KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Paulius Mačiulis

Programinės įrangos palaikymo sistema. Objektiškai orientuoto duomenų modelio tyrimas

Magistro darbas

Darbo vadovas
Doc. Dr. V. Pilkauskas

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Paulius Mačiulis

Programinės įrangos palaikymo sistema. Objektiškai orientuoto duomenų modelio tyrimas

Magistro darbas

Recenzentas
Prof. R. Butleris
2007-05-25

Vadovas
Doc. Dr. V. Pilkauskas
2007-05-24

Atliko
IFM-1/2 gr. stud.
Paulius Mačiulis
2007-05-23

Kaunas, 2007

Turinys

Paveikslai

Lentelės

SUMMARY

Software projects must meet development time constrains and continuously changing requirements. Software developers work under pressure to meet task deadlines and there is often no time left for exhausting software testing. Errors happen. It's very important to find them in proper time, collect and make bug reports. Also corrected errors and bugs can't be forgotten.

After shipping beta and final versions of the software, developers miss feedback from the users. To accumulate information about software errors can be a problematic task.

Work purpose is to create universal bug reporting system, adapted to software developers and end users. This system is accessible via local network and internet. Application developed using modern OOP technologies and .NET 2.0 Framework.

Finally, the object oriented data model research was accomplished. This data model was compared with standard data models. A performance and features of data models was tested. The critical conclusions were accepted.

1 ĮVADAS

Programinės įrangos inžinieriai turi užtikrinti, kad kuriama programinė įranga pasiektų atitinkamą kokybės lygį. **Programinės įrangos patvirtinimas** (angl. *verification*) yra procesas, kurio metu užtikrinama, kad PĮ atitinka savo reikalavimų specifikaciją. **Programinės įrangos testavimas**, tai procesas, kuris įvertina PĮ funkcionalumą ir teisingumą analizuojant ir vykdant ją. Tai vienas iš būdų verifikuoti programinę įrangą [1].

Testavimo metu siekiama įvertinti, ar programinė įranga atitinka apibrėžtus kokybės kriterijus ir nustatyti problemas (defektus), dėl kurių sistema negali atitikti vartotojo lūkesčių. Norint, kad atrastos problemos būtų pašalintos, apie jas visų pirma turi būti pranešta. Iš kitos pusės testavimas yra susijęs su rizikos valdymu, o valdyti rizikos neturint duomenų negalima. Todėl kyla poreikis atrastus defektus ne tik susieti, bet ir juos valdyti. Defektų valdymui visų pirma reikalingas jų dokumentavimas. Tokiu būdu defektų susiejimas ir valdymas tampa būtina testavimo proceso dalimi, o defekto ataskaita – techninė dokumentacija, aprašančia defekto simptomus, kai sistema elgiasi ne taip, kaip turėtų [3].

Deja, šiuolaikinę programinę įrangą neįmanoma ištestuoti visiškai [2]. Priežastys išvardintos pirmoje lentelėje [2].

Lentelė Nr. 1. Pilno programinės įrangos ištestavimo nepraktiškumo priežastys

Priežastis	**Komentaras**
PĮ priima įvairius įėjimus	Įvairūs įėjimai, t.y., sveikieji, realieji skaičiai, simbolių eilutės ir taip toliau, sukuria tokią didelę galimų įėjimų aibę, kad praktiškai negalima visų jų patikrinti testavimo metu.
PĮ įėjimai gali būti epizodiniai	Eilės tvarka, kuria įėjimai gali būti paduoti programinei įrangai, bendru atveju gali būti pertvarkyta į begalinį kombinacijų kiekį. Tokiu būdu, įėjimų sekų kiekis, kuris gali būti sudarytas naudojantis programine įranga, begalinis.
PĮ gali priimti ir sąveikauti su daugelypiais įėjimais	Neretai programinė įranga pritaikyta priimti įėjimus iš keleto šaltinių vienu metu. Tačiau testuotojai susiduria su sunkiai įveikiama užduotimi tinkamai pasirinkti įėjimų poaibį.

Taip pat dera pažymėti, kad realiam pasaulyje sukurti programinę įrangą dažnai skiriama per mažai laiko, programuotojai priversti skubėti. Dažnai nelieka laiko išsamiems programos testams atlikti, o produktas jau atiduodamas klientui. Programinę įrangą įsigiję vartotojai nori, kad rasti defektai būtų kuo greičiau pašalinti. Programinės įrangos kūrėjai suinteresuoti turėti patikimą grįžtamąjį ryšį su vartotojais ir kaupti defektų istoriją, kad vėliau iš sukauptų duomenų galėtų atlikti statistinius tyrimus.

2 ANALITINĖ DALIS

2.1 Tikslas

Programų sistemos pavadinimas – „Programinės Įrangos Palaikymo Sistema“ (toliau PĮPS). Siūloma sistema naudinga tiek programinės įrangos kūrėjams, tiek jos pirkėjams. Įsigydami programinį produktą (bendrąja prasme), klientai kartu gauna prieigą prie įsigyto produkto palaikymo sistemos – taip suteikiama galimybė stebėti ir įtakoti programinės įrangos palaikymo gyvavimo ciklą, bei sukuriamas patikimas grįžtamasis ryšys tarp PĮ vystytųjų ir vartotojų.

Projekto tikslas – realizuoti universalią programinės įrangos palaikymo sistemą (angl. *software support system*), pritaikytą tiek programinės įrangos kūrėjams, tiek galutiniams programinės įrangos pirkėjams. Panaudoti naujausias projektavimo ir programavimo technologijas, taip užtikrinant efektyvų tolimesnį sistemos palaikymą. Išanalizuoti ir panaudoti objektiškai orientuotą duomenų modelį.

Programų sistemos užsakovas yra UAB „Individualūs sprendimai“. Priede A pateiktas programinės įrangos diegimo pas užsakovą aktas.

Projekto vadovas KTU Verslo informatikos katedros docentas Vytautas Pilkauskas.

2.2 Informacija apie užsakovo organizaciją

UAB „Individualūs sprendimai“ (prieiga internetu http://www.it-partner1.com), įkurta 2002 metų rugpjūčio mėnesį Kaune, susibūrus ilgametę patirtį turintiems specialistams. Dirba programinės įrangos kūrimo, elektroninių įrenginių, prietaisų ir jų sistemų projektavimo, gamybos srityse.

Įmonė užsibrėžtų tikslų siekia daug investuodama į darbuotojų kvalifikacijos kėlimą, žingsnis į žingsnį eidama su naujausiomis technologijomis, užmegzdama tvirtus bendradarbiavimo ryšius su didžiosiomis kompanijomis.

2.3 Apžvalga

Dažniausiai perkant programinę įrangą, gaunamas ir tolimesnis jos palaikymas, priežiūra, tačiau labai dažnai bendravimas tarp programinės įrangos kūrėjų ir vartotojo apsiriboja elektroniniu paštu. Tai bendru atveju nėra pats tinkamiausias būdas pranešinėti apie rastus defektus ar nesklandumus programinėje įrangoje. Reikia įrankio, kuris leistų realiu laiku apsikeisti informacija tarp programinę įrangą įsigijusių vartotojų ir jos kūrėjų.

Siūloma programų sistema naudinga ir programinės įrangos kūrėjams ir jos pirkėjams. Įsigydami programinę įrangą vartotojai kartu gauna prieigą prie jos palaikymo sistemos – taip gaunama galimybė stebėti ir įtakoti programinio produkto palaikymo etapą. Sistemos populiarumą galutinių vartotojų tarpe užtikrins tai, kad ji bus prieinama internetu (jam nieko

nereikės instaliuoti). Galutinis vartotojas, susidūręs su defektu ar trūkumu sistemoje, internetu jungsis prie PĮPS (prisijungimo vardas ir slaptažodis bus sukuriami perkant programinę įrangą). Čia jis galės užpildyti klaidos radimo formą, peržiūrėti prieš tai registruotų klaidų būsenas, parsisiųsti naujausius atnaujinimus.

Programinės įrangos kūrėjams ši sistema pasitarnaus, nes jiems nereikės kiekvienam naujam programiniam produktui kurti atskiros programinės įrangos palaikymo sistemos, jie greitai ir struktūrizuotai gaus pranešimus apie rastus defektus.

2.4 Problemos

Pagrindine įgyvendinimo problema galima įvardinti didelę projekto apimtį, dar nevisiškai įsisavintas technologijas. Taip pat projekto apimtys gan didelės, lyginant su laiko resursais.

Svarbiausios problemos: pasirinkti tinkamą programavimo platformą, pasirinkti DBVS, apsispręsti dėl vieningos vartotojo sąsajos.

2.5 Galimi sprendimai

Panašios sistemos yra plačiai paplitusios, jas dažnai naudoja programinės įrangos kūrėjų komandos. Internete galima rasti tiek mokamų tiek ir nemokamų programų, kurios skirsis savo funkcionalumu, vartotojo sąsajos paprastumu, duomenų saugojimo ir apsikeitimo mechanizmu. Dažniausiai visos sistemos yra labiau orientuotos į programinės įrangos kūrėjus. Pritaikytos galutiniam vartotojui jos būna per sudėtingos ir tampa labai neefektyviomis.

1) Vienas iš stipriausių komercinių analogų yra Seapine Software kompanijos (prieiga internetu http://www.seapine.com/ttstudio.html) kuriama TestTrack produktų linija. Šios sistemos labiau orientuotos į didelių kompanijų poreikius. Didžiausiu pliusu galima įvardinti priedus programavimo aplinkoms, patogią grafinę sąsają. Minusas – kaina [15].

2) Sekanti sistema yra **BugLister** (prieiga internetu http://www.litwindow.com/BugLister/index.html)**.** Tai taip pat mokama programa, tačiau visiškai kito lygio nei prieš tai aptarta TestTrack. Demonstracinę versiją gali parsisiųsti kiekvienas norintis. Programa paprastai instaliuojama, vartotojo sąsaja panaši į elektroninio pašto kliento programos sąsają. Programa nenaudoja išorinės DBVS – įrašus saugo failuose su plėtiniu *delf*. Turi keletą naudingų funkcijų – įrašų filtravimas, klaidos istorijos peržiūra, išorinių failų, kaip papildomos informacijos, pridėjimas [12].

Didžiausi trūkumas šios sistemos yra tas, kad ji paremta žinučių apsikeitimo elektroniniu paštu principu – dėl to teršiasi pašto dėžutė, pranešimai apie klaidas gali vėluoti. Taip pat sunku įsivaizduoti, kaip tokia programa galėtų būti pritaikyta galutiniams

programinės įrangos vartotojams. Kitas trūkumas yra tas, kad programa nepalaiko ataskaitų kūrimo funkcijos, kas dažnai labai svarbu norint pateikti ataskaitas projektų vadovams.

3) Kita sistema, kuri naudojama klaidų sekimui ir registravimui, yra **PR-Tracker** (prieiga internetu: http://www.prtracker.com/). Tai mokama programa. Ji paprastai instaliuojama, nenaudoja išorinės DBVS. Turi keletą labai naudingų funkcijų – įvairių ataskaitų kūrimas, keleto projektų palaikymas. Programa pasižymi patogia vartotojo sąsaja [13].

Didžiausias šios programos trūkumas, kad ji veikia tik vietinio tinklo ribose (angl. *local network*). Taigi tuo pačiu negalimas programos pritaikymas ir galutiniams programinės įrangos vartotojams.

4) Labai plačiai naudojamas kuriamos sistemos analogas yra **Bugzilla** (prieiga internetu http://www.bugzilla.org/)**.** Tai jau visą eilę metų tobulinamas atviro kodo (angl. *open source*) projektas. Kaip ir daugelis tokio tipo projektų pasižymi sudėtingesniu instaliavimu. Naudoja tokias technologijas, kaip Perl, MySql, CGI. Vartotojo sąsaja paremta HTML, bendravimas su programa vyksta per interneto naršyklę. Palaiko tokias funkcijas kaip ataskaitų kūrimas ir spausdinimas, įrašų filtravimas, keleto produktų palaikymas, paieška pagal raktinį žodį [14].

Kaip didžiausią trūkumą galima įvardinti sudėtingą vartotoją sąsają. Programinės įrangos kūrėjams tai neturėtų sukelti didesnių problemų, tačiau patirtis rodo, kad norint **Bugzilla** pritaikyti galutiniams vartotojams tai tampa didžiausia problema.

Lentelė Nr. 2. Klaidų sekimo ir registravimo sistemų palyginimo lentelė

Sistemos pavadinimas	**Instaliavimas**	**Licencija**	**Išorinė DBVS**	**Vartotojo sąsaja**
TestTrack	Paprastas	Mokamas	MsSql, Oracle	Paprasta
BugLister	Paprastas	Mokama	Nėra, faile	Paprasta
PR-Lister	Labai paprastas	Mokama	Nėra, faile	Labai paprasta
Bugzilla	Sudėtingas	Atviras kodas	MySql	Sudėtinga

2.6 Bendri reikalavimai programų sistemai

2.6.1 Programų sistemos funkcijos

Galutinis programinės įrangos vartotojas

Vartotojo sprendžiami uždaviniai: siųsti pranešimus apie įsigytoje programinėje įrangoje rastus nesklandumus ir klaidas, stebėti registruotų klaidų būsenas ir naujausius atnaujinimus.

Patirtis dalykinėje srityje – įprastas darbuotojas.

Patirtis informacinėse technologijose – naujokas.

Papildomos vartotojo charakteristikos:

- kruopštumas.
- domėjimasis IT.
- greitas apsimokymo poreikis.
- bendri lietuvių arba anglų kalbos įgūdžiai.

Vartotojo prioritetas – didelis, jis naudosis sistema.

Funkcinis testuotojas

Vartotojo sprendžiami uždaviniai: gauti naujus pranešimus apie užregistruotas klaidas ir nesklandumus programinėje įrangoje. Nustatyti defekto prioritetą, priskirti jį taisyti programuotojui. Peržiūrėti programuotojų ištaisytas klaidas, patvirtinti ištaisytas klaidas.

Patirtis dalykinėje srityje – patyręs darbuotojas.

Patirtis informacinėse technologijose – informatikas.

Papildomos vartotojo charakteristikos:

- kruopštumas ir atsakingas požiūris į darbą.
- IT išmanymas.

Vartotojų prioritetai – didelis, jis naudosis sistema.

Programuotojas

Vartotojo sprendžiami uždaviniai: gauti pranešimus apie jam priskirtas taisyti klaidas, klaidų taisymas.

Patirtis dalykinėje srityje – patyręs darbuotojas.

Patirtis informacinėse technologijose – informatikas.

Papildomos vartotojo charakteristikos:

- kruopštumas ir atsakingas požiūris į darbą.
- IT išmanymas.

Vartotojų prioritetai – didelis, jis naudosis sistema.

Administratorius

Vartotojo sprendžiami uždaviniai: Kitų vartotojų registracija, vartotojų duomenų koregavimas, vartotojų naikinimas, sistemos pradinių duomenų suvedimas.

Patirtis dalykinėje srityje – įprastas darbuotojas.

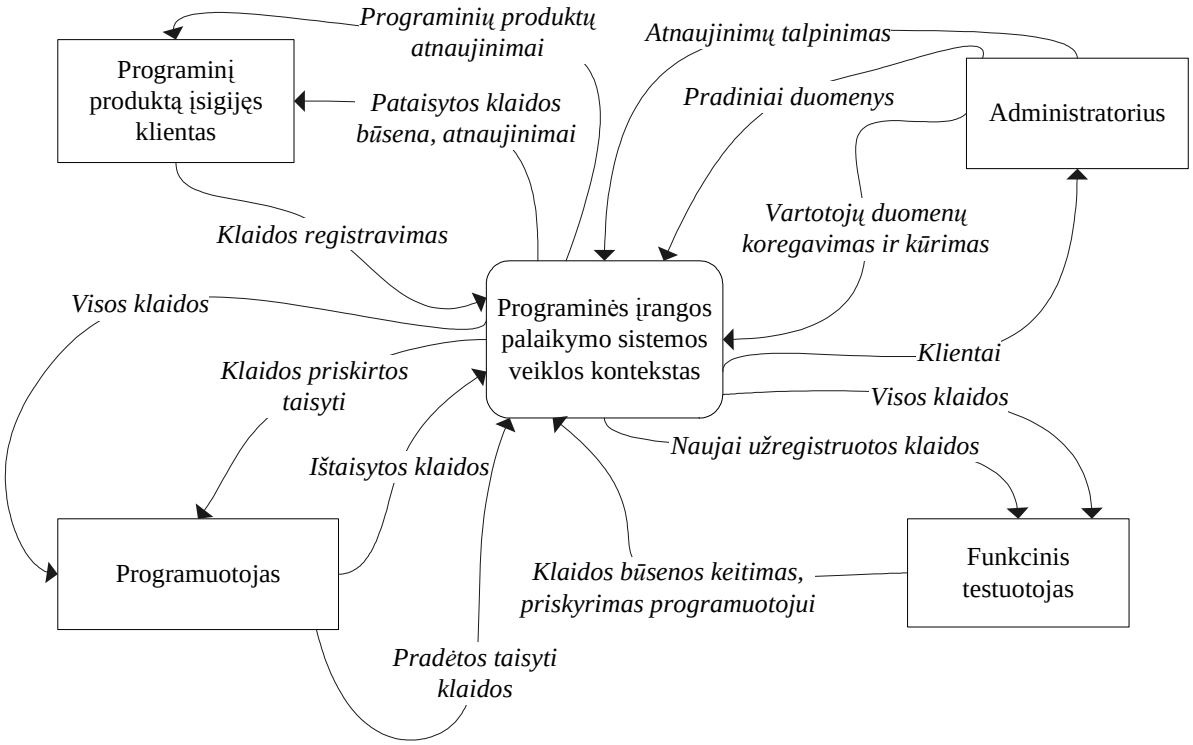
Patirtis informacinėse technologijose – informatikas.

Papildomos vartotojo charakteristikos:

- atsakingas požiūris į darbą
- domėjimasis IT.
- greitas apsimokymo poreikis.
- bendri lietuvių kalbos įgūdžiai.
- Vartotojų prioritetai – antraeiliai.

2.6.2 Sistemos kontekstas

Sukurta programa veiks kaip atskira taikomoji programa, ir nepriklausys kokiai nors kitai sistemai.



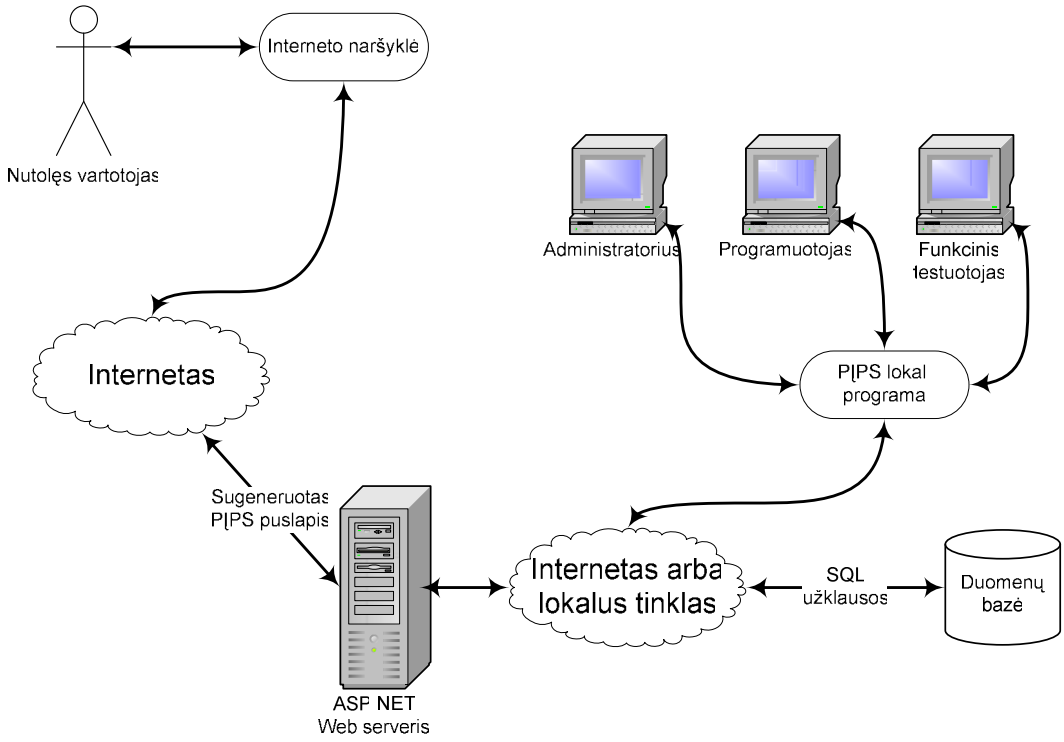
1 pav. Sistemos kontekstas

2.6.3 Apribojimai programų sistemai

Galutiniam programinės įrangos vartotojui sistema turi būti prieinama internetu ir būti nepriklausoma nuo operacinės sistemos. Programos valdymas turi būti paprastas, suprantamas, vartotojo sąsaja prieinama keliomis kalbomis.

Pagrindiniai apribojimai sistemai: *laikas* - sistema turi veikti greitai; ji turi pakankamai greitai persiųsti pranešimus jų adresatams – taip testuotojai greitai galės reaguoti į klientams iškilusias problemas; *naudojimo paprastumas* – vartotojo neturi apsunkinti darbo aplinka, bei konkrečiu momentu nereikalingos funkcijos; ataskaitų informatyvumas, aiškumas, paprastumas, bei įvairumas – kas leis padaryti tikslias išvadas apie pasirinkto laikotarpio atliktus darbus.

2.6.4 Programų sistemos diegimo aplinka



2 pav. Programų sistemos išdėstymo vaizdas

Asmeninis kompiuteris, kuriame bus leidžiama lokali PĮPS sistema, turi būti:

- aprūpintas internetiniu ryšiu, kurio įėjimo bei išėjimo greitis neturi būti mažesnis nei 128kbit/s arba būtų lokalus priėjimas prie duomenų bazės.
- aprūpintas bent 300MHz procesoriumi.
- Turėti bent 128Mb.
- Kietu disku nemažesniu nei 6 GB.

Serveris, kuriame bus patalpinta ASP.NET sistema, turi būti:

- aprūpintas internetiniu ryšiu, kurio įėjimo bei išėjimo greitis neturi būti mažesnis nei 512kbit/s.
- aprūpintas bent 1GHz greičio procesoriumi.
- 512MB darbinės atminties.
- kietu disku nemažesniu nei 10 GB.

PĮPS sistema bendradarbiaus su pasirinktomis sistemomis:

- MS SQL Server 2005 arba MS SQL Express duomenų bazės serveriu.
- IIS Web serveriu (ASP.NET internetinio puslapio publikavimui).
- Operacinė sistema – Windows 98/2000/XP.
- .NET sistema.

