KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Artūras Jurevičius

Centralizuoto informacijos sistemų atnaujinimo paslaugų modelis ir prototipas

Magistro darbas

Darbo vadovas
prof. L. Nemuraitė

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Artūras Jurevičius

Centralizuoto informacijos sistemų atnaujinimo paslaugų modelis ir prototipas

Magistro darbas

Recenzentas
doc. dr. V. Pilkauskas
2008-01-

Vadovas
prof. L. Nemuraitė
2008-01-

Atliko
IFM-2/4 gr. stud.
Artūras Jurevičius
2008-01-07

Kaunas, 2008

Service model and prototype of centralised information systems update

SUMMARY

Organisation that provides information system creation services often has to ensure support for installed systems and update them when it‘s necessary. Usually the company uses similar or identical software components to perform similar tasks in different projects. So when one of these components is left faulty, the company needs to make individual corrections in every system using that component and thus interrupt their work.

A centralised architecture is necessary to solve this problem. The idea is to place such problematic components into separate files, called code fragments. When an update occurs, a new file is created and the database link to file changes for the updated fragment. Thus customers‘ Web pages are always being generated using up-to-date software components.

A Web service based model of centralised information systems update is offered in this work. The main service design principles have been analyzed and are described before proposing a full model for prototype realization. Furthermore, a Web service definition (WSDL) model is extended with QoS (Quality of Service) characteristics. This model is useful when choosing one of many identical services, because service quality becomes the main criteria in such case. Experimental research has revealed the values of these characteristic dimensions for the model created in this work.

The centralised update model has been designed to reduce information systems update time. The implemented prototypes are convenient for end-users and do not interrupt their systems operation.

Key words: centralised information systems update, service design principles, quality of service, Web services.

TURINYS

1. ĮVADAS

Užsakovų informacijos sistemas kurianti bei diegianti organizacija dažnai turi užtikrinti ir įdiegtų sistemų palaikymą. Tačiau sistemų atnaujinimas reikalauja didelių laiko sąnaudų. Nemažai uždavinių kartojasi ne viename projekte, todėl paprastai tokia įmonė būna susikūrusi savo programinio kodo bibliotekas, šablonus ar komponentus. Panašiems uždaviniams spręsti skirtingose sistemose naudojamos tos pačios ar panašios priemonės. Taigi jeigu dėl kokių nors priežasčių būtina atlikti pakeitimus vienoje iš įmonės įdiegtų sistemų, didelė tikimybė, kad tokius pat pakeitimus reikės atlikti ir daugelyje kitų tos pačios įmonės realizuotų sistemų. Kai tokių koreguotinų sistemų yra kelios dešimtys, akivaizdu, kad greitai atlikti atnaujinimų visose sistemose nepavyks. Tokiu atveju tiek įmonė patirs nuostolių dėl perteklinio darbo, tiek užsakovams gali būti padaryta žala dėl užsitęsusio netinkamo sistemų veikimo. Natūralu, kad kuriamų sistemų programinio kodo fragmentus (komponentus, funkcijas), kuriems yra tikimybė ateityje keistis, naudinga realizuoti taip, kad atlikus pakeitimus vienoje sistemoje, jie atsispindėtų ir kitose to reikalaujančiose sistemose. Kitaip sakant, kad būtų galima sumažinti sistemų atnaujinimo laiko sąnaudas, būtina centralizuota architektūra.

Informacijos sistemų puslapių formavimui šiame darbe siūloma tokia idėja: kiekvienas probleminis programinio kodo fragmentas, kuriam yra bent menkiausia tikimybė ateityje keistis, realizuojamas atskiru failu, o centralizuoto atnaujinimo metu sukuriamas naujas fragmento failas bei pakeičiama nuoroda į jį duomenų bazėje. Tokiu būdu užtikrinama, kad kiekvieną kartą formuojant informacijos sistemos puslapį bus naudojami atnaujinti programinio kodo failai.

