažos (kaip ir numatomos realiu atveju) ir jų apdorojimo laikas praktiškai neturėjo įtakos eksperimento rezultatams.

Toliau buvo atliktas tas pats eksperimentas, tik vietoje duomenų bazės operacijos serveryje buvo vykdomi matematiniai skaičiavimai. Rezultatų grafikas pavaizduotas 24 paveiksle. Eksperimento metu buvo pastebėta, kad parinkus pirmo eksperimento vartotojų skaičiaus skalę, serverio apkrovimas buvo labai mažas, todėl buvo žymiai padidintas prisijungusių vartotojų skaičius. Galiausiai buvo nustatyta, kad nevykdant kreipinių į duomenų bazę ar kitas išorines laikmenas, galima pasiekti iki 800 vartotojų skaičių vienam serveriui. Tai daugiau nei 5 kartus daugiau, nei vykdant tokias operacijas. Vykdant šį eksperimentą žinučių vėlinimas visada buvo mažas, todėl jo grafikas nepateikiamas.

Iš eksperimentų matyti, kad pasirinkta technologija tenkina IVS sistemai apibrėžtus vartotojų skaičiaus reikalavimus (20 vienu metu prisijungusių vartotojų). Nustatyta, kad vienu metu prisijungusių vartotojų skaičių pirmiausia riboja serverio procesorių apkrovimas. Taip pat

nustatyta, jog serverio našumą galima padidinti, jei serverio pusėje informacija laikinai būtų saugojama spartinančiojoje atmintinėje.



Pav. 24 Serverio apkrovimas vykdant matematinius skaičiavimus

4.6. Modulinės architektūros tyrimas.

IVS sistemos sudaro Flex dalį sudaro pagrindinės programos kodas ir 3 moduliai. Sukompiliavus gaunami keturi SWF failai. Prieš išleidžiant programą naudojimui visada reikia atlikti dvi optimizacijas:

1. Failus reikia sukompiliuoti išleidimo (angl. release) režimu, nes tada iš kodo išimami daug kainuojantys derinimo sakiniai. IVS projekto atveju taip sukompiliavus projektą failų dydžiai sumažėjo 3 kartus.
2. Nuo Flex 3 versijos įvesta galimybė iškelti standartines karkaso bibliotekas į atskirą failą (dydis ~550KB), kuris yra bendras visoms Flex programoms. Jis yra parsiunčiamas vieną kartą ir išsaugojamas Flash Player įskiepyje. Įvykdžius šią optimizaciją IVS projekto atveju programos failas sumažėjo apie 300 KB.

Kompiliuojant failus, galima pasirinkti optimizuoti modulių failus pagal pagrindinį failą. Taip modulių failuose nekartojamas klasių kodas, kuris yra pagrindiniame programos faile. Failų dydžių priklausomybė tuo to ar jie buvo optimizuoti, ar ne, pavaizduota 2 lentelėje. Iš jos

matome, kad suoptimizavus galima gauti apie 2 kartus mažesnius failus, nei neoptimizavus (jei skaičiuosime tik modulių failus, tai 3 kartus).

Lentelė 2 Modulių failų optimizacijos poveikis failų dydžiams

Failas	Failų dydžiai, kai jie yra optimizuojami pagal Cat.swf failą, KB	Failų dydžiai, kai optimizacija nevykdoma, KB
Cat.swf	257	257
ForceModule.swf	48	147
IncidentModule.swf	56	220
UserModule.swf	46	99
Bendras failų dydis	407	723

Suoptimizavus modulius, juos galima naudoti tik toje programoje, todėl jų negalima pakartotinai panaudoti. Todėl šiuo klausimu reikia ieškoti kompromiso tarp failų dydžio ir pakartotinio panaudojimo.

Jei programa neskaidoma į modulius, o sukompiliuojama į vieną failą, tai failo dydis yra 296KB. Taigi naudojant modulinę architektūrą bendras failų dydis gaunasi didesnis nei naudojant monolitinę architektūrą, ir netgi artimas pagal modulinę architektūrą sukompiliuotam pagrindinio programos failo dydžiui. Tačiau iš šio bandymo negalima daryti išvados, kad modulinė sistema bendru atveju nepasiteisina, nes sistemoje buvo tik keli nesudėtingi moduliai. Jei modulių būtų daugiau, ir jie būtų sudėtingesni, tai pagrindinio failo dydis taptų daug mažesnis, nei monolitinės sistemos failas. Taip pat projektuojant sistemą nebuvo siekta padaryti pradinį failą kuo mažesnį. Modulinė architektūra didelėse sistemose pasiteisina šiais aspektais: logikos išskaidymas, paslaugomis grįstos architektūros naudojimas, sistemą kuriančios komandos suskaidymas į mažesnes ir operatyvesnes grupeles.