3 PROJEKTINĖ DALIS

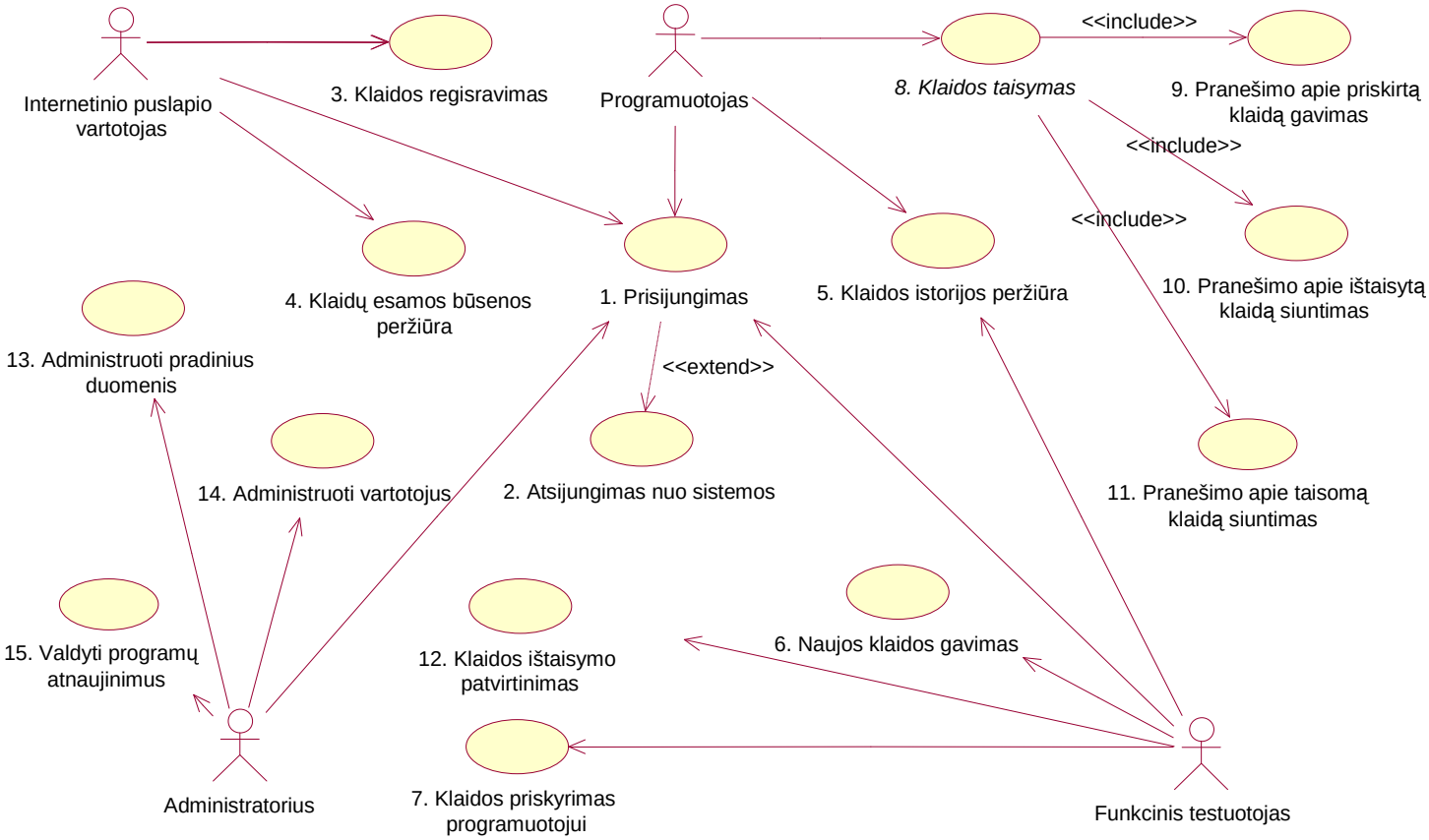
3.1 Tikslas

Ši dokumento dalis skirta pateikti sistemos architektūrą, pasinaudojant skirtingais architektūriniais vaizdais, tokiu būdu išreiškiant skirtingus sistemos architektūros aspektus.

Sistemos architektūra pateikiama keliais aspektais: panaudojimo atvejų (PA), funkcinių reikalavimų, nefunkcinių reikalavimų, išdėstymo ir realizavimo. Modeliavimui panaudotas Rational Rose paketas.

3.2 Sistemos sudėtis

3.2.1 Sistemos ribos



3 pav. Bendra panaudojimo atvejų diagrama

Kaip matosi iš panaudojimo atvejų diagramos, programų sistema naudosis keturi aktoriai:

- Internetinio puslapio vartotojas
- Programuotojas
- Funkcinis testuotojas
- Administratorius

3.2.2 Panaudojimo atvejų sąrašas

1. PANAUDOJIMO ATVEJIS: Prisijungimas.

Vartotojas / Aktorius: Administratorius, funkcinis testuotojas, programuotojas, internetinio puslapio vartotojo.

Aprašas: vartotojas jungiasi prie PĮPS gautu prisijungimo vardu ir slaptažodžiu.

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: aktyvuojama PĮPS sistema ir spaudžiamas meniu mygtukas „Prisijungti...“.

Po sąlyga: prisijungiama prie sistemos.

Ryšys su kitais PA: atsijungimas nuo sistemos

Scenarijus: įvedamas prisijungimo vardas ir slaptažodis.

Alternatyvus scenarijus: klaidingai įvedus prisijungimo vardą arba slaptažodį prie PĮPS neprisijungiama.

2. PANAUDOJIMO ATVEJIS: Atsijungimas nuo sistemos.

Vartotojas / Aktorius: Administratorius, funkcinis testuotojas, programuotojas, internetinio puslapio vartotojo.

Aprašas: prisijungusio vartotojo išsiregistravimas iš PĮPS sistemos.

Prieš sąlyga: vartotojas turi būti prisijungęs prie sistemos.

Sužadinimo sąlyga: spaudžiamas meniu mygtukas „Atsijungti...“.

Po sąlyga: išsiregistravimas iš sistemos.

Ryšys su kitais PA: prisijungimas.

Scenarijus: vartotojui prisijungus prie sistemos ir norint palikti darbo vietą nuo sistemos atsijungiama per meniu pasirinkus punktą „Atsijungti...“.

Alternatyvus scenarijus: nėra.

3. PANAUDOJIMO ATVEJIS: Klaidos registravimas.

Vartotojas / Aktorius: internetinio puslapio vartotojas.

Aprašas: pranešama apie programinėje įrangoje rastas klaidas**,** nurodomas produktas, produkto dalis. Išvardijami veiksmai, kurios atlikus įvyksta klaida.

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: prisijungus prie PĮPS puslapio ir užpildžius būtinus laukus spaudžiamas mygtukas „Registruoti...“.

Po sąlyga: išsaugomas naujas pranešimas apie rastą klaidą.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos.

Scenarijus: nurodomas produktas, produkto dalis. Išvardijami veiksmai, kurios atlikus įvyksta klaida.

Alternatyvus scenarijus: užpildžius nevisus būtinus laukus, klaidos registravimas kartojamas.

4. PANAUDOJIMO ATVEJIS: Klaidų esamos būsenos peržiūra.

Vartotojas / Aktorius: internetinio puslapio vartotojas.

Aprašas: galutinis programinės įrangos vartotojas jungiasi internetu prie PĮPS gautu prisijungimo vardu ir slaptažodžiu ir gali peržiūrėti savo registruotų klaidų būsenas**.**

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: prisijungus prie PĮPS puslapio ir spaudžiamas mygtukas „Klaidų būsenos“.

Po sąlyga: nėra.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos.

Scenarijus: prisijungiama prie sistemos ir spaudžiamas meniu mygtukas „Klaidų būsenos“.

Alternatyvus scenarijus: nėra.

5. PANAUDOJIMO ATVEJIS: Klaidos istorijos peržiūra.

Vartotojas / Aktorius: testuotojas, programuotojas.

Aprašas: gaunama visa su klaida susijusi informacija – kada registruota, kas registravo, kas taisė ir kada ištaisė**.**

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu aktyvuojama forma „Klaidos“, pasirenkama norima ir spaudžiamas mygtukas „Klaidos istorija...“.

Po sąlyga: parodomas langas su klaidos informacija.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos.

Scenarijus: prisijungiama prie sistemos, pasirenkama konkreti dominanti registruota klaida ir spaudžiamas mygtukas „Klaidos istorija...“, kuris atidarys formą su klaidos istorija.

Alternatyvus scenarijus: nėra.

6. PANAUDOJIMO ATVEJIS: Naujos klaidos gavimas.

Vartotojas / Aktorius: testuotojas.

Aprašas: sistema parodo pranešimą apie naujai užregistruotą klaidą (-as)**.**

Prieš sąlyga: vartotojas turi būti prisijungęs prie sistemos.

Sužadinimo sąlyga: galutiniam programinės įrangos vartotojui užregistravus naują klaidą.

Po sąlyga: nėra.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, klaidos priskyrimas programuotojui.

Scenarijus: vartotojas prisijungia prie sistemos, pranešimas apie gautą naują klaidą parodomas automatiškai.

Alternatyvus scenarijus: nėra.

7. PANAUDOJIMO ATVEJIS: Klaidos priskyrimas programuotojui.

Vartotojas / Aktorius: testuotojas.

Aprašas: priskirti programuotojams rastas klaidas taisymui**.**

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu punktu „Gautos klaidos...“ iškviečiama registruotų klaidų forma. Atsifiltruojamos tik naujai gautos klaidos.

Po sąlyga: programuotojui priskiriama klaida taisyti.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, naujos klaidos gavimas.

Scenarijus: Pasirenkama registruota klaida(-os). Pasirenkamas programuotojas iš atsakingos darbo grupės. Spaudžiamas mygtukas „Priskirti“.

Alternatyvus scenarijus: mygtuku „Nutraukti“ forma uždaroma.

8. PANAUDOJIMO ATVEJIS: Klaidos taisymas.

Vartotojas / Aktorius: programuotojas.

Aprašas: susideda iš pranešimo apie priskirtą klaidą gavimo, pranešimo apie taisomą ir ištaisytą klaidą siuntimo

panaudos atvejų panaudos atvejų**.**

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: nėra.

Po sąlyga: nėra.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, pranešimo apie jam priskirtą klaidą gavimas, pranešimo apie ištaisytą klaidą siuntimas, pranešimo apie taisomą klaidą siuntimas.

Scenarijus: nėra.

Alternatyvus scenarijus: nėra.

9. PANAUDOJIMO ATVEJIS: Pranešimo apie priskirtą klaidą gavimas.

Vartotojas / Aktorius: programuotojas.

Aprašas: pamatyti programuotojui priskirtas klaidas.

Prieš sąlyga: programuotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu punktu „Mano klaidos...“ iškviečiama priskirtų klaidų forma. Atsifiltruojamos tik naujai priskirtos klaidos.

Po sąlyga: nėra.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, klaidos taisymas.

Scenarijus: nėra.

Alternatyvus scenarijus: mygtuku „Nutraukti“ forma uždaroma.

10. PANAUDOJIMO ATVEJIS: Pranešimo apie ištaisytą klaidą siuntimas.

Vartotojas / Aktorius: programuotojas.

Aprašas: pranešti apie klaidos taisymo pabaigą**.**

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu punktu „Mano klaidos...“ iškviečiama priskirtų klaidų forma. Atfiltruojamos tik taisomos klaidos.

Po sąlyga: priskirta klaida pažymima kaip baigta taisyti.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, klaidos taisymas.

Scenarijus: pasirenkama klaida(-os). Spaudžiamas mygtukas „Baigta taisyti“.

Alternatyvus scenarijus: mygtuku „Nutraukti“ forma uždaroma.

11. PANAUDOJIMO ATVEJIS: Pranešimo apie taisomą klaidą siuntimas.

Vartotojas / Aktorius: programuotojas.

Aprašas: pranešti apie klaidos taisymo pradžią**.**

Prieš sąlyga: vartotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu punktu „Mano klaidos...“ iškviečiama priskirtų klaidų forma. Atsifiltruojamos tik naujai priskirtos klaidos.

Po sąlyga: klaida pažymima kaip pradėta taisyti.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, klaidos taisymas.

Scenarijus: pasirenkama klaida(-os). Spaudžiamas mygtukas „Pradėta taisyti“.

Alternatyvus scenarijus: mygtuku „Nutraukti“ forma uždaroma.

12. PANAUDOJIMO ATVEJIS: Klaidos ištaisymo patvirtinimas.

Vartotojas / Aktorius: testuotojas.

Aprašas: patikrinti, ar klaida ištaisyta**.** Jei neištaisyta – siųsti pakartotiniam taisymui.

Prieš sąlyga: testuotojas prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu punktu „Gautos klaidos...“ iškviečiama registruotų klaidų forma. Atsifiltruojamos tik taisytos klaidos.

Po sąlyga: klaida pažymima kaip pataisyta/grąžinta taisyti.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos, klaidos istorijos peržiūra.

Scenarijus: Pasirenkama taisyta klaida(-os). Patikrinama, ar klaida tikrai ištaisyta programinėje įrangoje. Pagal situaciją spaudžiamas mygtukas „Ištaisyta“ arba „Pagrąžinti“.

Alternatyvus scenarijus: mygtuku „Nutraukti“ forma uždaroma.

13. PANAUDOJIMO ATVEJIS: Administruoti pradinius duomenis.

Vartotojas / Aktorius: administratorius.

Aprašas: tikslas suvesti pradinius sistemos duomenis (produktus, produktų dalis, galimus reikalavimus sistemoms, darbo grupes)**.**

Prieš sąlyga: administratorius prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu pasirenkamas punktas „Pradiniai duomenys...“.

Po sąlyga: išsaugomi padaryti pakeitimai.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos.

Scenarijus: atidaroma pradinių duomenų forma, suvedami nauji, redaguojami esami duomenys. Spaudžiamas mygtukas „Išsaugoti“.

Alternatyvus scenarijus: mygtuku „Nutraukti“ forma uždaroma.

14. PANAUDOJIMO ATVEJIS: Administruoti vartotojus.

Vartotojas / Aktorius: administratorius.

Aprašas: kuriami nauji vartotojai, suteikiamos konkrečius teisės sistemoje, pakeičiamas slaptažodis, naikinami nereikalingi vartotojai**.**

Prieš sąlyga: administratorius prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu.

Sužadinimo sąlyga: meniu pasirenkamas punktas „Vartotojų administravimas“.

Po sąlyga: išsaugomi padaryti pakeitimai duomenų bazėje.

Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos.

Scenarijus: pasirenkamas vienas iš sukurtų vartotojų ir keičiamas jo slaptažodis arba mygtuku „Šalinti“ pašalinamas; mygtuku „Naujas“ kuriamas naujas vartotojas įvedant jo prisijungimo duomenis.

Alternatyvus scenarijus: išjungiama forma, paspaudus mygtuką „Nutraukti“.

15. PANAUDOJIMO ATVEJIS: Valdyti atnaujinimus.

Vartotojas / Aktorius: administratorius.

Aprašas: talpinami nauji atnaujinimai į internetinį puslapį**.**

Prieš sąlyga: administratorius prie sistemos prisijungia savo prisijungimo vardu ir slaptažodžiu..

Sužadinimo sąlyga: meniu pasirenkamas punktas „Atnaujinimų administravimas“.
Po sąlyga: nauji atnaujinimai matosi internetiniame puslapyje.
Ryšys su kitais PA: prisijungimas, atsijungimas nuo sistemos.
Scenarijus: kuriamas naujas atnaujinimas (nurodomas pavadinimas, data, pastabos ir įkeliamas atnaujinimo faliukas iš kietojo disko). Spaudžiamas mygtukas „Saugoti“.
Alternatyvus scenarijus: išjungiama forma, paspaudus mygtuką „Nutraukti“.

3.3 Funkciniai reikalavimai ir reikalavimai duomenims

3.3.1 Funkciniai reikalavimai

Reikalavimas #:	1	**Reikalavimo tipas:**	1	**Įvykis/panaudojimo atvejis #:**	1
Aprašymas:	Prie sistemos kiekvienas vartotojas turi prisijungti savo vartotojo vardu ir slaptažodžiu.				
Pagrindimas:	Registruojant naują klaidą, skiriant ją taisyti ar ištaisius turi figūruoti atitinkamą darbą atlikęs vartotojas.				
Šaltinis:	Sistemos administratorius.				
Tikimo kriterijus:	Sistema turi žinoti, kokio tipo vartotojas yra prisijungęs, kad jam leistų/neleistų vykdyti atitinkamų funkcijų.				
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	5
Priklausomybės:	#2, #3			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	2	**Reikalavimo tipas:**	1	**Įvykis/panaudojimo atvejis #:**	2
Aprašymas:	Prisijungęs vartotojas turi turėti galimybę atsijungti nuo sistemos.				
Pagrindimas:	Vienu kompiuteriu gali naudotis keli darbuotojai, todėl yra reikalas vartotojui atsijungti nuo sistemos neišjungiant pačios programos.				
Šaltinis:	Sistemos administratorius.				
Tikimo kriterijus:	Sistema negali funkcionuoti be prisijungusio vartotojo.				
Užsakovo tenkinimas:	2			**Užsakovo netenkinimas:**	4
Priklausomybės:	#3			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	3	**Reikalavimo tipas:**	1	**Įvykis/panaudojimo atvejis #:**	1
Aprašymas:	Sistema naudosis 4 tipų vartotojai: administratorius, testuotojas, programuotojas, galutinis programinės įrangos vartotojas.				
Pagrindimas:	Toks vartotojų išskaidymas leis apriboti kiekvieno vartotojo teises.				
Šaltinis:	Sistemos administratorius				
Tikimo kriterijus:	Prisijungus skirtingiems vartotojas galimi skirtingi veiksmai.				
Užsakovo tenkinimas:	4			**Užsakovo netenkinimas:**	5
Priklausomybės:	#1, #2			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	4	**Reikalavimo tipas:**	2	**Įvykis/panaudojimo atvejis #:**	3
Aprašymas:	Galutinis programinės įrangos vartotojas registruoja naujai rastas klaidas internetu.				

Pagrindimas:	Galutiniam programinės įrangos vartotojui sistema turi būti prieinama kuo paprastesniu būdu.		
Šaltinis:	Testuotojas		
Tikimo kriterijus:	Greitas klaidų registravimas internetu.		
Užsakovo tenkinimas:	3	**Užsakovo netenkinimas:**	5
Priklausomybės:	#6	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimas #:	5	**Reikalavimo tipas:**	2	**Įvykis/panaudojimo atvejis #:**	4
Aprašymas:	Sistema parodo registruotų klaidų būsenas.				
Pagrindimas:	Prisijungęs vartotojas turi matyti dabartinę situaciją sistemoje.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:	Visų registruotų klaidų dabartinių būsenų peržiūra.				
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	3
Priklausomybės:	#6, #4			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	6	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	5
Aprašymas:	Testuotojai ir programuotojai peržiūri klaidos istoriją.				
Pagrindimas:	Esant neaiškumams galima atsekti, kas vyko su registruota klaida.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:	Turi matytis visa su klaida susijusi informacija (kas kada registravo, taisė, priskyrė, keitė būseną).				
Užsakovo tenkinimas:	4			**Užsakovo netenkinimas:**	2
Priklausomybės:	#4, #5, #6			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	7	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	6
Aprašymas:	Testuotojas automatiškai turi gauti pranešimą apie naujai užregistruotą klaidą.				
Pagrindimas:	Būtų nepatogu ir neoptimalu laiko atžvilgiu, jei pačiam testuotojui reikėtų kas kažkiek laiko tikrinti sistemą dėl naujai registruotų klaidų.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:					
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	3
Priklausomybės:	#3			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	8	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	7
Aprašymas:	Testuotojui turi galimybę priskirti klaidą taisyti programuotojui.				
Pagrindimas:	Jeigu testuotojui žino, kas turi taisyti klaidą, jis klaidą priskiria konkrečiam programuotojui.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:	Testuotojas priskiria klaidą programuotojui.				
Užsakovo tenkinimas:	4			**Užsakovo netenkinimas:**	5

Priklausomybės:	#3	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimas #:	9	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	8
Aprašymas:	Programuotojas raportuoja apie einamą klaidos būseną (pradėjo taisyti, sutiko taisyti, ištaisė).				
Pagrindimas:	Testuotojas turi žinoti, kokioje stadijoje yra klaidos taisymas.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:					
Užsakovo tenkinimas:	4			**Užsakovo netenkinimas:**	3
Priklausomybės:	#3, #4			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	10	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	9
Aprašymas:	Programuotojas automatiškai gauna pranešimą apie jam priskirtą taisyti klaidą.				
Pagrindimas:	Būtų nepatogu ir neoptimalu laiko atžvilgiu, jei pačiam programuotojui reikėtų kas kažkiek laiko tikrinti sistemą dėl naujai priskirtų klaidų.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:	Automatiškas priskirtos klaidos gavimas per nustatytą laiko tarpą.				
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	2
Priklausomybės:	#3, #9			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	11	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	10
Aprašymas:	Programuotojas ištaisęs klaidą apie tai praneša testuotojui.				
Pagrindimas:	Testuotojas turi žinoti, kokioje stadijoje yra klaidos taisymas.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:	Programuotojas nusiunčia pranešimą apie ištaisytą klaidą testuotojui.				
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	5
Priklausomybės:	#3, #9			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	12	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	11
Aprašymas:	Pradėjęs taisyti klaidą programuotojas apie tai praneša testuotojui.				
Pagrindimas:	Testuotojas turi žinoti, kokioje stadijoje yra klaidos taisymas.				
Šaltinis:	Testuotojas				
Tikimo kriterijus:	Programuotojas nusiunčia pranešimą apie pradėtą taisyti klaidą testuotojui.				
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	5
Priklausomybės:	#3, #9			**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	13	**Reikalavimo tipas:**	3	**Įvykis/panaudojimo atvejis #:**	12
Aprašymas:	Testuotojas pranešimą apie programuotojo ištaisytą klaidą gauna automatiškai.				

Pagrindimas:	Būtų nepatogu ir neoptimalu laiko atžvilgiu, jei pačiam testuotojui reikėtų kas kažkiek laiko tikrinti sistemą dėl naujai ištaisytų klaidų.		
Šaltinis:	Testuotojas		
Tikimo kriterijus:	Automatiškas ištaisytos klaidos gavimas per nustatytą laiko tarpą.		
Užsakovo tenkinimas:	3	**Užsakovo netenkinimas:**	5
Priklausomybės:	#3, #9	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimas #:	14	**Reikalavimo tipas:**	4	**Įvykis/panaudojimo atvejis #:**	13
Aprašymas:	Administratorius suveda sistemai funkcionuoti reikalingus pradinius duomenis.				
Pagrindimas:	Sistemai funkcionuoti reikalinga, kad būtų suvesti vartotojai, produktai, produktų dalys ir pan.				
Šaltinis:	Sistemų administratorius				
Tikimo kriterijus:	Pradinių duomenų įvedimas ir koregavimas.				
Užsakovo tenkinimas:	4		**Užsakovo netenkinimas:**		5
Priklausomybės:	#3		**Konfliktai:**		Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	15	**Reikalavimo tipas:**	4	**Įvykis/panaudojimo atvejis #:**	14
Aprašymas:	Administratorius kuria naujus vartotojus, keičia esamų duomenis, šalina nereikalingus.				
Pagrindimas:	Vartotojų kūrimas ir jų duomenų koregavimas yra būtinas.				
Šaltinis:	Sistemų administratorius				
Tikimo kriterijus:	Administratorius gali kurti/redaguoti vartotojus.				
Užsakovo tenkinimas:	3		**Užsakovo netenkinimas:**		5
Priklausomybės:	#1, #2, #3		**Konfliktai:**		Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

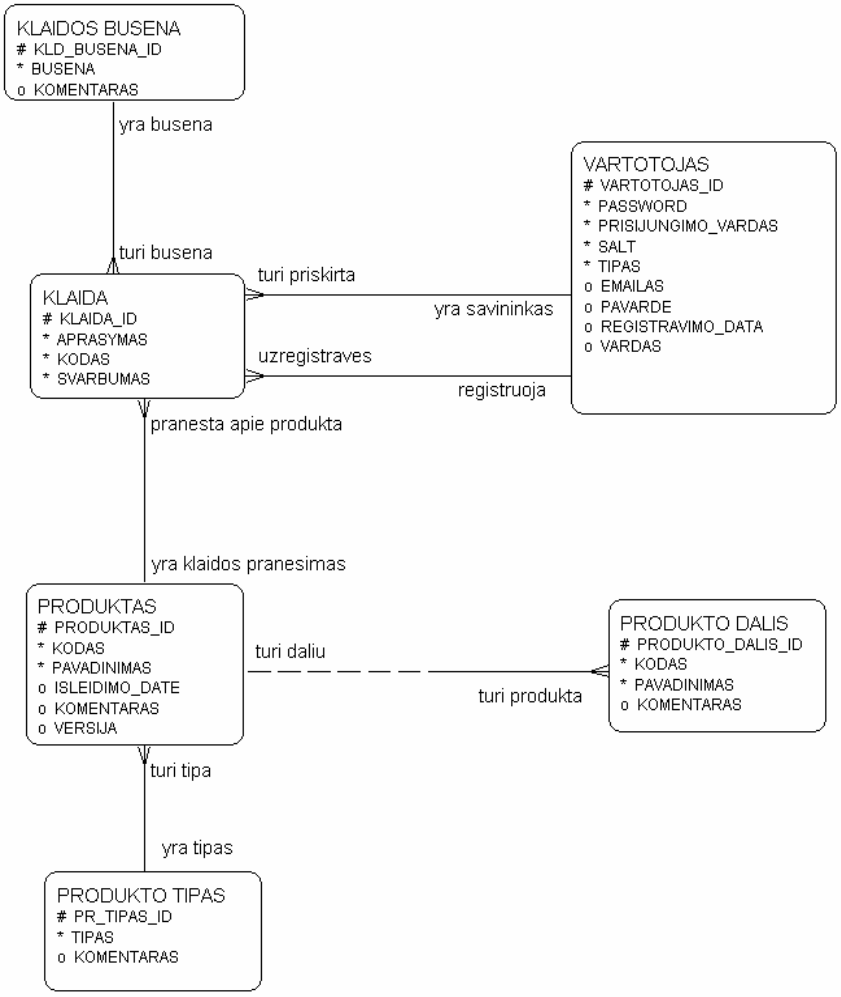
Reikalavimas #:	16	**Reikalavimo tipas:**	4	**Įvykis/panaudojimo atvejis #:**	15
Aprašymas:	Administratorius valdo atnaujinimus – atsiradus naujiems, juos patalpiną į internetinį puslapį.				
Pagrindimas:	Atnaujinimai reikalingi galutiniams programinės įrangos vartotojams. Jei jų įsigyta programa neturi atnaujinimų mechanizmo, atnaujinimus galima bus parsisiųsti per internetinį puslapį.				
Šaltinis:	Sistemų administratorius				
Tikimo kriterijus:	Administratorius gali kurti/redaguoti atnaujinimus.				
Užsakovo tenkinimas:	3		**Užsakovo netenkinimas:**		5
Priklausomybės:	#3		**Konfliktai:**		Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.				

Reikalavimas #:	17	**Reikalavimo tipas:**	4	**Įvykis/panaudojimo atvejis #:**	14
Aprašymas:	Administratorius gali išvalyti visą informaciją susijusią su klaida.				
Pagrindimas:	Gali būti atvejų, kai dėl neapsižiūrėjimo klaida palaikoma funkcionalumo trūkumas arba kitas su klaida nesusijęs faktorius.				
Šaltinis:	Projektų vadovė				
Tikimo kriterijus:	Šalinant klaidą, pašalinama visa su ja susijusi informacija (duomenų bazės šiukšlės).				
Užsakovo tenkinimas:	3		**Užsakovo netenkinimas:**		5

Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

3.3.2 Reikalavimai duomenims

Pradinėje stadijoje, išanalizavus panaudos atvejus ir funkcinius reikalavimus, buvo sudaryti pradiniai reikalavimai duomenims:



4 pav. Pradinė duomenų struktūra

3.4 Nefunkciniai reikalavimai

Reikalavimai sistemos išvaizdai

Reikalavimas #:	18 **Reikalavimo tipas:**	5	
Aprašymas:	Sistemos išvaizda turi būti aiški ir suprantama eiliniam vartotojui.		
Pagrindimas:	Nereikalingos spalvos ir animacija gali nukreipti dėmesį nuo darbo.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Vartotojo sąsajoje nebus dėmesį blaškančių komponentų.		
Užsakovo tenkinimas:	4	**Užsakovo netenkinimas:**	2
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimai panaudojamumui

Reikalavimas #:	19	**Reikalavimo tipas:**	6
Aprašymas:	Intuityviai išdėstyta informacija		
Pagrindimas:	Naudojimo patogumas.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Vartotojas pripranta prie informacijos išdėstymo viename lange greičiau nei per 5min, dalyvaujant konsultantui.		
Užsakovo tenkinimas:	3	**Užsakovo netenkinimas:**	4
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimas #:	20	**Reikalavimo tipas:**	6
Aprašymas:	Duomenų filtravimas		
Pagrindimas:	Esant daug duomenų greitesnei paieškai reikalingas įrašų filtravimas.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Formose realizuotas filtravimas pagal būdingiausius laukus.		
Užsakovo tenkinimas:	4	**Užsakovo netenkinimas:**	5
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimai vykdymo charakteristikoms

Reikalavimas #:	21	**Reikalavimo tipas:**	7
Aprašymas:	Apie naujai registruotą, priskirtą taisyti ar ištaisytą klaidą sistema turi automatiškai pranešti per 2 min.		
Pagrindimas:	Sistema kuriama, kad pagreitintų darbą.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Naujai registruota, priskirta taisyti ar ištaisyta klaida sistemoje pasirodo greičiau nei per 2 min.		
Užsakovo tenkinimas:	3	**Užsakovo netenkinimas:**	5
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimai veikimo sąlygoms

Reikalavimas #:	22	**Reikalavimo tipas:**	8
Aprašymas:	Internetinis puslapis turi veikti su visomis populiariausiomis naršyklėmis.		
Pagrindimas:	Vartotojai naudojasi įvairiomis naršyklėmis.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Internetinis puslapis atrodys taip pat ant visų populiariausių naršyklių.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:**	3
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Reikalavimai saugumui

Reikalavimas #:	22	**Reikalavimo tipas:**	9
Aprašymas:	Vartotojų asmeninė informacija turi būti saugoma užkoduota.		
Pagrindimas:	El. paštas, slaptažodis negali patekti į negerų žmonių rankas.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Vartotojų asmeninė informacija saugoma užkoduota saugiu kodu.		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:**	4
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

Kultūriniai-politiniai reikalavimai

Reikalavimas #:	23	**Reikalavimo tipas:**	10
Aprašymas:	Sistema negali turėti frazių ar paveikslų, įžeidžiančių kitos rasės ar tikėjimo žmones.		
Pagrindimas:	Tokia sistema automatiškai būtų atmesta didelio rato vartotojų.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Sistema neturės frazių ar paveikslų, kurie galėtų įžeisti kažkokios rasės ar tikėjimo žmogų.		
Užsakovo tenkinimas:	3	**Užsakovo netenkinimas:**	5
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

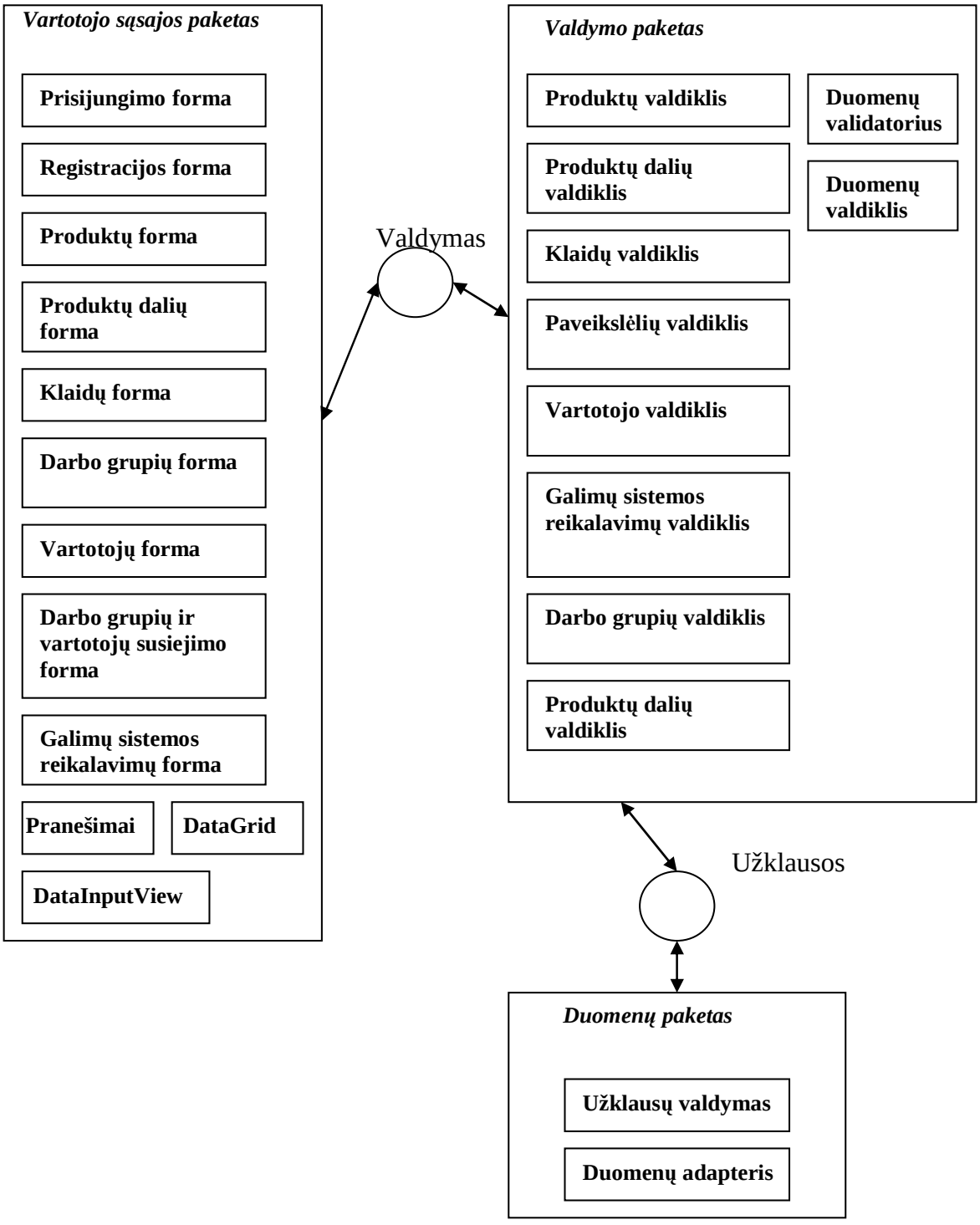
Teisiniai reikalavimai

Reikalavimas #:	17	**Reikalavimo tipas:**	11
Aprašymas:	Sistema negali būti kuriama naudojantis nelegalia programine įranga ar nelegaliais komponentais.		
Pagrindimas:	Tai prasilenktų su LR įstatymais.		
Šaltinis:	Projektų vadovė		
Tikimo kriterijus:	Sistema bus kuriama su legalia programine įranga ir nenaudos nelegalių komponentų.		
Užsakovo tenkinimas:	4	**Užsakovo netenkinimas:**	5
Priklausomybės:	Nėra	**Konfliktai:**	Nėra
Istorija:	Užregistruotas 2005 m. gruodžio 18 d.		

3.5 Programų sistemos statinis vaizdas

3.5.1 Apžvalga

Sistemą sudarantys paketai:



5 pav. Sistemą sudarantys paketai

3.5.2 Paketų detalizavimas

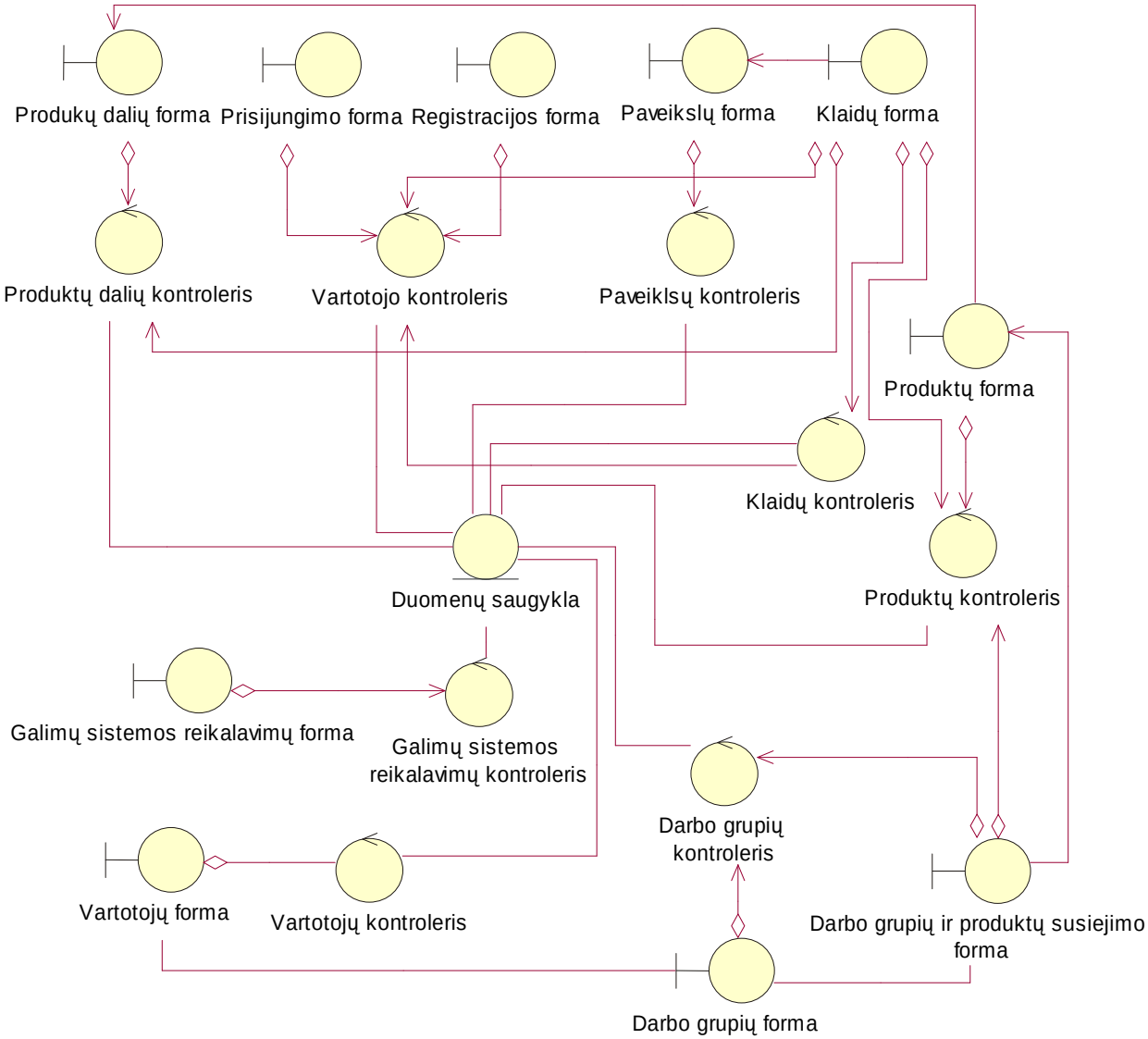
Vartotojo sąsajos paketas skirtas saugoti formas, kurių pagalba vartotojas bendrauja su sistema. Į ši paketą įeina: prisijungimo forma, registracijos forma, produktų forma, produktų dalių forma, klaidų forma, darbo grupių forma, vartotojų forma, darbo grupių ir vartotojų susiejimo forma, galimų sistemos reikalavimų forma. Taip pat šis paketas nauda

papildomus komponentus: pranešimų (*Messages*), duomenų lentelės (*DataGrid*) ir duomenų įvedimo (*DataInputView*).

Valdymo paketas skirtas talpinti klases darbui su duomenimis. Į šį paketą įeina: produktų kontroleris, produktų dalių kontroleris, klaidų kontroleris, paveikslėlių kontroleris, vartotojo kontroleris, galimų sistemos reikalavimų kontroleris, darbo grupių kontroleris, produktų dalių kontroleris.

Duomenų paketas skirtas žemiausio lygio bendravimui su duomenimis. Susideda iš dviejų klasių: užklausų valdymo klasė (*QueryManager*) ir duomenų adapterio klasės (*DataBaseAdapter*).

Visų šių paketų klasės ir jų tarpusavio ryšiai pateikti paveiksle:

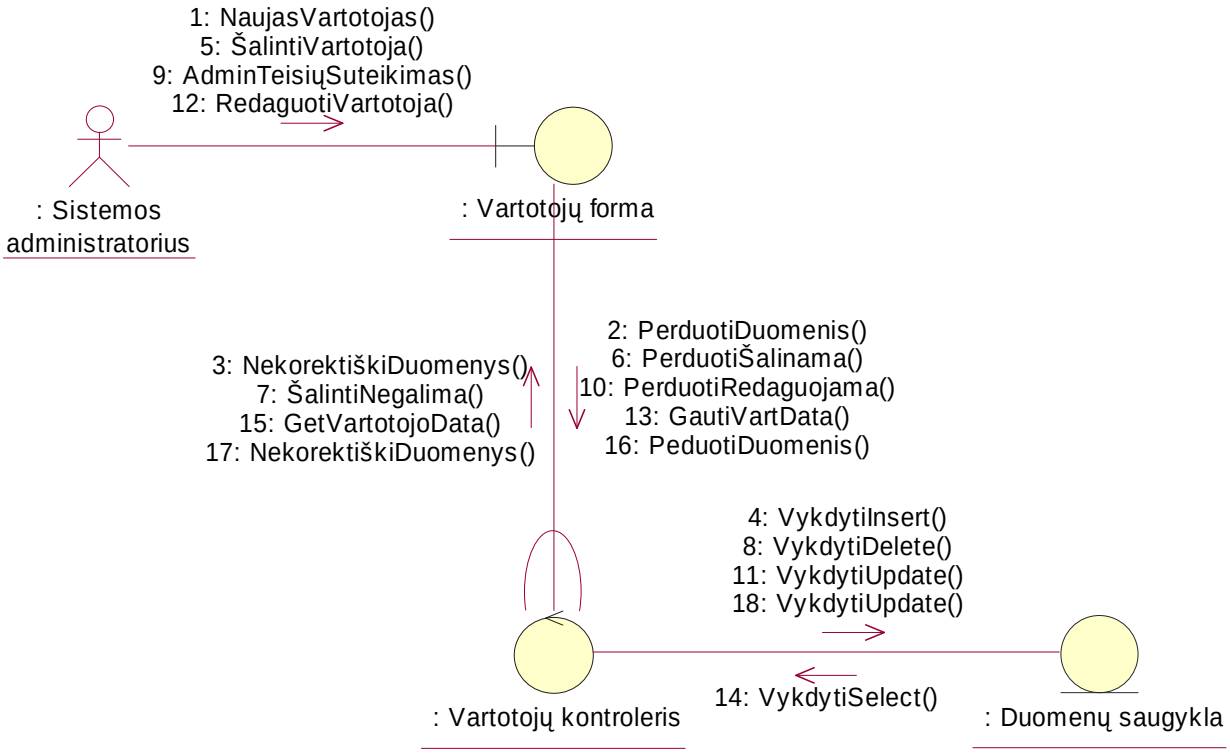


6 pav. Bendra klasių diagrama

3.6 Sistemos dinaminis vaizdas

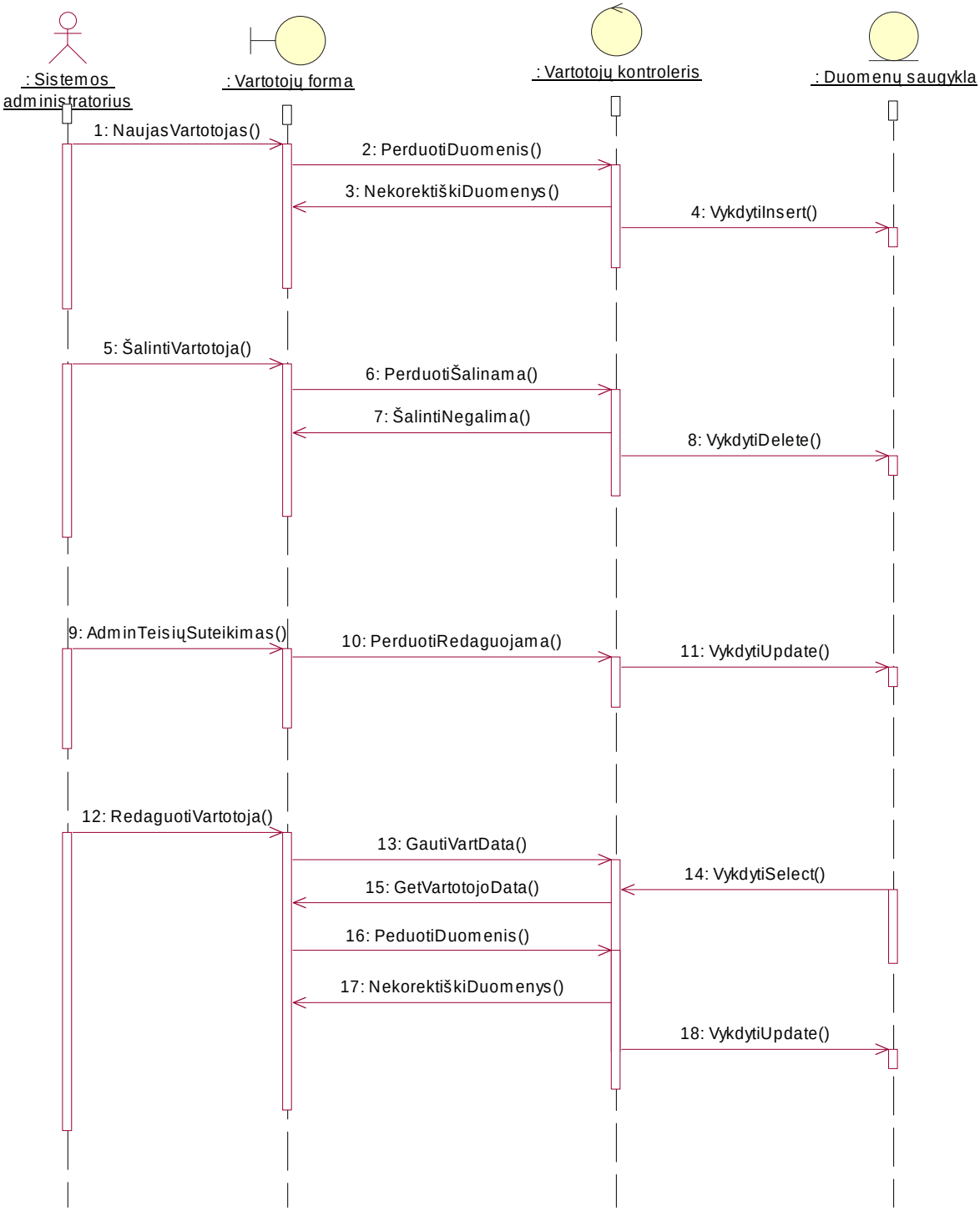
3.6.1 Panaudojimo atvejui „Sistemos administravimas“

Bendradarbiavimo diagrama



7 pav. PA „Sistemos administravimas“. Bendradarbiavimo diagrama.