4.7. Eksperimentų rezultatai ir išvados

Atlikus eksperimentus buvo įsitikinta Flex technologijos našumu.

Perdavimo formatų greitaveikos tyrimo metu buvo nustatyta, jog AMF žinučių formatas yra geriausiai informaciją užkoduojantis ir greičiausiai veikiantis duomenų formatas, palyginus

su XML, SOAP bei JSON formatais. Dėl savo greičio jis yra tinkamas naudoti realaus laiko duomenų perdavimui.

Serverio apkrovimo eksperimento metu buvo nustatyta, kad prie vieno serverio vienu metu gali būti prisijungę nuo 140 iki 800 vartotojų. Šis vartotojų skaičius tenkina IVS sistemai iškeltus apkrovimo reikalavimus. Serverio apkrovimas labai priklauso nuo to, kokios operacijos vykdomos apdorojant duomenis. Jei kiekviena užklausa iššaukia duomenų bazės operaciją, tai maksimalus vartotojų skaičius bus mažas, jei vykdomi skaičiavimai, kurie nereikalauja kreipinio į išorinę atmintį, tai maksimalus vartotojų skaičius drastiškai išauga. Todėl, projektuojant serverio paslaugas, operacijas reikia optimizuoti saugant laikinus duomenis spartinančiojoje atmintinėje. Jei sistemos vartotojų skaičius viršytų išmatuotas apkrovimo ribas, tai didesnio našumo būtų galima pasiekti pagerinant techninę serverio įrangą. Kitas našumo pagerinimo būdas yra išskaidyti incidentų įvykių aptarnavimą į atskirus serverius. Kiekvienas incidentas gali būti apdorojamas atskiro serverio nepriklausomai.

Modulinės architektūros tyrimo metu buvo išbandyti būdai sumažinti sukompiliuotų programos failų dydžius. Atlikus optimizacijas, gauti failai užėmė apie 6 kartus mažiau vietos nei neoptimizavus ir bendras dydis buvo apie 400 KB. Toks failų dydis nėra didelis netgi lyginant su įprastinėmis interneto programomis, ypač todėl, kad failai parsiunčiami tik vieną kartą visai darbo sesijai. Sistemą kuriant moduliais gaunami keli failai, kurie gali būti užkraunami dinamiškai. Modulių failai tampa labai maži, jei kompiliavimo metu jie būna optimizuojami (iš naujo nekompiliuojamos klasės, kurios jau būna sukompiliuotos pagrindiniame faile). Iš šio tyrimo galima daryti išvadą, kad modulinė architektūra smarkiai nepadidina failų dydžių. Kadangi modulinė architektūra leidžia sukurti lankstesnes ir didesnes programas, tai ji yra tinkama naudoti interaktyviose paskirstytose sistemose.

5. Rezultatai ir išvados

- Darbo metu buvo atlikta RIA technologijų analizė ir palygintos šiuo metu egzistuojančios populiariausios technologijos (Flex, Silverlight, JavaFX). Nustatyta, kad šiuo metu populiariausia ir labiausiai išvystyta RIA technologija yra Adobe Flex, todėl ji buvo pasirinkta detaliam nagrinėjimui. Nustatyta, kad renkantis RIA technologiją reikia atsižvelgti į klientų naudojamų įrenginių pobūdį (asmeniniai kompiuteriai ar mobilieji įrenginiai), taip pat į serveryje naudojamą technologiją.
- Nustatyti architektūriniai pokyčiai, į kuriuos reikia atsižvelgti pereinant nuo darbastalio ar interneto programų prie RIA programų projektavimo. Apžvelgtos naujos technologinės galimybės. Galimybės palygintos pagal populiariausias RIA technologijas.
- Architektūros tyrimo metu nustatyta, kad kuriant didelio masto RIA sistemas, sistemą naudinga skaidyti į atskirus modulius. Vidinei programos struktūrai pagerinti reikia naudoti standartizuotas mikroarchitektūras. Išnagrinėtos RIA populiariausios projektavimo metodologijos, ir mikroarchitektūrų karkasai.
- Pasirinktų sprendimų tyrimui buvo sukurta ir išanalizuota Incidentų Valdymo Sistema. Sprendimui įgyvendinti pasirinkta Flex technologija pasiteisino, nes leido įgyvendinti visus reikiamus komunikacijos protokolus ir sukurti interaktyvią sąsają.
- Atliktų greitaveikos ir apkrovimo eksperimentų metu nustatyta, kad Flex technologija yra labai naši ir tinka kurti Incidentų Valdymo Sistemą, bei kitas interaktyvias paskirstytas sistemas. Didelė skaičiavimo sparta (žr. 4.4 skyrių - Duomenų perdavimo formatų apimties ir greitaveikos tyrimas) ir pažangūs komunikavimo protokolai leidžia užtikrinti sklandų sistemos darbą.
- RIA technologijos po truputį keičia požiūrį į darbastalio ir interneto programas. Darbo metu buvo sukurta sistema, sėkmingai apjungianti šių dviejų programų tipų savybes – didelę greitaveiką ir interaktyvumą bei veikimą visose operacinėse aplinkose ir naršyklėse.