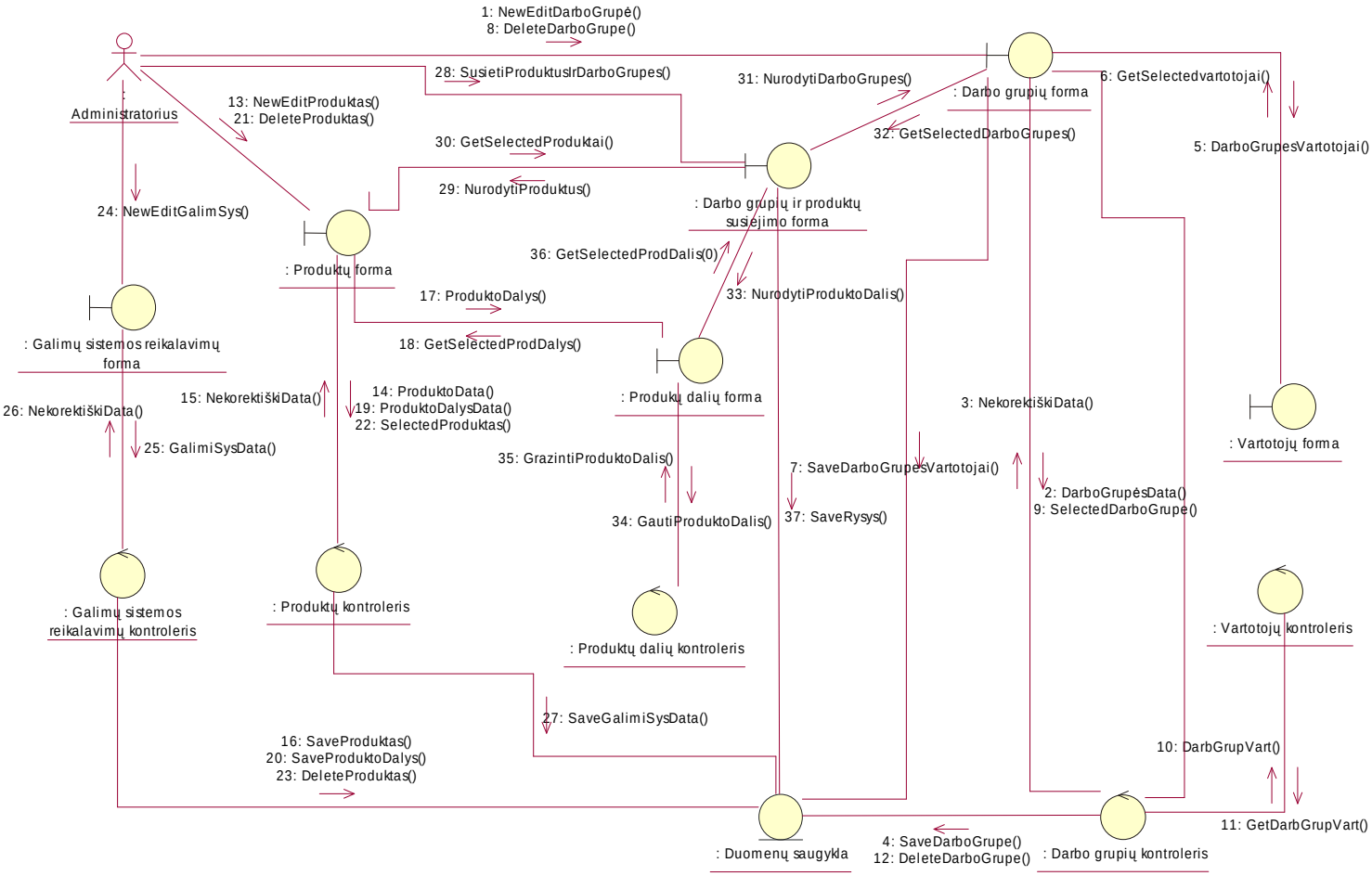
Sekos diagrama



8 pav. PA „Sistemos administravimas“. Sekos diagrama.

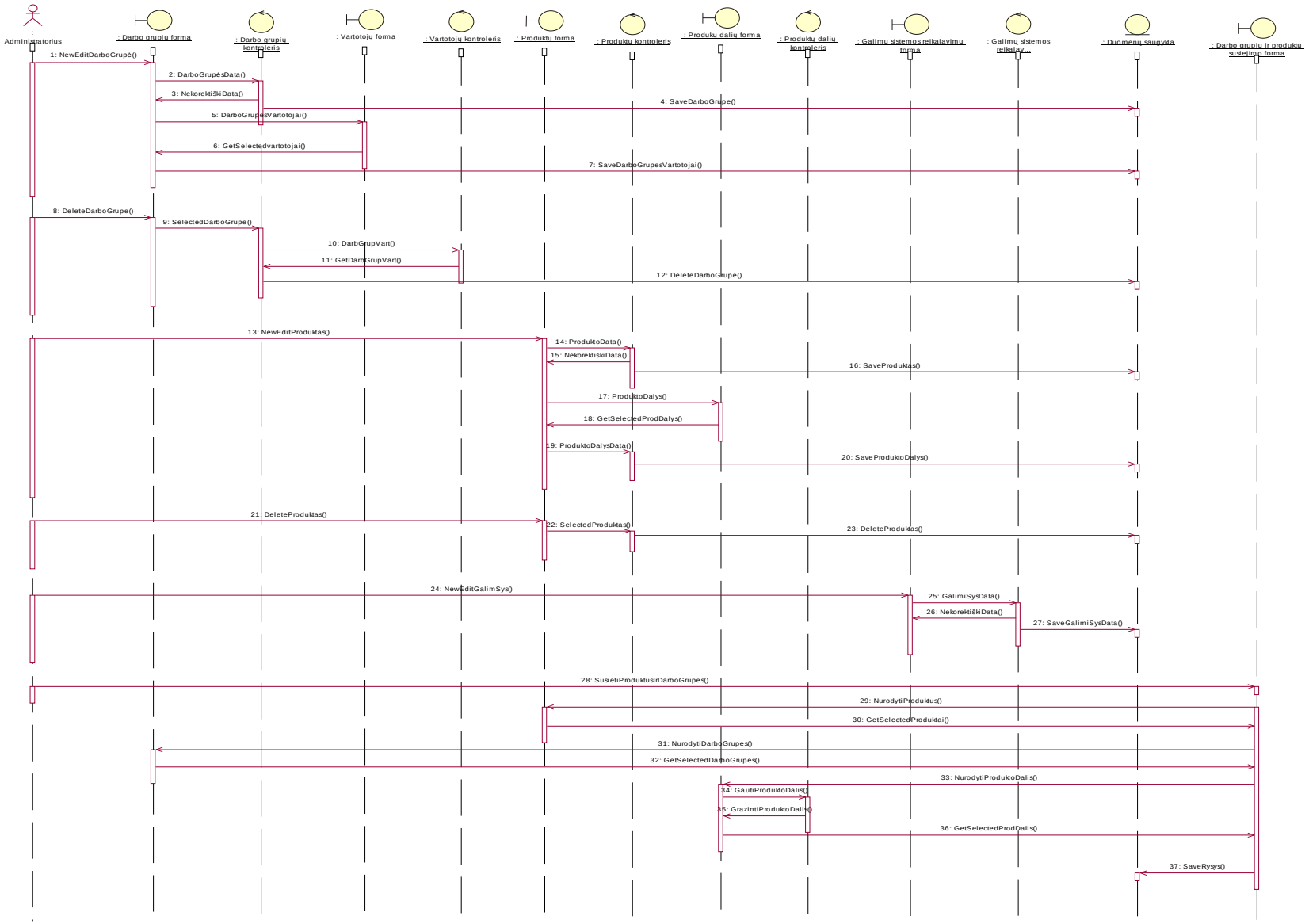
3.6.2 Panaudojimo atvejui „PĮPS administravimas“

Bendradarbiavimo diagrama



9 pav. PA „PĮPS administravimas“. Bendradarbiavimo diagrama.

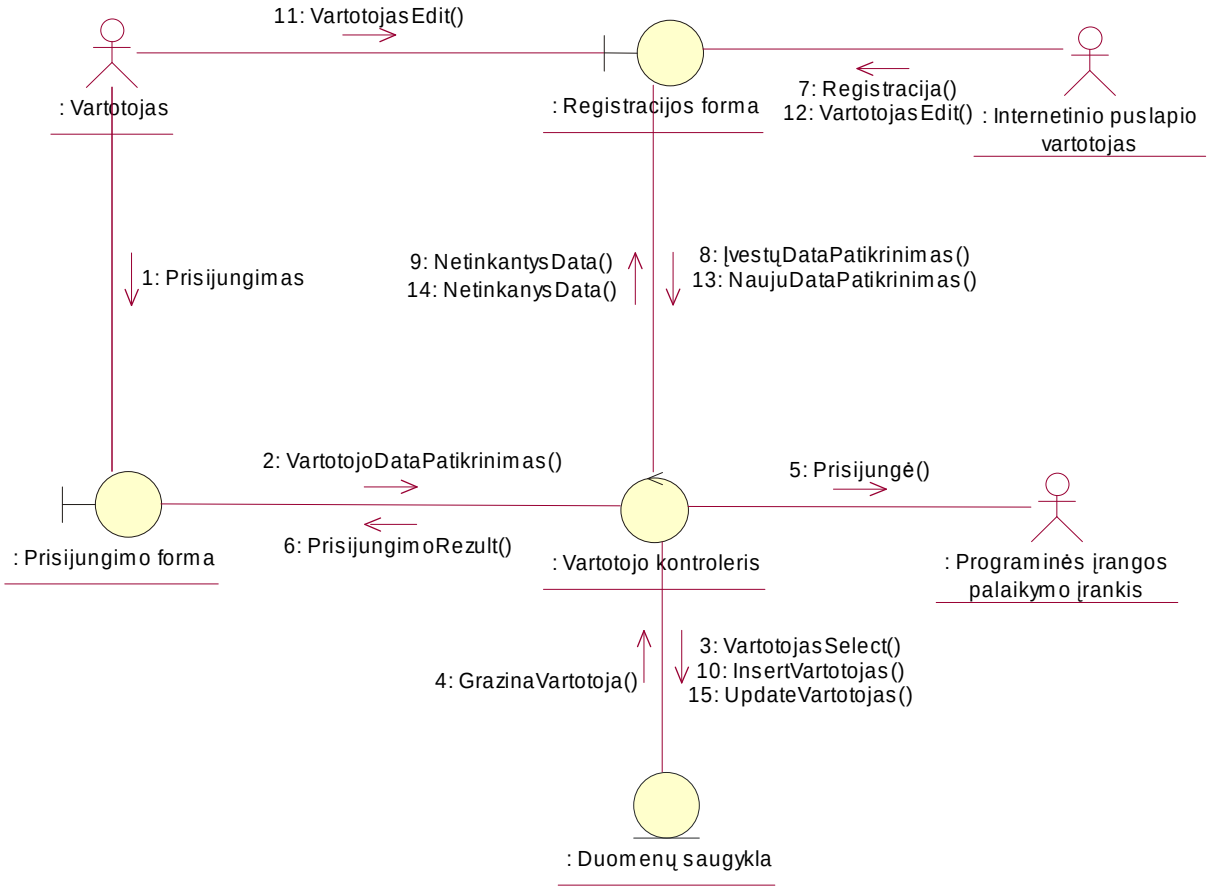
Sekos diagrama



10 pav. PA „PĮPS administravimas“. Sekos diagrama.

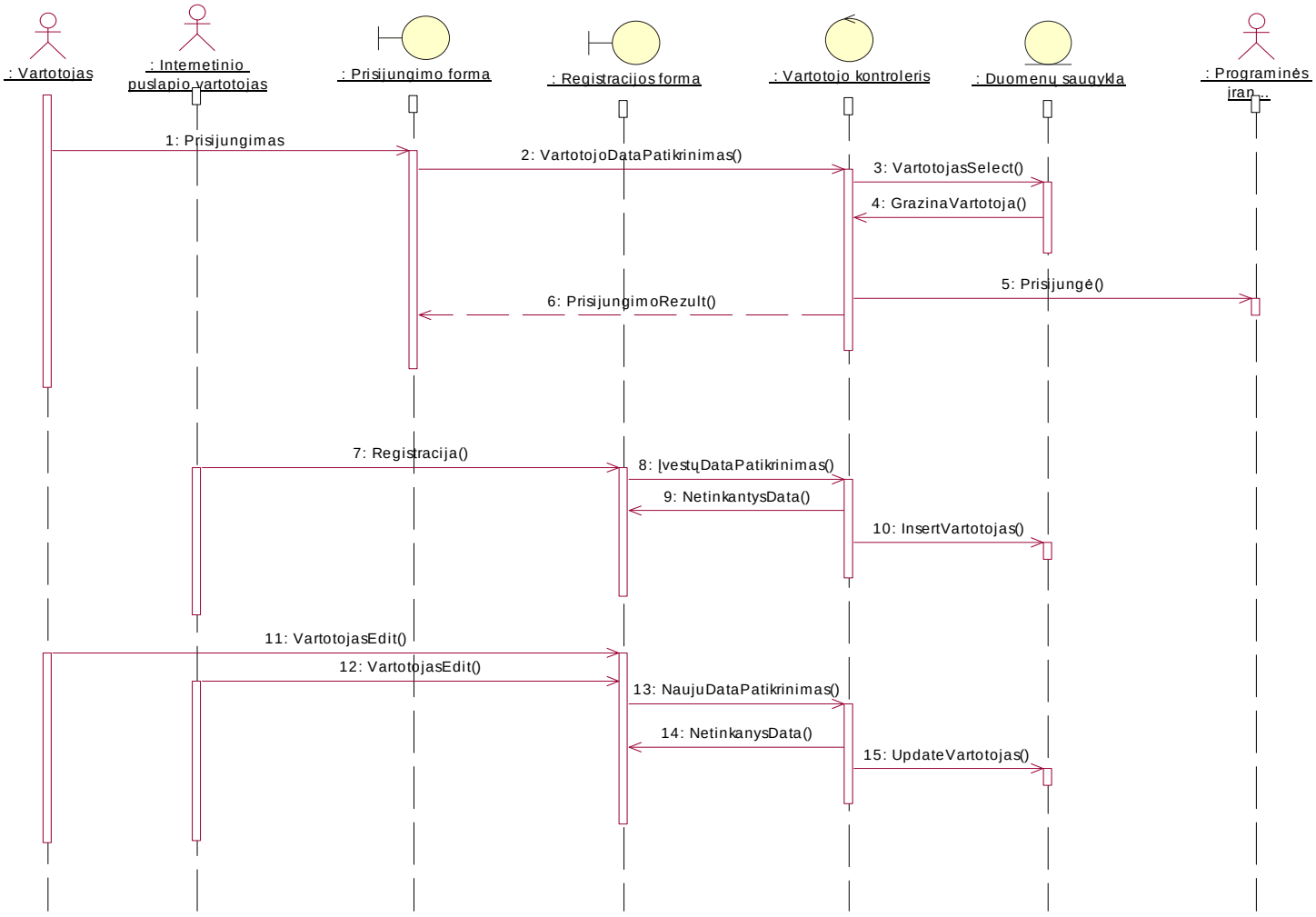
3.6.3 Panaudojimo atvejui „Galutinio programinės įrangos vartotojo prisijungimas“

Bendradarbiavimo diagrama



11 pav. PA „Galutinio programinės įrangos vartotojo prisijungimas“. Bendradarbiavimo diagrama.

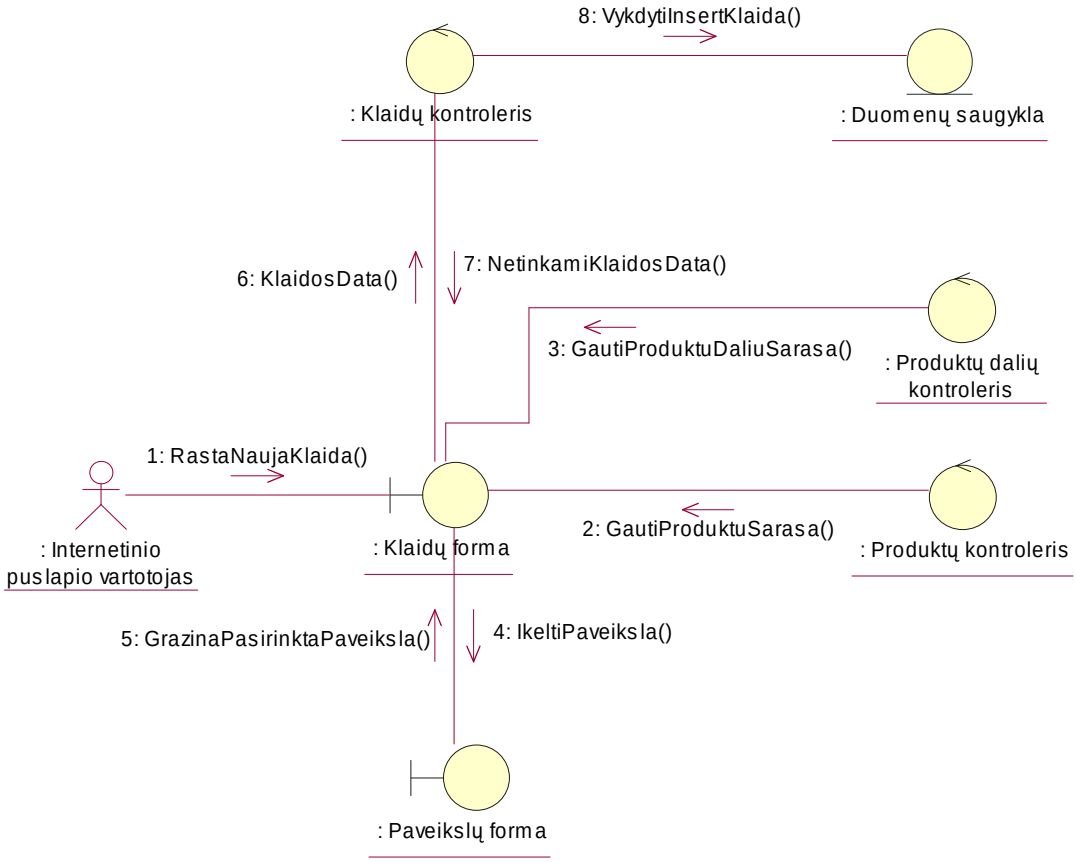
Sekos diagrama



12 pav. PA „Galutinio programinės įrangos vartotojo prisijungimas“. Sekos diagrama.

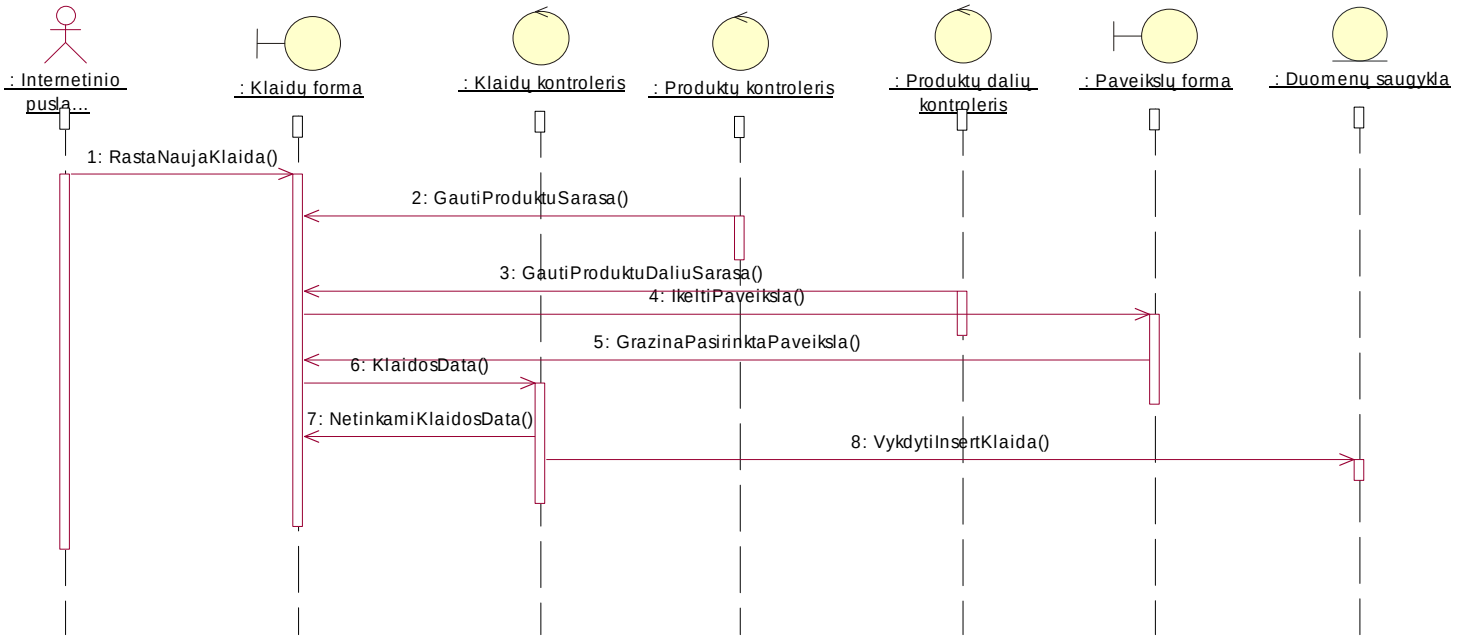
3.6.4 Panaudojimo atvejui „Rastos klaidos užregistravimas“

Bendradarbiavimo diagrama



13 pav. PA „Rastos klaidos užregistravimas“. Bendradarbiavimo diagrama.

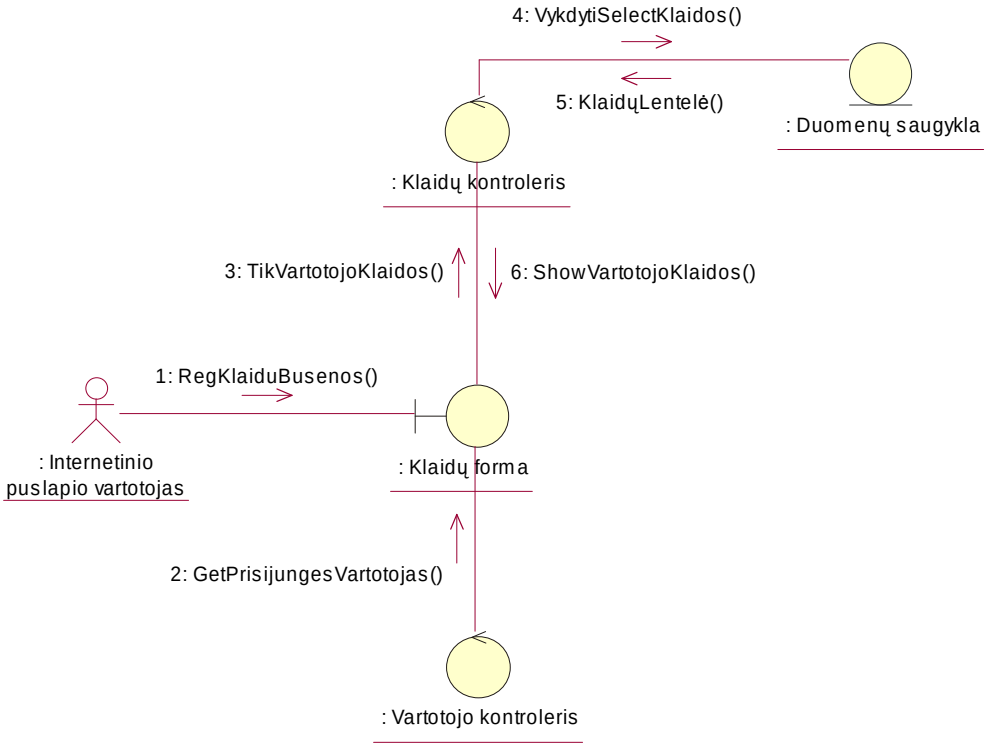
Sekos diagrama



14 pav. PA „Rastos klaidos užregistravimas“. Sekos diagrama.

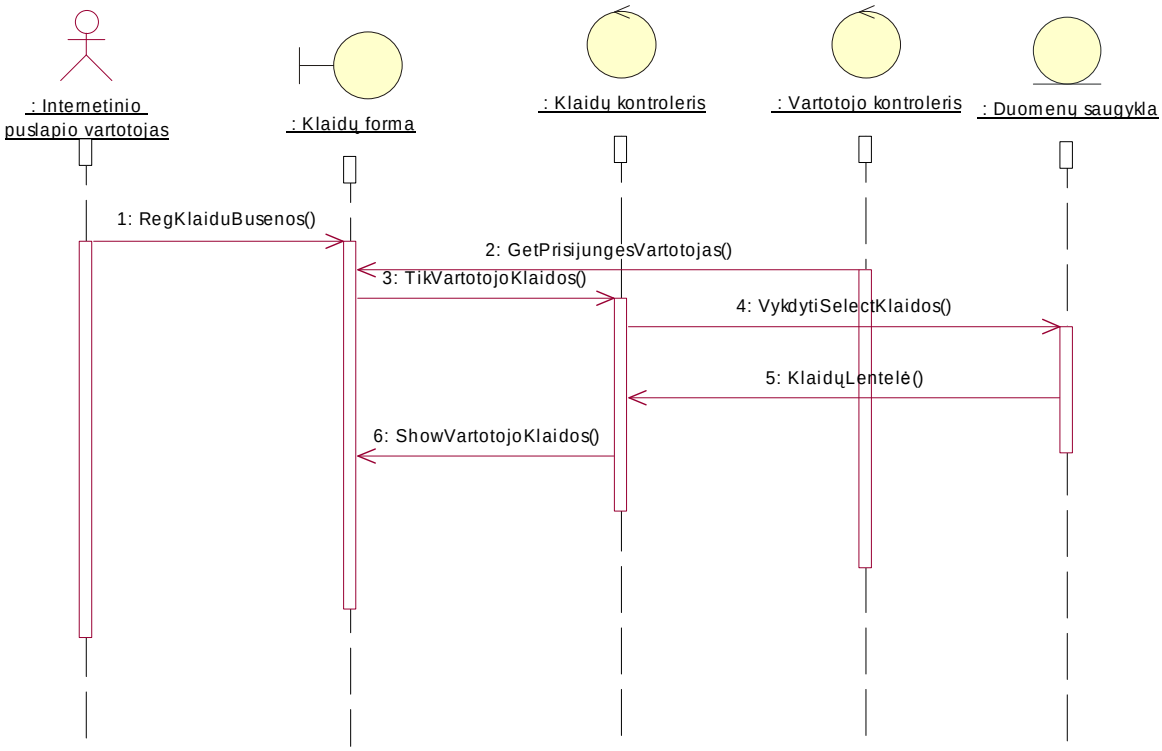
3.6.5 Panaudojimo atvejui „Registruotų klaidų būsenų peržiūra“

Bendradarbiavimo diagrama



15 pav. PA „Registruotų klaidų būsenų peržiūra“. Bendradarbiavimo diagrama.

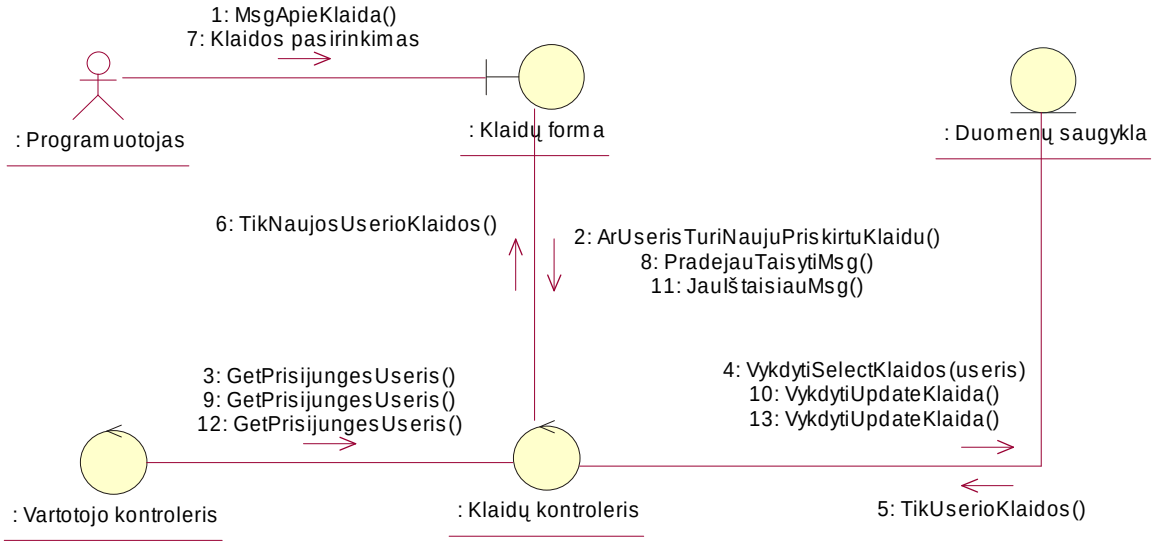
Sekos diagrama



16 pav. PA „Registruotų klaidų būsenų peržiūra“. Sekos diagrama.

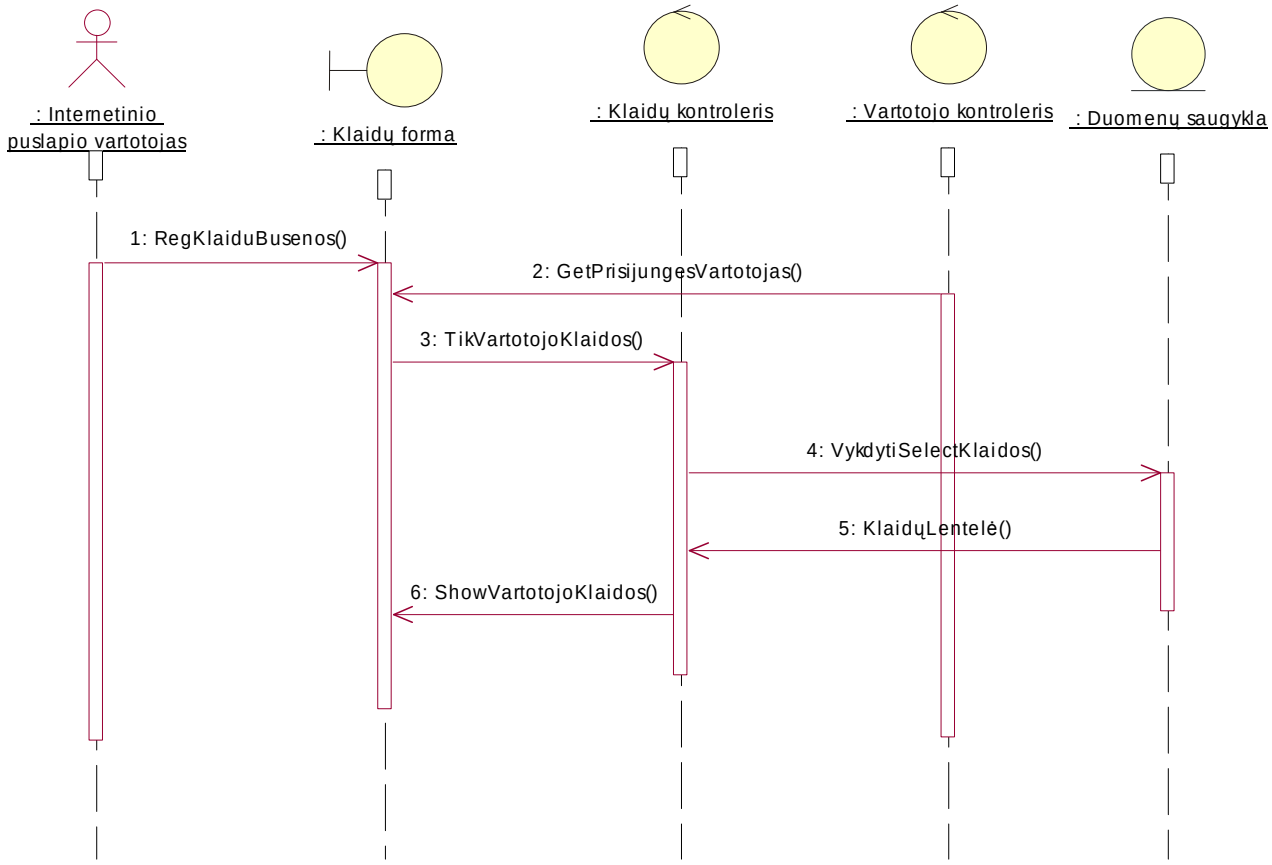
3.6.6 Panaudojimo atvejui „Klaidos taisymas“

Bendradarbiavimo diagrama



17 pav. PA „Klaidos taisymas“. Bendradarbiavimo diagrama.

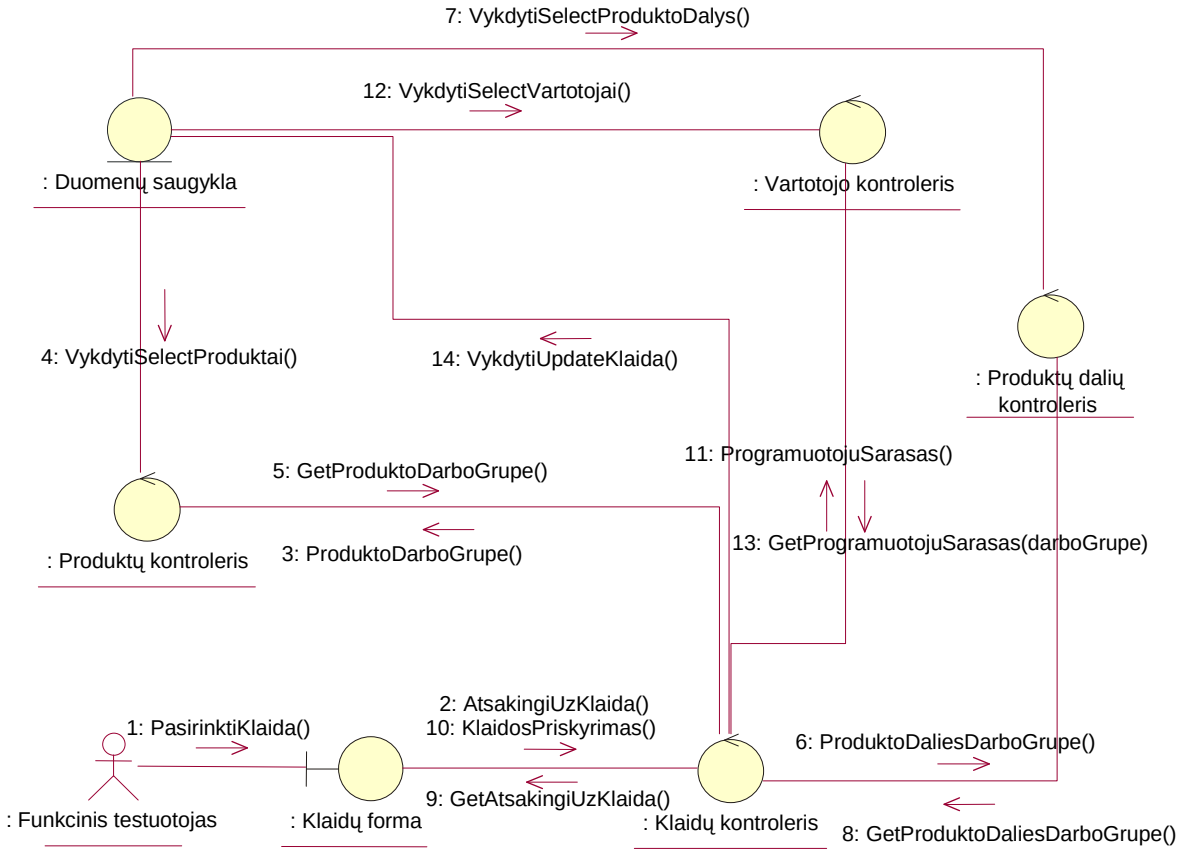
Sekos diagrama



18 pav. PA „Klaidos taisymas“. Sekos diagrama.

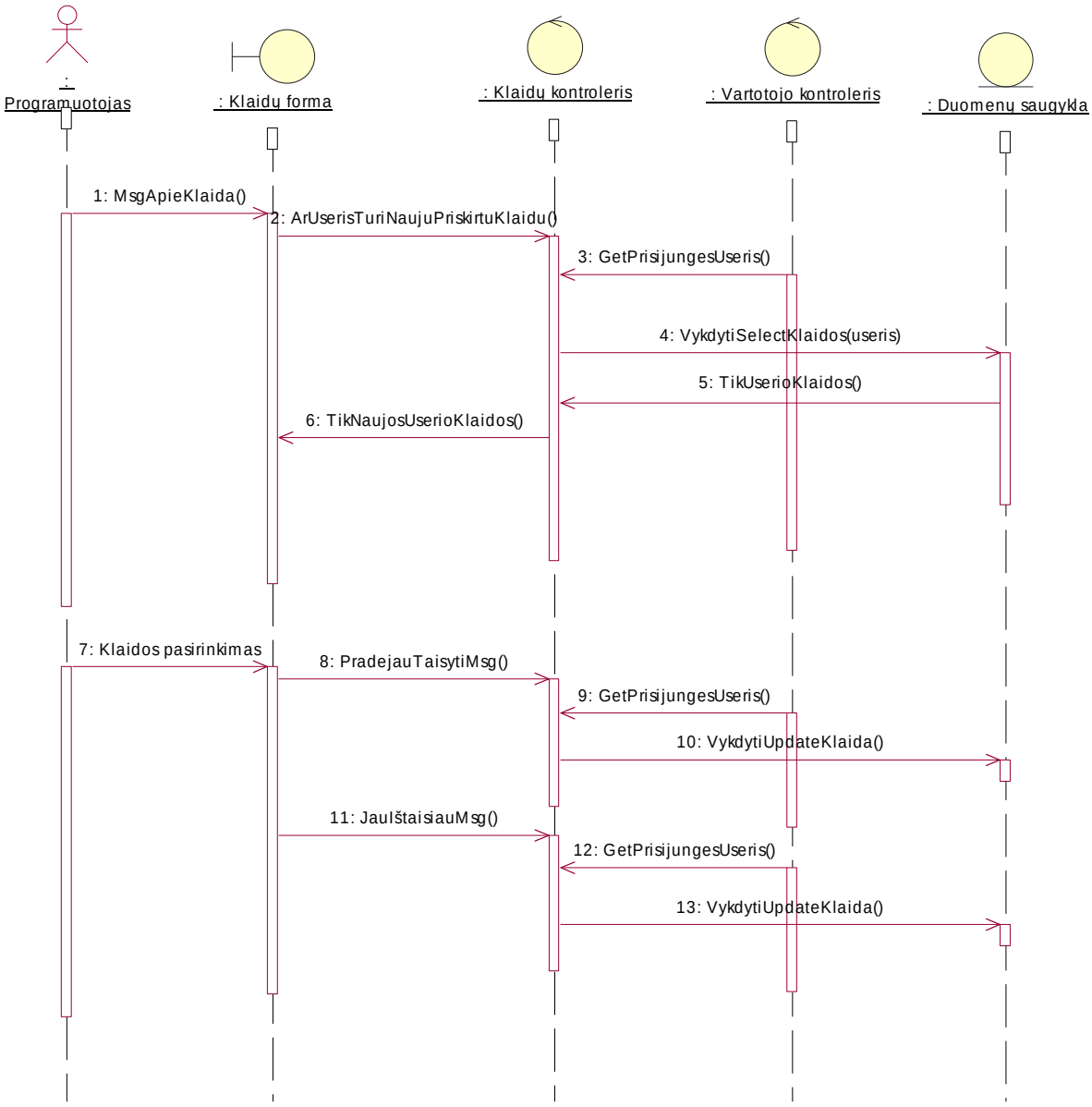
3.6.7 Panaudojimo atvejui „Klaidos priskyrimas programuotojui, ištaisytos klaidos analizė“

Bendradarbiavimo diagrama



19 pav. PA „Klaidos priskyrimas programuotojui, ištaisytos klaidos analizė“. Bendradarbiavimo diagrama.

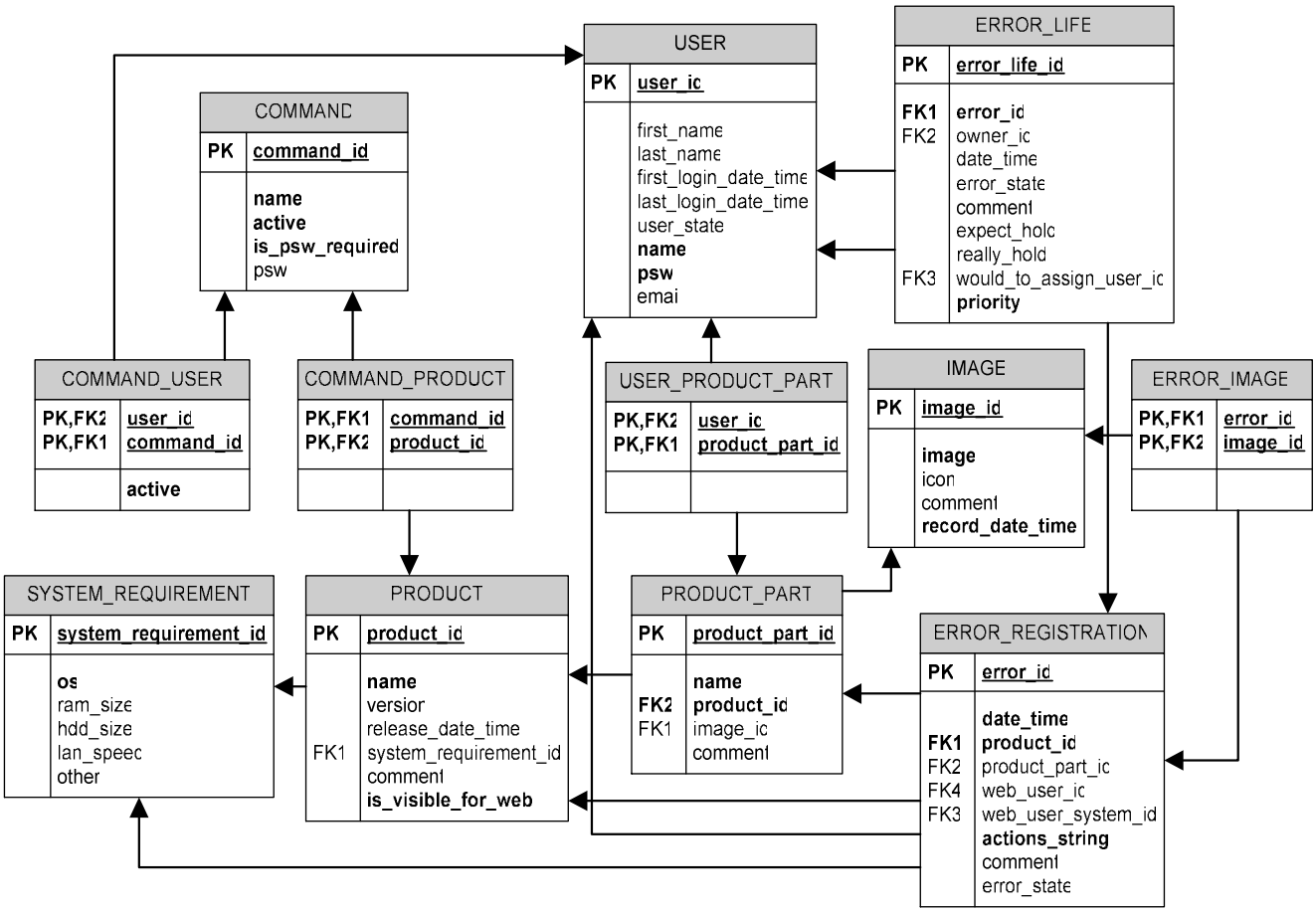
Sekos diagrama



20 pav. PA „Klaidos priskyrimas programuotojui, ištaisytos klaidos analizė“. Sekos diagrama.

3.7 Duomenų vaizdas

Duomenų vaizdas nubraižytas su MS Visio 2003 paketu[1].



21 pav. Galutinis duomenų vaizdas

3.8 Dydis ir našumas

Kuriama programų sistema yra vidutinio dydžio. Sistemos našumą daugiausia įtakos pasirinktos kūrimo priemonės ir interneto ryšio sparta.

3.9 Apibendrinimas

Pagal aprašytą PS architektūrą buvo realizuota užsakyta programinė įranga, kuri šiuo metu yra sėkmingai naudojama užsakovo įmonėje.

[1] Lentelių ir atributų pavadinimai anglų kalba, nes to reikalavo programų sistemos užsakovo įmonė.

4 TYRIMO DALIS

4.1 Paskirtis

Tiriamosios dalies tikslas apibendrinti programų sistemos kokybės vertinimo tikslus, apžvelgti kokybės vertinimo procesą ir jo rezultatus. Kartu šioje dalyje tyrinėjamos programų sistemos kokybės tobulinimo galimybės.