6. Literatūros sąrašas

1. ALLAIRE, J. Macromedia Flash MX—A next-generation rich client. March 2002. Prieiga per internetą: <http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf>
2. MORITZ, F. Rich Internet Applications (RIA): A Convergence of User Interface Paradigms of Web and Desktop - Exemplified by JavaFX: magistro darbas. University of Applied Science [Kaiserslautern, Vokietija], 2008.
3. PRECIADO, J.C. et al. University de Extremadur. Necessity of methodologies to model Rich Internet Applications. 7-tasis IEEE tarptautinis simpoziumas. 2005.
4. PRECIADO, J.C. et al. Univ. de Extremadur. Designing Rich Internet Applications with Web Engineering Methodologies. 9-tasis IEEE tarptautinis seminaras. 2007.
5. PRECIADO, J.C. et al. Univ. de Extremadur. Designing Rich Internet Applications Combining UWE and RUX-Method. 8-toji tarptautinė interneto inžinerijos konferencija. 2008.
6. Hoon KIM, Young-Jun JEON; Seung-Ho SHIN. Platform of Rich Internet Application for Wireless Sensor Network. Tarptautinė informatikos ir programų inžinerijos konferencija. 2008.
7. FARRELL, J.; NEZLEK, G.S. Rich Internet Applications The Next Stage of Application Development. Tarptautinė ITI konferencija. 2007.
8. MEIER, J.D. et al. Rich Internet Application Architecture Guide, Paterns and Practices. Microsoft. 2008.
9. Adobe Systems Inc. Action Message Format - AMF 3. [interaktyvus]. Prieiga per internetą: <http://download.macromedia.com/pub/labs/amf/amf3_spec_121207.pdf>
10. WEBSTER, S., TANNER, L. Developing Flex RIAs with Cairngorm microarchitecture. Adobe.com [interaktyvus] 2008 gegužė [žiūrėta 2009-05-16]. Prieiga per internetą: <http://www.adobe.com/devnet/flex/articles/cairngorm_pt1.html>
11. HALL, C. PureMVC framework overview with UML diagrams. PureMVC [interaktyvus]. 2008 gegužė [žiūrėta 2009-05-16]. Prieiga per internetą: <http://puremvc.org/component/option,com_wrapper/Itemid,35/>
12. BLOM, S. et al. Write Once, Run Anywhere A Survey of Mobile Runtime Environments. Tarptautinė GPC konferencija. 2008.