4.2 Realiai atlikto darbo kokybės analizės tikslai

- ✓ Patikrinti ar programų sistema atitinka reikalavimų specifikaciją.
- ✓ Įsitikinti ar programų sistema sukurta pagal standartus.
- ✓ Įvertinti sistemos našumą, patikimumą bei testuojamumą.
- ✓ Įvertinti ar vartotojams lengva naudotis sistema.
- ✓ Įvertinti programų sistemos tobulinimo galimybes.

4.3 Funkcionalumo tyrimas

Tyrimo eigoje programų sistema naudojosi įvairaus lygio vartotojai. Buvo siekiama patikrinti sistemos funkcionalumo atitikimą vartotojo reikalavimams. Vartotojams tyrimo metu buvo pateiktas sistemos funkcijų sąrašas, kuriame jie pažymėjo programų sistemos atitikimą specifikacijai (Lentelė Nr. 3. Programų sistemos peržiūros rezultatai). Išpildyti reikalavimai pažymėti pliusu, o atskiriems panaudojimo atvejams nenumatyti veiksmai pažymėti minusu.

Lentelė Nr. 3. Programų sistemos peržiūros rezultatai

Reikalavimo Nr.	**Funkcionalumas**	**Aktoriai[2]**			
		A	**T**	**P**	**V**
1	Prisijungimas	+	+	+	+
2	Atsijungimas nuo sistemos	+	+	+	+
3	Klaidos registravimas	-	-	-	+
4	Klaidų esamos būsenos peržiūra	-	+	+	+
5	Klaidos istorijos peržiūra	-	+	+	+
6	Naujos klaidos gavimas	-	+	+	-
7	Klaidos priskyrimas programuotojui	-	+	-	-
8	Klaidos taisymas	-	-	+	-
9	Pranešimo apie priskirtą klaidą gavimas	-	-	+	-
10	Pranešimo apie ištaisytą klaidą siuntimas	-	-	+	-
11	Pranešimo apie taisomą klaidą siuntimas	-	+	-	-
12	Klaidos ištaisymo patvirtinimas	-	+	-	-
13	Administruoti pradinius duomenis	+	-	-	-
14	Administruoti vartotojus	+	-	-	-
15	Valdyti atnaujinimus	+	-	-	-

Gauti tyrimo rezultatai patvirtino programų sistemos funkcionalumo atitikimą sistemos specifikacijai.

[2] Aktoriai: A – administratorius, T – funkcinis testuotojas, P – programuotojas, V – vartotojas.

4.4 Standartų laikymasis

Programų sistemos kūrimo procesas aprašytas projektinėje dalyje. Kuriant PS buvo laikomasi šio dokumento ir jame aprašytų metodikų.

4.5 Sistemos našumas

PS veikimo greitis didele dalimi priklauso nuo interneto ryšio greičio, nuo vartotojo ryšio spartos. Papildomai sistemos darbo greitį gali įtakoti duomenų bazės serverio ir IIS serverio didelis apkrovimas. Sistemos išnaudojama kietojo disko vieta yra nedidelė ir daug resursų šiuo atžvilgiu nereikalauja. Daugiausia vietos gali prireikti duomenų bazei, jeigu vartotojų skaičius išaugtų iki labai didelio.

4.6 Vartotojo sąsaja

Realizuotos programų sistemos vienas iš nefunkcinių reikalavimų buvo paprasta ir lengvai supranta vartotojo sąsaja. Šis reikalavimas buvo sėkmingai įgyvendintas. Sistemoje realizuoti tik pagrindiniai funkciniai mygtukai, ji neapkrauta papildomais nereikalingais dalykais, paaiškinimai ir pranešimai yra aiškūs ir tikslūs. Vartotojui norint naudotis programų sistema jokių specialių žinių ar įgūdžių įgyti nereikia, pakanka elementarių naudojimosi kompiuterių žinių.

4.7 Kokybės vertinimo procesas

4.7.1 Interviu su užsakovu

Interviu su užsakovu metu, buvo aptariami reikalingi programų sistemos patobulinimai. Buvo sukurtas reikalingų pakeitimų sąrašas, išskiriant būtinus ir pageidautinus patobulinimus. Dauguma patobulinimų jau įgyvendinti.

Lentelė Nr. 4. Reikalingų patobulinimų sąrašas

Nr.	Reikalingi pataisymai	Būtinas/Pageidautinas	Įgyvendinimas
1.	Draugiški klaidų pranešimai	Būtinas	+
2.	Automatiškai atsinaujinanti klaidos būsena	Būtinas	+
3.	Vartotojo sąsaja vokiečių kalba	Pageidautinas	-
4.	Programuotojų darbo ataskaita	Pageidautinas	+
5.	Testuotojų darbo ataskaita	Pageidautinas	+
6.	Programos ikona Windows meniu juostoje	Pageidautinas	+
7.	Pilnai automatizuota instaliavimo programa	Pageidautinas	-

4.7.2 Formalios techninės peržiūros

Formalios techninės peržiūros metu kvalifikuotų programuotojų grupė skaito programos kodą ir ieško galimų programos kodo defektų. Grupę sudarė iki keturių žmonių grupę: kodo autorius, sekretorius, du programuotojai. Atsitiktinai pasirenkami kodo failai ir projektoriaus pagalba pateikiami visiems peržiūros dalyviams. Kodas ne vykdomas, o normaliu greičiu slenkamas žemyn ir skaitomas. Rasti defektai dokumentuoti ir ištaisyti.

4.8 Programų sistemos tobulinimo galimybės

Nors esminių trūkumų, kuriuos būtina tobulinti, nei interviu su užsakovu, nei funkcionalumo tyrimo metu nerasta, tačiau formalios techninės peržiūros atskleidė, kad programos kodo apimtis gali būti sumažinta panaudojus tobulesnį objektiškai orientuotą duomenų modelį.

1) Dabartinė situacija

   Darbo metu realizuota programų sistema naudoja standartinį .NET duomenų valdymo mechanizmą su duomenų aibėmis (angl. *data set*) kartu su išsaugotomis SQL procedūromis (angl. *stored procedures*), kurias valdo duomenų adapteriai. Papildomai sukurtos pagalbinės klasės (valdikliai), skirti duomenų valdymui ir patikrinimui.

   Naudojantis šiuo šablonu, kiekviena duomenų bazės lentelė turi keturias išsaugotas procedūras (įrašo gavimo, naujo įrašo, redagavimo ir trynimo) duomenų bazėje, plius tris realizuotus objektus, skirtus darbui su duomenimis: duomenų aibės[3], duomenų adapterio[4] ir valdiklio, kurį tenka rašyti rankiniu būdu[5].

   Didžiausia problema iškyla tada, kada reikia padaryti pakeitimus duomenų bazėje (pvz. pervardinti ar pridėti naują stulpelį lentelėje). Tokiu atveju tektų eiti per visas išsaugotas procedūras ir išvardintus objektus bei pasirūpinti, kad padaryti pakeitimai duomenų bazėje atsispindėtų atitinkamuose objektuose. Tokie pakeitimai būtų potencialūs klaidų šaltiniai.

2) Galimybės patobulinti

   Vienas iš būdų efektyviau organizuoti darbą su duomenimis yra objektiškai orientuoto duomenų modelio naudojimas. Šio modelio esmė yra ta, kad darbas su duomenų baze bendru atveju organizuojamas per dinamines SQL užklausas, kurias automatiškai formuoja modelio vidinis mechanizmas. Programuotojui, kaip įrankis valdyti duomenis, pateikiamas objektų rinkinys, atspindintis duomenų bazės schemą (pvz. objektas su savo atributais atspindi vieną lentelės įrašą, o objektų

[3] Duomenų aibė (angl. *data set*) – duomenų bazės atvaizdas .NET mechanizme.
[4] Duomenų adapteris (angl. *data adapter*) numato mechanizmą susieti išsaugotas procedūros su .NET mechanizmu.
[5] Duomenų vaizdo ir duomenų adapterio objektai sugeneruojami automatiškai, naudojantis Visual Studio įrankį.

masyvas visą lentelės turinį) ir papildomi objektai, vykdantys konkrečius veiksmus su duomenimis: nuskaitymą, įterpimą, redagavimą, šalinimą, filtravimą ir pan.

Aišku, rankiniu būdu realizuoti tokį modelį reikėtų daug pastangų ir pats modelis prarastų savo praktiškumą. Tačiau rinkoje šiuo metu yra tiek mokamų, tiek nemokamų įrankių[6], kurie pagal duomenų bazės schemą gali sugeneruoti objektiškai orientuoto duomenų modelio programinį kodą.

Privalumai aiškūs: paprastas duomenų bazės pakeitimų realizavimas (tik pačių pakeitimų įdiegimas ir modelio kodo pergeneravimas), mažiau kodo rašymo rankiniu būdu.

Minusai: reikia papildomai mokytis, norint įsisavinti objektiškai orientuotą duomenų modelį.

Išsamiai šio modelio privalumai ir trūkumai išanalizuoti bei su konkrečiais pavyzdžiais pateikti eksperimentinėje darbo dalyje.

4.9 Išvados

Esminių trūkumų programinė įranga neturi. Sistema suprojektuota pagal įmonės keliamus standartus. Laikytasi OOP ir OOD principų. Gerai pritaikytas MVC projektavimo šablonas. Sudarytos prielaidos ir numatytos galimybės programų sistemos tobulinimui.

[6] Paieška internete pagal raktinius žodžius „O/R mapper generator tool“.

5 EKSPERIMENTINĖ DALIS

5.1 Apžvalga

Objektiškai orientuoto duomenų modelio automatinis kodo generavimo įrankis sugeneruoja programinį kodą su atitinkamais objektais. Jis kartu yra modeliavimo ir kodo generavimo įrankis, kuris prisijungia prie duomenų bazės ir nuskaitęs duomenų bazės schemą leidžia susieti objektus su duomenų bazės lentelėmis bei vaizdais (angl. *view*), nustatyti vienos eilutės įterpimo, atnaujinimo, nuskaitymo ar trynimo operacijas, užklausas ar išsaugotas procedūras (angl. s*tored procedures*), kaip atitinkamų objektų metodus. Pagal ryšius, nuskaitytus iš duomenų bazės schemos, sudaromi ryšiai tarp objektų, atitinkantys ryšius tarp lentelių. Sugeneruojamas pilnai veikiantis geriausias objektinio projektavimo praktikas atitinkantis kodas [4].

5.2 Objektiškai orientuoto duomenų modelio privalumai

5.2.1 Sutrumpinamas programinės įrangos kūrimo laikas

Didžiausias objektiškai orientuoto duomenų modelio privalumas yra tas, kad jis taupo programinės įrangos kūrimo laiką (nuo 20% iki 50% priklausomai nuo kuriamos sistemos)[4].

Matematika paprasta. Tipinė programa su 15-20 lentelių turi 30-50 objektų (įskaitant srities (angl. *domain*) ir gamybinius (angl. *factory*) objektus). Tai apytiksliai nuo 5000 iki 10.000 kodo eilučių. Vienam programuotojui parašyti ir ištestuoti tokį kiekį kodo užtruktų nuo kelių savaičių iki keleto mėnesių. Jeigu programinė įranga dirba su dar daugiau lentelių (o daugelis taip daro), šie skaičiai dar padidėja [4].

Naudojant automatinį generavimo įrankį, visas objektiškai orientuoto duomenų modelio kodas bus gautas per keletą dienų. Tiek laiko truks nustatyti ir apsispręsti dėl duomenų bazės struktūros. Pats generavimas vykdomas akimirksniu.

5.2.2 Gaunamas geriau suprojektuotas kodas

Antras objektiškai orientuoto duomenų modelio privalumas yra tas, kad automatinis kodo generavimo įrankis sukurs geriau suprojektuotą kodą, nei tai padarytų programuotojų komanda, kodą rašydama pati. Gal būti, kad kas nors nesutiks su šiuo teiginiu. Tačiau reikia atsiminti, kad ne kiekvienas programuotojas kartu yra geras projektuotojas. Rezultate tikriausiai nebus panaudoti tinkami projektavimo šablonai (angl. *design pattern*), o parašytas kodas bus blogesnės kokybės. Jeigu duomenų sluoksnį kurs keli programuotojai, jų kodas tarpusavyje gali būti sunkiai suderinamas ir rezultate bus gautas didelės apimties abejotinos kokybės programinis kodas [4].

Naudojantis automatiniu generavimo įrankiu gaunamas puikiai suprojektuotas kodas, nes šablonai, kuriuos naudoja generatorius, yra sukurti aukšto lygio programuotojų. Šie šablonai beveik visada laikosi nustatytų projektavimo šablonų (angl. *design pattern*). Taigi, kodas,

gautas naudojant automatinį generavimo įrankį, tikriausiai bus geresnės kokybės, nei kodas, sukurtas programuotojų komandos rankiniu būdu [4].

5.2.3 Nebūtina būti .NET ekspertu

Duomenų apdorojimo programinis kodas yra kritinė sritis ir nulemia visos sistemos charakteristikas. Jei šis kodas bus blogai suprojektuotas ir naudojamas neefektyviai, tai gali atitinkamai paveikti visą sistemą. Taigi, jeigu duomenų apdorojimo programinis kodas yra rašomas rankiniu būdu, programuotojas privalo būti .NET, COM+, nepriklausomų komponentų ir MTS ekspertu [4].

Jeigu naudojamas automatiškai sugeneruotas objektiškai orientuotas duomenų modelis, programuotojai gali susikoncentruoti tik į loginį objektų projektavimą, įskaitant objektų susiejimą, įvairių sąveikų su duomenų baze, įskaitant įterpimą, atnaujinimą, užkrovimą, trynimą ir t.t., realizaciją. Viskuo likusiu pasirūpins objektiškai orientuoto duomenų modelio vidinis mechanizmas [4].

5.2.4 Taupomas testavimo laikas

Automatiškai sugeneruoto objektiškai orientuoto duomenų modelio naudojimas dramatiškai sumažina testavimo laiką. Jeigu duomenų modelis sukurtas rankiniu būdu, patys modelio kūrėjai bus atsakingi už jo testavimą.

Naudojant automatinį generavimo įrankį, kodo šablonai jau bus nuosekliai ištestuoti pačių įrankio kūrėjų. Dar daugiau. Tikriausiai šimtai ar tūkstančiai vartotojų jau naudojasi šiuo įrankiu ir yra puikiai ištestavę automatinį kodo generatorių. Rezultate daugelis defektų, su kuriais galbūt net nesusidurtumėte testavimą atlikdami patys, jau yra rasti ir ištaisyti [4].

5.2.5 Supaprastinamas programavimas

Naudojantis standartiniais duomenų valdymo modeliais, programuotojai turi gilintis į ADO.NET, COM+, MTS technologijas ir nagrinėti nepriklausomus komponentus. Taip daug laiko sugaištama vietose, kur visai nereikia.

Naudojantis objektiškai orientuotu duomenų modeliu, dirbama su gerai suprojektuotais objektais ir publikuotomis objektų sąsajomis (angl. *public interfaces*).

Trumpas pavyzdys, kuris pademonstruoja, kaip iš duomenų bazės lentelės *Klientai*, gaunamas atitinkamas įrašas su unikaliu numeriu *17*.

```
try
{
     KlientaiEntity klientas = new KlientaiEntity();
     KlientaiFactory klientasFactory = new KlientaiFactory ();

     klientas.KlientasID = 17;
     klientasFactory.Load(klientas);

     // Toliau jau galima naudoti objektą klientą. Visi atributai užpildyti.
}
catch(Exception ex)
{
     // Kodas įvykus nenumatytai klaidai.
     return;
}
```

Iš pateikto pavyzdžio matosi, kad objektiškai orientuoto duomenų modelio sugeneruotas kodas labai lengvai panaudojamas.

5.3 Populiariausių įrankių privalumai ir trūkumai

Testavimui ir bandymams pasirinkti sekantys objektiškai orientuoto duomenų modelio automatinio generavimo įrankiai[7]:

- LLBLGenPro v1.0 – mokamas. Naudoja užsakovo įmonė, taigi puikios galimybės gauti profesionalias konsultacijas ir pilną įrankio versiją bandymams.
- NHibernate v0.95 – nemokamas populiarus produktas.
- ORM.NET v1.7 – dar vienas populiarus nemokamas įrankis.

Bandymų metu stengtasi nustatyti, kokiomis svarbiausiomis savybėmis kiekvienas įrankis pasižymi/nepasižymi.

5.3.1 Integruotas pagalbos vadovas

Tai tikriausiai pati svarbiausia savybė, nes prieš naudojantis konkrečiu įrankiu, reikia jį išmokti.

- LLBLGenPro v1.0 – turi.
- NHibernate v0.95 – turi.
- ORM.NET v1.7 – turi.

5.3.2 Nuolatinis palaikymas

Susidūrus su nesklandumais ar defektu, tampa labai svarbia savybe. Taip pat galima žiūrėti kaip antrinį požymį, rodantį ar įrankis vis dar „gyvas“.

- LLBLGenPro v1.0 – turi.
- NHibernate v0.95 – turi.
- ORM.NET v1.7 – turi.

5.3.3 Objektiškai orientuotos užklausos

Objektiškai orientuoto duomenų modelio savybė, apsprendžianti galimybę dinamiškai iš kodo sudarinėti užklausas, tiesiogiai nerašant SQL kodo.

- LLBLGenPro v1.0 – taip.
- NHibernate v0.95 – taip.
- ORM.NET v1.7 – ne.

5.3.4 Kelių DBVS palaikymas

Galimybė su tuo pačiu įrankiu sugeneruoti kodą įvairioms DBVS.

- LLBLGenPro v1.0 – taip.
- NHibernate v0.95 – taip.
- ORM.NET v1.7 – taip.

[7] Šiuos įrankius rekomenduoja programavimo forumų dalyviai.

5.3.5 Duomenų susiejimo savybė

Ar duomenys gali būti tiesiogiai susieti su standartiniais vartotojo sąsajos komponentais?

- LLBLGenPro v1.0 – yra.
- NHibernate v0.95 – nėra.
- ORM.NET v1.7 – nėra.

5.3.6 Išsaugotų procedūrų iškvietimai

Ar palaikoma galimybė tiesiogiai iškvietinėti išsaugotas procedūras (angl. s*tored procedures*).

- LLBLGenPro v1.0 – palaikomi.
- NHibernate v0.95 – nepalaikomi.
- ORM.NET v1.7 – palaikomi.

5.3.7 Grafinis objektų ir atributų susiejimo įrankis

Ar įrankis turi grafinę sąsają, kurios pagalba galima atlikti visas manipuliacijas su modeliu ir atributų susiejimu.

- LLBLGenPro v1.0 – yra.
- NHibernate v0.95 – nėra. Susiejimas aprašomas rankiniu būdu per XML failus.
- ORM.NET v1.7 – yra.

5.3.8 Tingaus duomenų užkrovimo režimas

Tai režimas, kai duomenys iš duomenų bazės pilnai užkraunami tik tada, kai jų iš tiesų reikia.

- LLBLGenPro v1.0 – palaikomas.
- NHibernate v0.95 – palaikomas.
- ORM.NET v1.7 – nepalaikomas.

5.3.9 Duomenų skaidymas į puslapius serverio pusėje

Ypač aktualu kuriant internetines sistemas. Savybė nurodo, ar duomenų modelis gali atlikti duomenų suskaidymą į puslapius dar serverio pusėje.

- LLBLGenPro v1.0 – taip.
- NHibernate v0.95 – ne.
- ORM.NET v1.7 – ne.

5.3.10 Duomenų „kešavimas“

Galimybė naudoti tarpinė atmintį (angl. caching) su tikslu pagerinti laikines duomenų modelio charakteristikas.

- LLBLGenPro v1.0 – palaikomas.
- NHibernate v0.95 – palaikomas.
- ORM.NET v1.7 – nepalaikomas.

5.4 LLBLGenPro įrankio sugeneruoto objektiškai orientuoto duomenų modelio laikinių charakteristikų palyginimas su standartinių duomenų modelių laikinėmis charakteristikomis

Eksperimento tikslas nustatyti laikines įvairių duomenų modelių charakteristikas ir jas tarpusavyje palyginti. Tam sukurta programinė įranga, kuri kviečia atitinkamų modelių funkcijas ir pateikia jų vykdymo laiką. Kad rezultatai būtų tikslesni, kiekvienas bandymas kartojamas 20 kartų. Vertinant į sugaištą laiką neįtraukti paruošiamieji programos darbai – tai tik konkrečios funkcijos vykdymo laikas.

Vertinant laikines charakteristikas dėmesį reikia atkreipti į tai, kad testų rezultatai skirsis skirtingose platformose. Tačiau nustatyti konkrečias laikines charakteristikas šiuo eksperimentu nebuvo siekiama. Eksperimento tikslas įvertinti, kuris duomenų modelis yra pranašesnis/silpnesnis. Bandymams pasirinkti (a) objektiškai orientuotas duomenų modelis, (b) duomenų aibe (angl. *DataSet*) paremtas duomenų modelis ir (c) išsaugotomis procedūromis (angl. *stored procedures*) paremtas duomenų modelis.

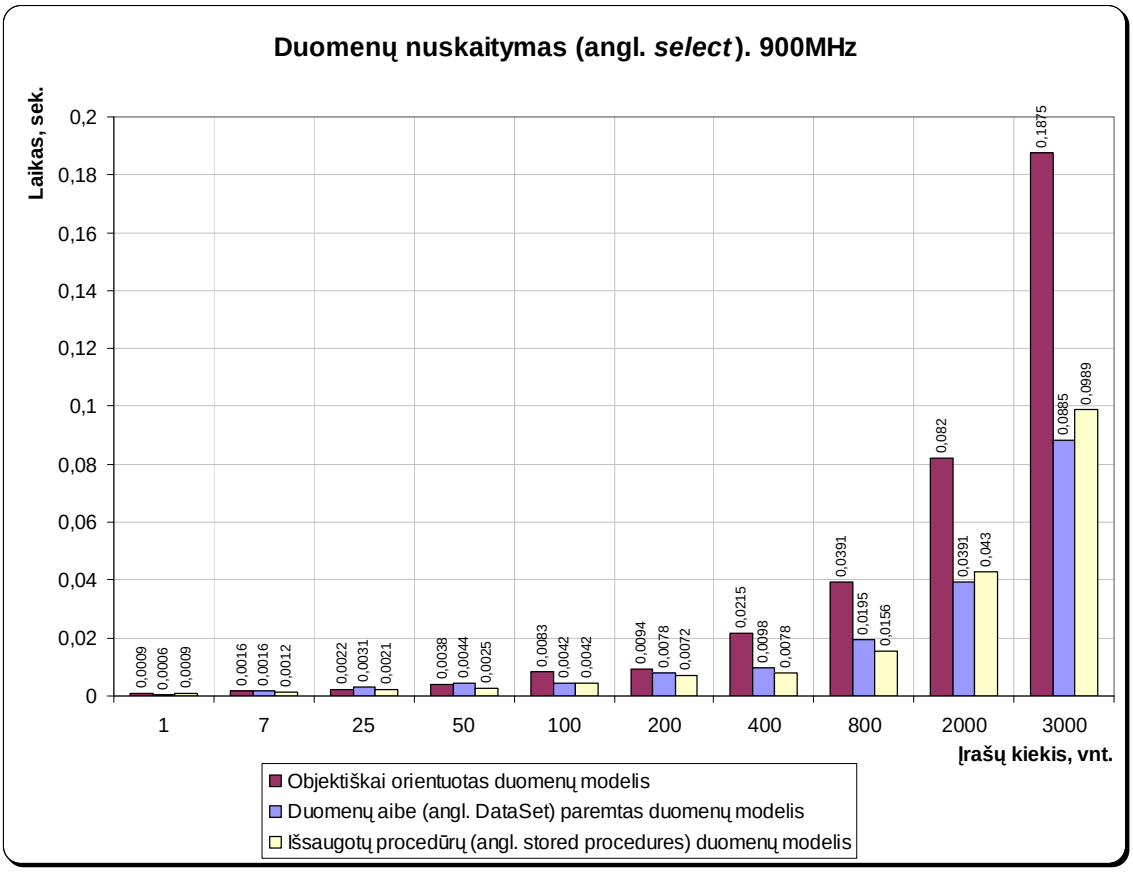
Bandymai atlikti keliose platformose:

I variantas: 900MHz taktinio dažnio procesorius ir 512MB operatyvinės atminties.