13. Rich Internet Application (RIA) Market Share / Global Usage. StatOWL [interaktyvus]. 2009 gegužė [žiūrėta 2009-05-16]. Prieiga per internetą: <http://www.statowl.com/custom_ria_market_penetration.php?1=1&timeframe=last_6&interval=month&chart_id=13&fltr_br=&fltr_os=&fltr_se=&fltr_cn=&chart_id=16>
14. JavaFX Tools | JavaFX 1.1 Production Suite Release Notes | JavaFX. java.sun.com [interaktyvus]. 2009 vasaris [žiūrėta 2009-05-16]. Prieiga per internetą: <http://java.sun.com/javafx/1/reference/releasenotes/production-suite-release-notes-1-1.html>
15. Silverlight 3 beta – list of new features. UX Passion [interaktyvus]. 2009 kovas [žiūrėta 2009-05-16]. Prieiga per internetą: <http://www.uxpassion.com/2009/03/silverlight-3-beta-list-of-new-features/>
16. Adobe Flex 3.3 Language Reference. Livedocs.adobe.com [interaktyvus]. 2009 vasaris [žiūrėta 2009-04-19]. Prieiga per internetą: <http://livedocs.adobe.com/flex/3/langref/index.html>
17. Adobe Flex 3 Developer Guide. Adobe. 2008
18. STAMOS, A. et al. RIA Security Workshop: Blurring the Line between Web and Desktop Security. Slideshare[interaktyvus]. 2008 spalis [žiūrėta 2009-05-04]. Prieiga per internetą: <http://www.slideshare.net/astamos/ria-and-ajax-security-workshop-part-2-presentation>
19. Adobe Flash Player 9 Security. White Paper. Adobe [interaktyvus]. 2008 liepa [žiūrėta 2009-04-26]. Prieiga per internetą: <http://www.adobe.com/devnet/flashplayer/articles/flash_player_9_security.pdf>

7. Santrumpų ir terminų žodynėlis

ActionScript – objektiškai orientuota scenarijų kalba, dažniausiai naudojama Flash platformoje.

ActiveX – komponentų, kurie gali būti paleidžiami nepriklausomai nuo jų programavimo kalbos, karkasas.

AJAX (Asynchronous JavaScript and XML) – Asinchroninio JavaScript ir XML technologija skirta kurti interaktyvias internetines programas.

AMF (Action Message Format) - yra dvejetainis formatas, sukurtas naudojantis SOAP formato principais, skirtas duomenų apsikeitimui tarp Flash kliento ir serverio

CSS (Cascading Style Sheets) – pakopinių stilių aprašymo kalba, skirta HTML kalba rašomų dokumentų stiliams aprašyti.

GPS (Global Positioning System) –globali palydovinė navigacijos sistema.

HTML (HyperText Markup Language) – Hipertekstų ženklinimo kalba, dažniausiai naudojama tinklalapiams kurti.

JavaScript - objektiškai orientuota scenarijų kalba, dažniausiai naudojama naršyklėse.

JSON (JavaScript Object Notation) – JavaScript objektinis žymėjimas.

LINQ (Language Integrated Query) – Į programavimo kalbą integruota užklausų kalba. Viena iš .NET technologijų.

MSSQL (Microsoft SQL Server) – reliacinės duomenų bazės kuriamas Microsoft kompanijos.

MXML – XML pagrindo vartotojo sąsajos aprašymo kalba, naudojama Flex technologijoje.

.NET – programinės įrangos platforma, skirta suvienodinti programų veikimo aplinkai Microsoft Windows operacinės sistemos viduje.

RIA (Rich Internet Application) (lietuviškai: Raiškiosios interneto programos) – tai interneto programų klasė, turinti darbalaukio programų charakteristikų, dažniausiai veikiančių naršyklės įskiepių ar nepriklausomų virtualių mašinų aplinkoje.

SVG (Scalable Vector Graphics) - dvimačių vektorinių vaizdų saugojimo formatas XML žymėjimo kalbos pagrindu.

TCP (Transmission Control Protocol) – vienas pagrindinių interneto protokolų.

UML (Unified Modeling Language) – yra standartizuota bendros paskirties modeliavimo kalba, naudojama programų inžinerijoje.

WPF (Windows Presentation Foundation) – grafinė .NET karkaso posistemė, skirta turtingų vartotojų sąsajų kūrimui.

XAML (Extensible Application Markup Language) – deklaratyvi XML pagrindo žymėjimo kalba, sukurta Microsoft kompanijos.

XML (Extensible Markup Language) - universali dokumentų ženklinimo kalba, skirta dokumentų struktūrai aprašyti.

Summary

Rich Internet Application technology and its usage in interactive distributed systems

Internet technologies are being developed ever faster, and new ways for representing information are emerging. One such technology is Rich Internet Applications (RIA). This technology allows better and more responsive UI‘s, overwhelming graphic support, new computational, communication and interoperability capabilities. This technology is getting more and more support from key technology developers.

In this work you will be presented with RIA technologies and their usage. New ways for working with data, business and presentation layers will be presented. RIA technology characteristics will be thoroughly analyzed and compared with desktop and traditional web applications. An example interactive distributed system will be presented to show possible technology choices and solutions. Finally performance and stress tests will be conducted to show RIA technology feasibility.