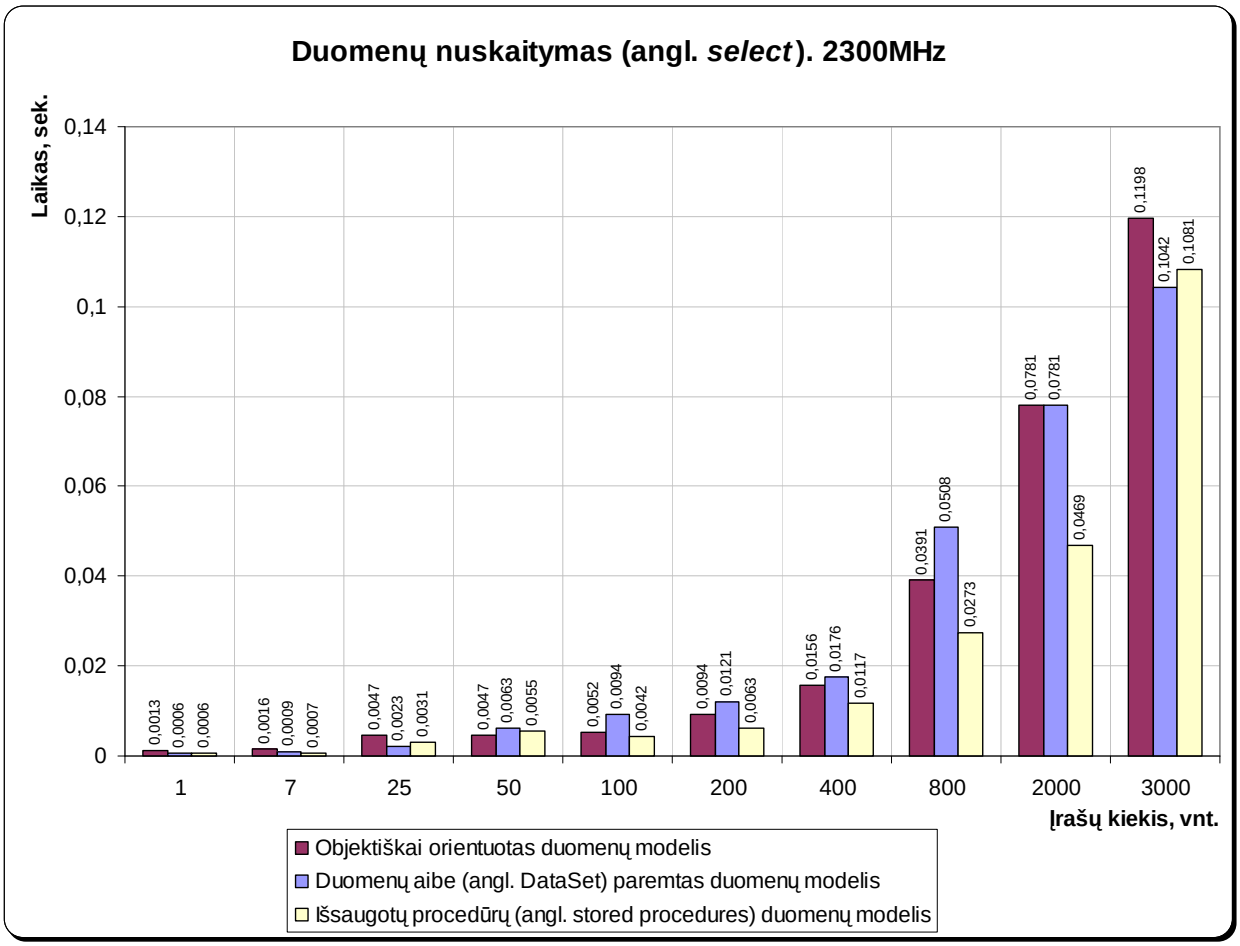
II variantas. 2.3GHz taktinio dažnio procesorius ir 1024MB operatyvinės atminties.

Abiejose platformose bandymai atlikti su MS Sql Server 2005 DBVS. Išsamūs rezultatai pateikti prieduose.

5.4.1 Duomenų nuskaitymo sparta



22 pav. Duomenų nuskaitymo charakteristikos. I variantas.

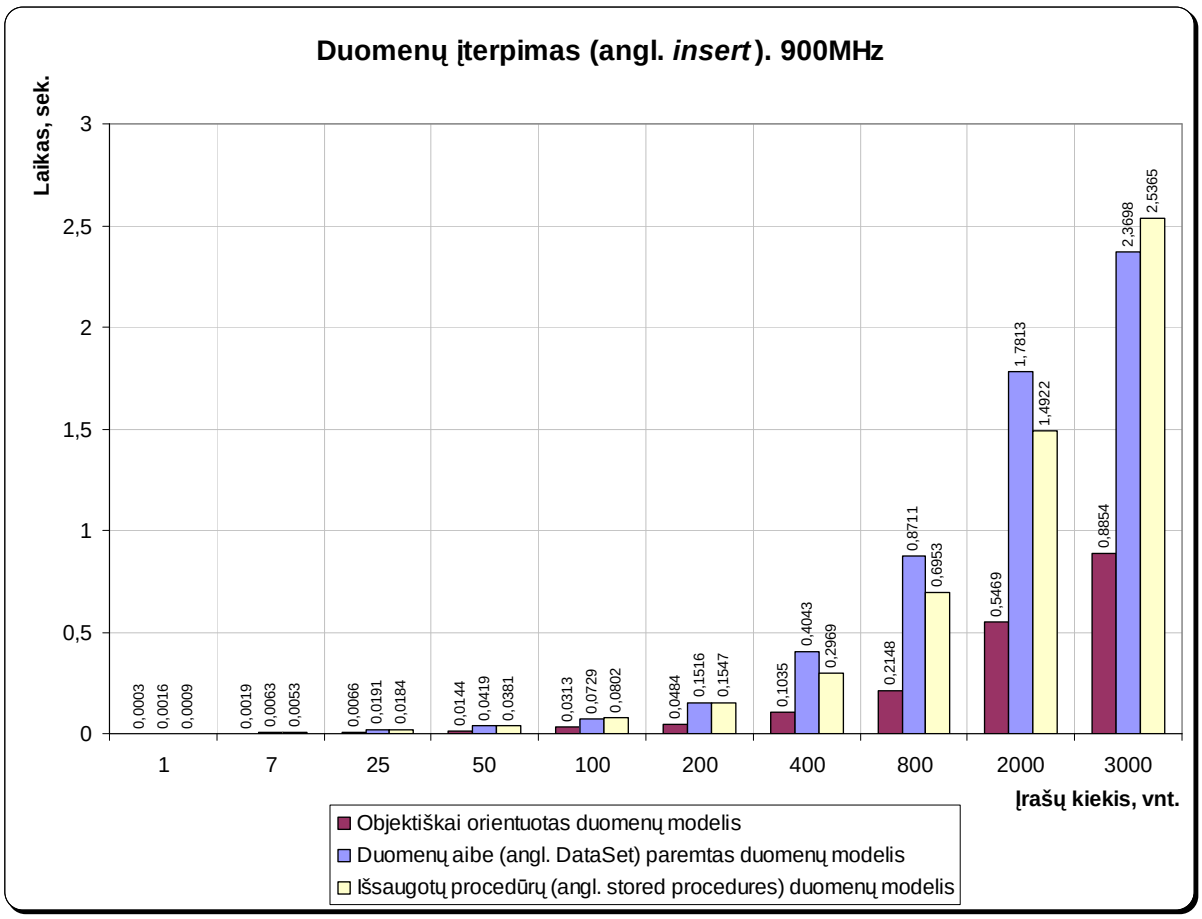


23 pav. Duomenų nuskaitymo charakteristikos. II variantas.

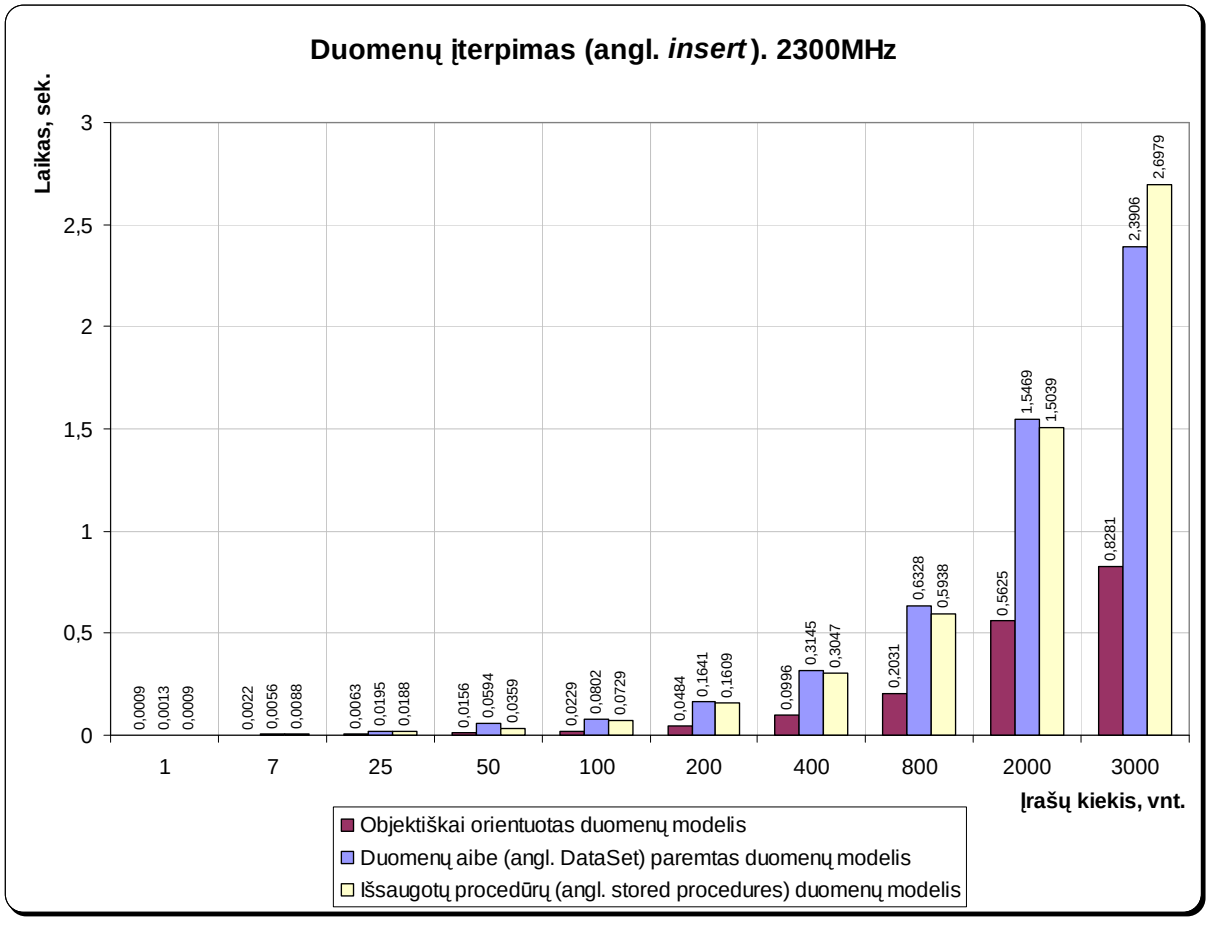
Kaip matome iš grafikų, duomenų nuskaitymas su objektiškai orientuotu duomenų moduliu yra lėtesnis lyginant su likusiais duomenų modeliais. Tai galima būtų paaiškinti tuo, kad kiekvienai nuskaitytai duomenų eilutei yra sukuriamas atskiras objektas.

Skirtumas ypač išryškėja prie didelių duomenų masyvų arba su lėtesne techninę įranga (I variantas).

5.4.2 *Duomenų įterpimo sparta*

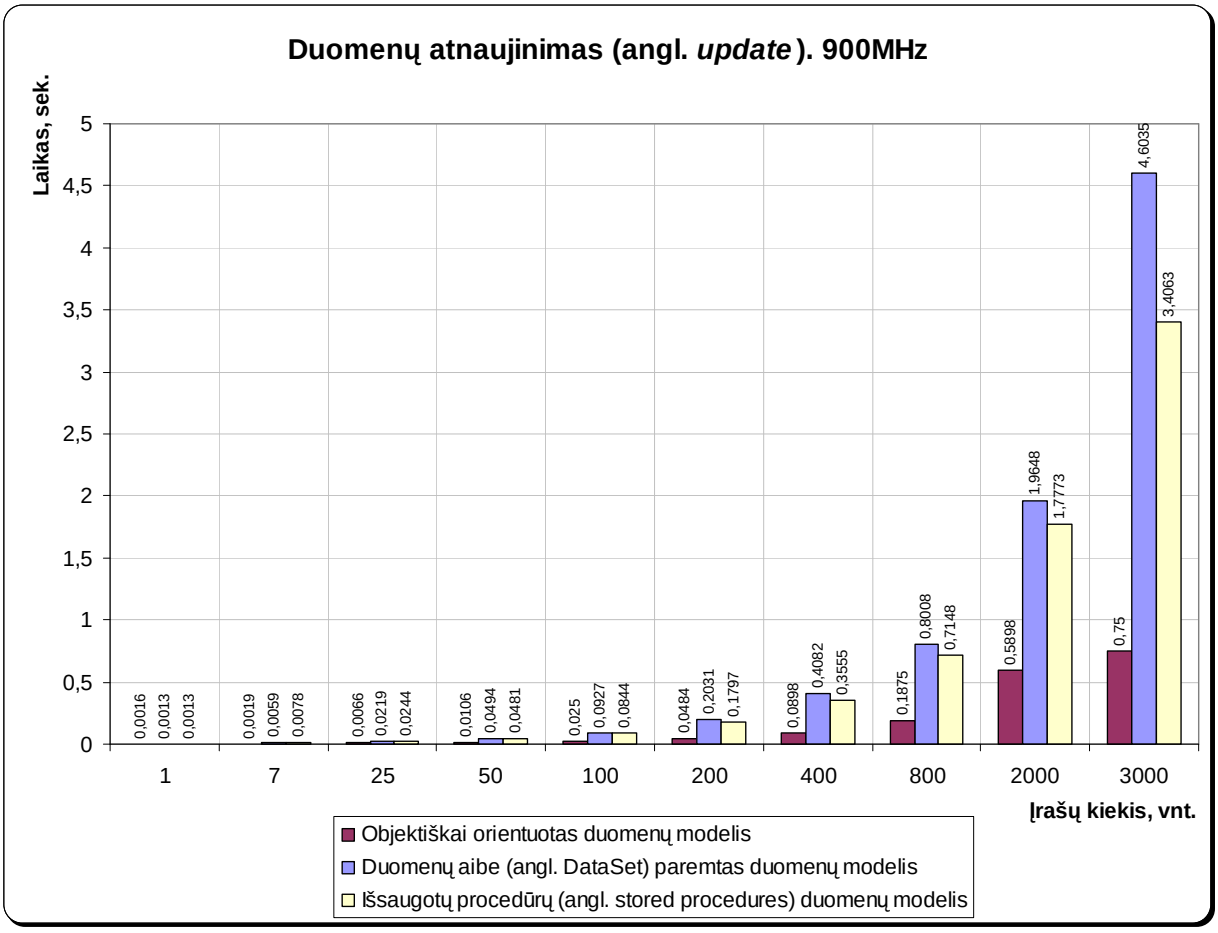


24 pav. Duomenų įterpimo charakteristikos. I variantas.

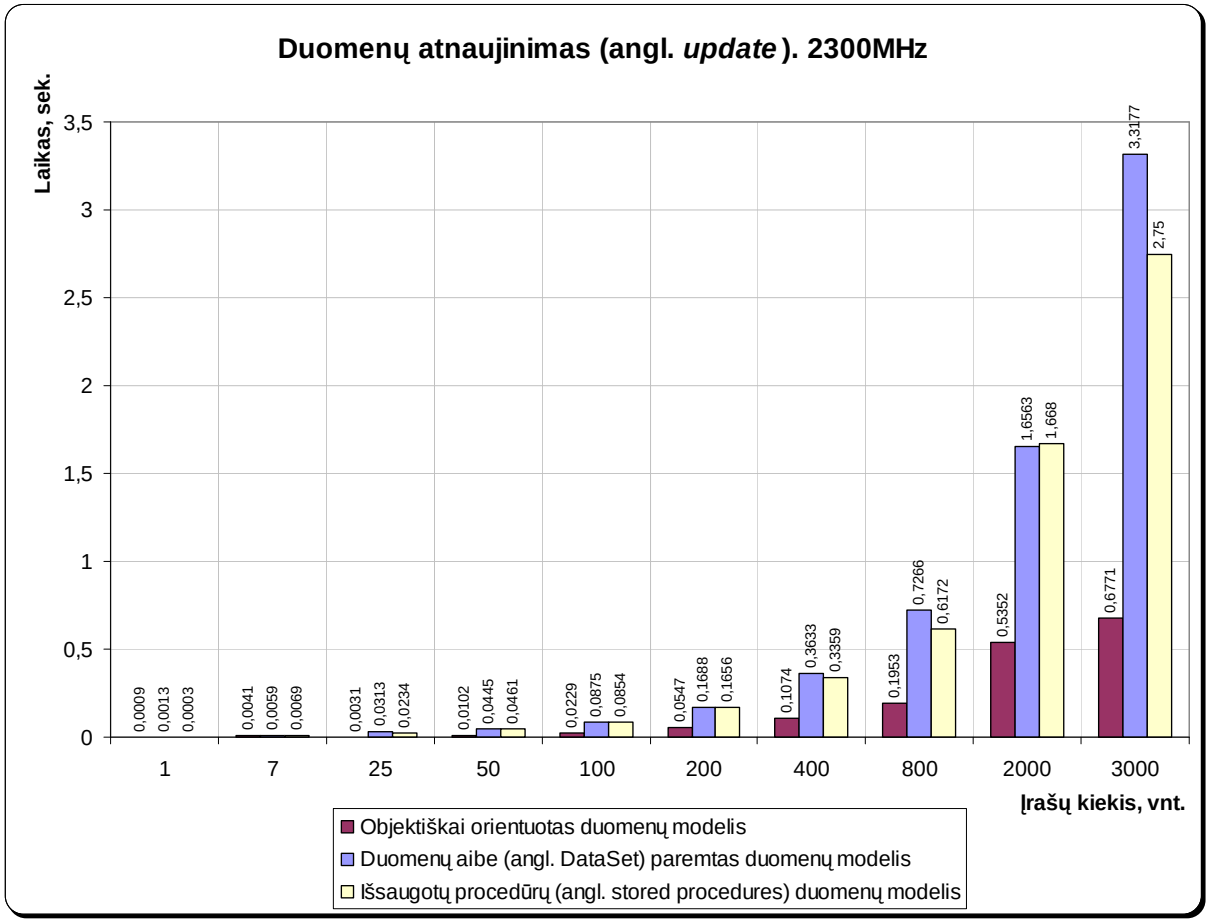


25 pav. Duomenų įterpimo charakteristikos. II variantas.

5.4.3 *Duomenų atnaujinimo sparta*

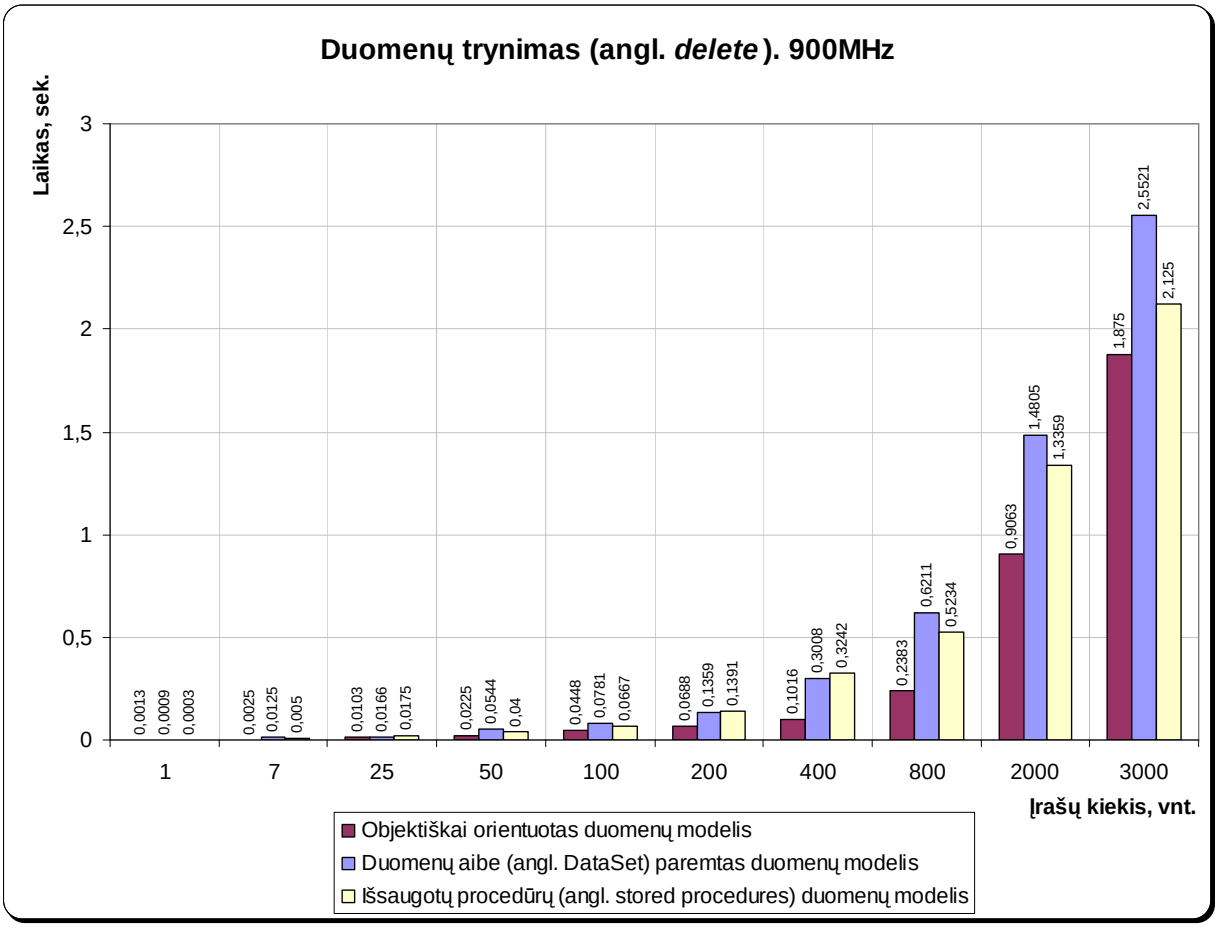


26 pav. Duomenų atnaujinimo charakteristikos. I variantas.

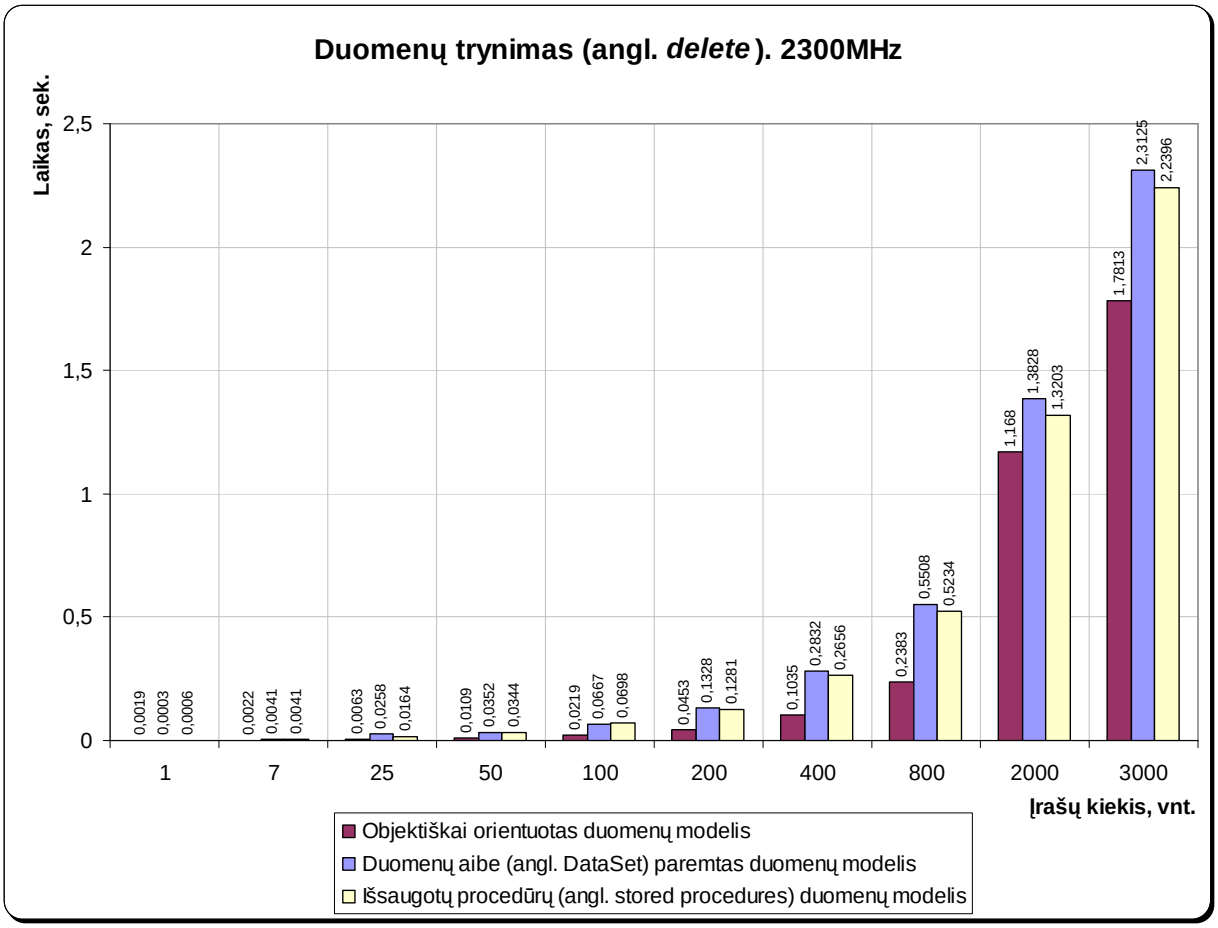


27 pav. Duomenų atnaujinimo charakteristikos. II variantas.

5.4.4 *Duomenų trynimo sparta*



28 pav. Duomenų trynimo charakteristikos. I variantas.



29 pav. Duomenų trynimo charakteristikos. II variantas.

5.5 Eksperimento rezultatai

Įvertinus gautus eksperimento rezultatus galima teigti, kad esant mažiems duomenų kiekiams, laikinių charakteristikų atžvilgiu nėra skirtumo, kokį duomenų apdorojimo modelį naudoti. Esant didesniems duomenų masyvams, kuriuos reikia apdoroti vienu kartu[8], išryškėja skirtumai tarp modelių.

Nagrinėtas objektiškai orientuotas duomenų modelis silpniausiai pasirodė nuskaitant duomenis. Tai būtų galima paaiškinti tuo, kad nuskaitant duomenis, kiekvienai nuskaitytai duomenų lentelės eilutei yra sukuriamas atskiras objektas.

Tačiau nagrinėjamas modelis buvo kelis kartus spartesnis įterpiant arba atnaujinant duomenų bazės įrašus. Trynimo funkciją visi modeliai vykdė panašiu greičiu.

Iš grafikų matome, kad platforma, kurioje vykdomas eksperimentas, neturi esminės įtakos rezultatams ir pačių faktų nekeičia.

Realizuotos programinės įrangos tobulinimo atžvilgiu, objektiškai orientuoto duomenų modelio panaudojimas laikinių charakteristikų ženkliai nepakeistų. Sukurta programinė įranga nedirba su dideliais duomenų masyvais, todėl laikinės charakteristikos būtų artimos dabartinėms.

Tačiau atsižvelgiant į tai, kad ateityje gali tekti programinę įrangą papildyti ir keisti (o tai tikrai būtų susiję su duomenų bazės pakeitimais), verta pereiti prie objektiškai orientuoto duomenų modelio. Perėjimas prie minėto duomenų modelio papildomai pagerintų kodo objektiškumą, būtų aiškesnės MVC projektavimo šablono nubrėžtos ribos.

[8] Tai duomenų aibė, kurią vienu kartu reikia nuskaityti, įterpti ar pašalinti iš duomenų bazės.

6 IŠVADOS

Apžvelgus atliktą darbą, padarytos tokios išvados:

- ✓ Iš užsakovo surinkti programinės įrangos reikalavimai. Atlikta sistemos analogų analizė. Programinė įranga suprojektuota, realizuota ir ištestuota.
- ✓ Programinė įranga sukurta remiantis MVC projektavimo šablonu, naudojant C# kalbą ir .NET Framework 2.0.
- ✓ Šiuo metu realizuota sistema sėkmingai naudojama užsakovo įmonėje[9].
- ✓ Atliekant programinės įrangos kokybės vertinimą ir atliekant formalias kodo peržiūras nustatyta, kad programinės įrangos kodo struktūra gali būti patobulinta panaudojus objektiškai orientuotą duomenų modelį. Tokiu būdu būtų sumažinta kodo apimtis ir pagerintas programos objektiškumas.
- ✓ Surinkta informacija apie objektiškai orientuotą duomenų modelį ir jo kodo automatinius generavimo įrankius.
- ✓ Atliktas eksperimentinis duomenų modelių palyginimas. Nustatyti esminiai modelių privalumai ir trūkumai, įvertintos laikinės charakteristikos.
- ✓ Atlikti bandymai su duomenų modeliais patvirtino objektiškai orientuoto duomenų modelio privalumus. Nustatytas minusas: lėtesnės, lyginant su kitais duomenų modeliais, duomenų nuskaitymo funkcija.
- ✓ Numatytos programinės įrangos tobulinimų galimybės.

[9] Priede A pateiktas programinės įrangos diegimo aktas.

7 LITERATŪRA

1 KAPFHAMMER, G. M. *Software Testing*. Meadville, 2004. 1-44p.

2 WHITTAKER, A. J. Stochastic Software Testing. *Annals of Software Engineering*, 1997, Nr. 4, p. 115-131.

3 MACKIENĖ, J. *PĮ defektų valdymas* [interaktyvus]. 2005 [žiūrėta 2007-05-11]. Prieiga per internetą: <http://www.bpi.lt/text.php?lang=1&item=224&arg=202>.

4 KHAN, I. M. *Five Reasons for using an O/R Mapping Tool* [interaktyvus]. 2005 [žiūrėta 2007-05-12]. Prieiga per internetą: <http://www.dnzone.com/ShowDetail.asp?NewsId=1276>

5 RICHTER, J. *Applied Microsoft .NET Framework Programming.* New Jersey, 2002. 626p.

6 WALSH, N. *A Technical Introduction to XML* [interaktyvus]. 1998 [žiūrėta 2007-03-18]. Prieiga per internetą: http://www.xml.com/pub/a/98/10/guide0.html

7 MARGUERIE, F. *Choosing an object-relational mapping tool* [interaktyvus]. 2005 [žiūrėta 2007-05-07]. Prieiga per internetą: http://madgeek.com/Articles/ORMapping/EN/mapping.htm

8 BALLARD, P. *Ask TheServerSide: Which .NET ORM is best?*2004 [žiūrėta 2007-05-07]. Prieiga per internetą: http://www.theserverside.net/news/thread.tss?thread_id=29914

9 KULAK, D.; and GUINEY E. *Use Cases: Requirements in Context, Second Edition.* Addison Wesley, 2003. 272p.

10 *Code Project for C#* [interaktyvus]. [žiūrėta 2006-10-12]. Prieiga per internetą: http://www.codeproject.com/c#

11 *Code Project for ASP.NET* [interaktyvus]. [žiūrėta 2006-10-12]. Prieiga per internetą: http://www.codeproject.com/asp.net

12 *Bug tracking system BugLister* [interaktyvus]. [žiūrėta 2006 10 14]. Prieiga per internetą: http://www.litwindow.com/Products/BugLister/overview_buglister.html

13 *Bug tracking system PR Tracker* [interaktyvus]. [žiūrėta 2006 10 16]. Prieiga per internetą: http://www.prtracker.com/

14 *Bug tracking system Bugzilla* [interaktyvus]. [žiūrėta 2006 10 19]. Prieiga per internetą: http://www.bugzilla.org/

15 *Seapine Software TestTrack* [interaktyvus]. [žiūrėta 2006 11 18]. Prieiga per internetą: http://www.seapine.com/ttstudio.html

8 TERMINŲ IR SANTRUMPŲ ŽODYNAS

PĮPS	–	**p**rogramin**ė**s **į**rangos **p**alaikymo **s**istema.
ADO.NET	–	standartinis Microsoft duomenų valdymo modelis.
COM+	–	komponentinių objektų modelis (angl. ***c****omponent* ***o****bject* ***m****odel*).
DBVS	–	duomenų bazės valdymo sistema.
OS	–	operacinė sistema (angl. ***o****perating* ***s****ystem*).
API	–	programavimo sąsaja (angl. ***a****pplication* ***p****rogramming* ***i****nterface*).
Perl	–	bendros paskirties programavimo kalba.
MySql	–	reliacinių duomenų bazių apdorojimo sistema.
CGI	-	technologija, naudojama interneto serveriuose (angl. ***c****ommon* ***g****ateway* ***i****nterface*).
MS SQL Server	–	reliacinių duomenų bazių apdorojimo sistema.
IIS	–	internetinių puslapių publikavimo serveris (angl. *internet information services).*
UML	–	tarptautinis objektiškai orientuotos analizės ir projektavimo standartas dokumentacijai (angl. ***u****nified* ***m****odeling* ***l****anguage*).
MVC	–	objektinio programavimo šablonas (angl. ***m****odel-****v****iew-****c****ontroller design pattern*).
MTS	–	komponentas, kuris leidžia kitomis programoms nesunkiai palaikyti transakcijas (angl. ***M****icrosoft* ***t****ransaction* ***s****erver*).
XML	–	universalus hierarchinis duomenų aprašymo formatas (angl. e**x**tensible **m**arkup **l**anguage).

9 PRIEDAS A. PROGRAMINĖS ĮRANGOS DIEGIMO AKTAS

UŽDAROJI AKCINĖ BENDROVĖ
“INDIVIDUALŪS SPRENDIMAI”
Kodas 135921360, PVM mokėtojo kodas LT359213610, Neries krantinė 16, LT-48402 Kaunas,
Tel. 8-700-55223

Kauno technologijos universitetui

DĖL PAULIAUS MAČIULIO PRAKTIKOS METU SUKURTOS PROGRAMOS

Studentas Pauliaus Mačiulio praktikos metu sukūrė programinę įrangą pavadinimu „PROGRAMINĖS ĮRANGOS PALAIKYMO SISTEMA“, kuri šiuo metu yra įdiegta ir naudojama įmonės vidiniams tikslams.

Direktorius 

Kęstutis Ramonaitis

10 PRIEDAS B. DUOMENŲ MODELIŲ LAIKINĖS CHARAKTERISTIKOS

Lentelė Nr. 5. Duomenų modelių laikinės charakteristikos. I variantas.[10]

Objektiškai orientuotas duomenų modelis	**Įrašų kiekis, vnt.**	**Vid. laikas, sek.**	**Duomenų aibe (angl. *DataSet*) paremtas duomenų modelis**	**Įrašų kiekis, vnt.**	**Vid. laikas, sek.**	**Išsaugotų procedūrų (angl. *stored procedures*) duomenų modelis**	**Įrašų kiekis, vnt.**	**Vid. laikas, sek.**
Nuskaitymas	1	0,0009	Nuskaitymas	1	0,0006	Nuskaitymas	1	0,0009
Nuskaitymas	7	0,0016	Nuskaitymas	7	0,0016	Nuskaitymas	7	0,0012
Nuskaitymas	25	0,0022	Nuskaitymas	25	0,0031	Nuskaitymas	25	0,0021
Nuskaitymas	50	0,0038	Nuskaitymas	50	0,0044	Nuskaitymas	50	0,0025
Nuskaitymas	100	0,0083	Nuskaitymas	100	0,0042	Nuskaitymas	100	0,0042
Nuskaitymas	200	0,0094	Nuskaitymas	200	0,0078	Nuskaitymas	200	0,0072
Nuskaitymas	400	0,0215	Nuskaitymas	400	0,0098	Nuskaitymas	400	0,0078
Nuskaitymas	800	0,0391	Nuskaitymas	800	0,0195	Nuskaitymas	800	0,0156
Nuskaitymas	2000	0,082	Nuskaitymas	2000	0,0391	Nuskaitymas	2000	0,043
Nuskaitymas	3000	0,1875	Nuskaitymas	3000	0,0885	Nuskaitymas	3000	0,0989
Įterpimas	1	0,0003	Įterpimas	1	0,0016	Įterpimas	1	0,0009
Įterpimas	7	0,0019	Įterpimas	7	0,0063	Įterpimas	7	0,0053
Įterpimas	25	0,0066	Įterpimas	25	0,0191	Įterpimas	25	0,0184
Įterpimas	50	0,0144	Įterpimas	50	0,0419	Įterpimas	50	0,0381
Įterpimas	100	0,0313	Įterpimas	100	0,0729	Įterpimas	100	0,0802
Įterpimas	200	0,0484	Įterpimas	200	0,1516	Įterpimas	200	0,1547
Įterpimas	400	0,1035	Įterpimas	400	0,4043	Įterpimas	400	0,2969
Įterpimas	800	0,2148	Įterpimas	800	0,8711	Įterpimas	800	0,6953
Įterpimas	2000	0,5469	Įterpimas	2000	1,7813	Įterpimas	2000	1,4922
Įterpimas	3000	0,8854	Įterpimas	3000	2,3698	Įterpimas	3000	2,5365
Atnaujinimas	1	0,0016	Atnaujinimas	1	0,0013	Atnaujinimas	1	0,0013
Atnaujinimas	7	0,0019	Atnaujinimas	7	0,0059	Atnaujinimas	7	0,0078
Atnaujinimas	25	0,0066	Atnaujinimas	25	0,0219	Atnaujinimas	25	0,0244
Atnaujinimas	50	0,0106	Atnaujinimas	50	0,0494	Atnaujinimas	50	0,0481
Atnaujinimas	100	0,025	Atnaujinimas	100	0,0927	Atnaujinimas	100	0,0844
Atnaujinimas	200	0,0484	Atnaujinimas	200	0,2031	Atnaujinimas	200	0,1797
Atnaujinimas	400	0,0898	Atnaujinimas	400	0,4082	Atnaujinimas	400	0,3555
Atnaujinimas	800	0,1875	Atnaujinimas	800	0,8008	Atnaujinimas	800	0,7148
Atnaujinimas	2000	0,5898	Atnaujinimas	2000	1,9648	Atnaujinimas	2000	1,7773
Atnaujinimas	3000	0,75	Atnaujinimas	3000	4,6035	Atnaujinimas	3000	3,4063
Pašalinimas	1	0,0013	Pašalinimas	1	0,0009	Pašalinimas	1	0,0003
Pašalinimas	7	0,0025	Pašalinimas	7	0,0125	Pašalinimas	7	0,005
Pašalinimas	25	0,0103	Pašalinimas	25	0,0166	Pašalinimas	25	0,0175
Pašalinimas	50	0,0225	Pašalinimas	50	0,0544	Pašalinimas	50	0,04
Pašalinimas	100	0,0448	Pašalinimas	100	0,0781	Pašalinimas	100	0,0667
Pašalinimas	200	0,0688	Pašalinimas	200	0,1359	Pašalinimas	200	0,1391
Pašalinimas	400	0,1016	Pašalinimas	400	0,3008	Pašalinimas	400	0,3242
Pašalinimas	800	0,2383	Pašalinimas	800	0,6211	Pašalinimas	800	0,5234
Pašalinimas	2000	0,9063	Pašalinimas	2000	1,4805	Pašalinimas	2000	1,3359
Pašalinimas	3000	1,875	Pašalinimas	3000	2,5521	Pašalinimas	3000	2,125

[10] 900MHz taktinio dažnio procesorius ir 512MB operatyvinės atminties.

Lentelė Nr. 6. Duomenų modelių laikinės charakteristikos. II variantas.[11]

Objektiškai orientuotas duomenų modelis	Įrašų kiekis, vnt.	Vid. laikas, sek.
Nuskaitymas	1	0,0013
Nuskaitymas	7	0,0016
Nuskaitymas	25	0,0047
Nuskaitymas	50	0,0047
Nuskaitymas	100	0,0052
Nuskaitymas	200	0,0094
Nuskaitymas	400	0,0156
Nuskaitymas	800	0,0391
Nuskaitymas	2000	0,0781
Nuskaitymas	3000	0,1198
Įterpimas	1	0,0009
Įterpimas	7	0,0022
Įterpimas	25	0,0063
Įterpimas	50	0,0156
Įterpimas	100	0,0229
Įterpimas	200	0,0484
Įterpimas	400	0,0996
Įterpimas	800	0,2031
Įterpimas	2000	0,5625
Įterpimas	3000	0,8281
Atnaujinimas	1	0,0009
Atnaujinimas	7	0,0041
Atnaujinimas	25	0,0031
Atnaujinimas	50	0,0102
Atnaujinimas	100	0,0229
Atnaujinimas	200	0,0547
Atnaujinimas	400	0,1074
Atnaujinimas	800	0,1953
Atnaujinimas	2000	0,5352
Atnaujinimas	3000	0,6771
Pašalinimas	1	0,0019
Pašalinimas	7	0,0022
Pašalinimas	25	0,0063
Pašalinimas	50	0,0109
Pašalinimas	100	0,0219
Pašalinimas	200	0,0453
Pašalinimas	400	0,1035
Pašalinimas	800	0,2383
Pašalinimas	2000	1,168
Pašalinimas	3000	1,7813

Duomenų aibe (angl. *DataSet*) paremtas duomenų modelis	Įrašų kiekis, vnt.	Vid. laikas, sec.
Nuskaitymas	1	0,0006
Nuskaitymas	7	0,0009
Nuskaitymas	25	0,0023
Nuskaitymas	50	0,0063
Nuskaitymas	100	0,0094
Nuskaitymas	200	0,0121
Nuskaitymas	400	0,0176
Nuskaitymas	800	0,0508
Nuskaitymas	2000	0,0781
Nuskaitymas	3000	0,1042
Įterpimas	1	0,0013
Įterpimas	7	0,0056
Įterpimas	25	0,0195
Įterpimas	50	0,0594
Įterpimas	100	0,0802
Įterpimas	200	0,1641
Įterpimas	400	0,3145
Įterpimas	800	0,6328
Įterpimas	2000	1,5469
Įterpimas	3000	2,3906
Atnaujinimas	1	0,0013
Atnaujinimas	7	0,0059
Atnaujinimas	25	0,0313
Atnaujinimas	50	0,0445
Atnaujinimas	100	0,0875
Atnaujinimas	200	0,1688
Atnaujinimas	400	0,3633
Atnaujinimas	800	0,7266
Atnaujinimas	2000	1,6563
Atnaujinimas	3000	3,3177
Pašalinimas	1	0,0003
Pašalinimas	7	0,0041
Pašalinimas	25	0,0258
Pašalinimas	50	0,0352
Pašalinimas	100	0,0667
Pašalinimas	200	0,1328
Pašalinimas	400	0,2832
Pašalinimas	800	0,5508
Pašalinimas	2000	1,3828
Pašalinimas	3000	2,3125

Išsaugotų procedūrų (angl. *stored procedures*) duomenų modelis	Įrašų kiekis, vnt.	Vid. laikas, sec.
Nuskaitymas	1	0,0006
Nuskaitymas	7	0,0007
Nuskaitymas	25	0,0031
Nuskaitymas	50	0,0055
Nuskaitymas	100	0,0042
Nuskaitymas	200	0,0063
Nuskaitymas	400	0,0117
Nuskaitymas	800	0,0273
Nuskaitymas	2000	0,0469
Nuskaitymas	3000	0,1081
Įterpimas	1	0,0009
Įterpimas	7	0,0088
Įterpimas	25	0,0188
Įterpimas	50	0,0359
Įterpimas	100	0,0729
Įterpimas	200	0,1609
Įterpimas	400	0,3047
Įterpimas	800	0,5938
Įterpimas	2000	1,5039
Įterpimas	3000	2,6979
Atnaujinimas	1	0,0003
Atnaujinimas	7	0,0069
Atnaujinimas	25	0,0234
Atnaujinimas	50	0,0461
Atnaujinimas	100	0,0854
Atnaujinimas	200	0,1656
Atnaujinimas	400	0,3359
Atnaujinimas	800	0,6172
Atnaujinimas	2000	1,668
Atnaujinimas	3000	2,75
Pašalinimas	1	0,0006
Pašalinimas	7	0,0041
Pašalinimas	25	0,0164
Pašalinimas	50	0,0344
Pašalinimas	100	0,0698
Pašalinimas	200	0,1281
Pašalinimas	400	0,2656
Pašalinimas	800	0,5234
Pašalinimas	2000	1,3203
Pašalinimas	3000	2,2396

[11] 2.3GHz taktinio dažnio procesorius ir 1024MB operatyvinės atminties.