Šiame darbe informacijos sistemų atnaujinimo problema sprendžiama kuriant centralizuoto atnaujinimo modelį tinklo paslaugų pagrindu. Tinklo paslauga (angl. Web service) – tai programinė įranga ar jos dalis, kuri gali būti pasiekiama konkrečiu URI (Uniform Resource Identifier) adresu, jos sąsaja, įvedimo bei išvedimo parametrai yra tinkamai aprašyti naudojant WSDL (Web Services Description Language) standartą. Tinklo paslaugų pirminė paskirtis yra spręsti įmonių taikomųjų uždavinių integravimo uždavinius pasauliniu mastu. Tačiau šiame darbe parodoma, kad atnaujinimo tikslais ir uždarame informacijos sistemų rate galima pritaikyti tinklo paslaugų teikiamas galimybes.

Problemą spręsti tinklo paslaugų pagrindu pasirinkta dėl standartizavimo ir plečiamumo – tai yra pagrindiniai paslaugų pranašumai, lyginant, pavyzdžiui, su duomenų perdavimu paprastomis HTTP užklausomis. Tinklo paslaugomis paremtai sistemai neturi įtakos naudojamos technologijos, sąveikaujančios sistemos gali būti realizuotos skirtingose platformose.

Šio darbo tyrimo sritis – informacijos sistemų kūrimas paslaugų architektūroje (Service Oriented Architecture (SOA)), tinklo paslaugų architektūra. Tyrimo objektas – automatizuoto informacijos sistemų atnaujinimo procesas paslaugų architektūros pagrindu.

Tyrimo tikslas – sukurti centralizuoto informacijos sistemų atnaujinimo paslaugų modelį ir programinį prototipą, kuris užtikrintų mažesnes atnaujinimo laiko sąnaudas, būtų patogus galutiniam vartotojui ir turėtų patikimas kokybines charakteristikas. Tyrimo uždaviniai:

- atlikti egzistuojančių atnaujinimo sistemų, tinklo paslaugų standartų ir architektūros analizę bei nustatyti, kaip būtų galima efektyviai atlikti užsakovų informacijos sistemų atnaujinimą;
- išnagrinėti paslaugų projektavimo principus bei siekti, kad projektavimo bei realizavimo metu jų būtų laikomasi;
- išplėsti tinklo paslaugų aprašymo modelį kokybinėmis charakteristikomis, kad jį būtų galima naudoti ne tik funkciniais tikslais, bet ir įvertinant paslaugų kokybę;
- nustatyti atnaujinimo sistemos reikalavimus ir pagal juos sukurti atnaujinimo sistemos modelį;
- realizuoti sistemos prototipą, kad būtų galima atlikti eksperimentą su keletu centrinės bei klientinės sistemų pavyzdžių;
- atlikti eksperimentą, skirtą nustatyti paslaugų kokybines charakteristikas bei patikrinti, ar ženkliai sukurtas modelis padidina klientinių sistemų puslapių formavimo trukmę.

Pagrindinė tyrimo metodologija yra konstruktyvusis tyrimas. Be to, taikoma lyginamoji analizė ir eksperimentinis tyrimas. Prototipui projektuoti naudojamas unifikuotas procesas (angl. Rational Unified Process, RUP) [14] ir unifikuota modeliavimo kalba (angl. Unified Modelling Language, UML).

Analizuojant paslaugų architektūrą remtasi [1], [9], [10], [12] šaltiniais. Išskirtinai tinklo paslaugoms nagrinėti naudotas [16] šaltinis. SOAP bei WSDL standartams analizuoti naudoti atitinkamai [2] ir [3], [4] šaltiniai. Paslaugų projektavimo principams nagrinėti remtasi [7], o paslaugų aprašymo išplėtimui kokybinėmis charakteristikomis – [5] ir [13] šaltiniais. Konkrečiai tinklo paslaugų realizacijai PHP kalba remtasi [6] bei [11] šaltiniais. Tarp sistemų perduodamų duomenų saugumui užtikrinti remtasi IB Pay elektroninės bankininkystės paslaugų technine specifikacija [8] bei OpenSSL dokumentacija [15].

Šiame darbe išanalizuota standartinė paslaugų architektūra bei nuspręsta, kad informacijos sistemų atnaujinimo problemai spręsti reikalinga centralizuota architektūra. Nustatyti funkciniai bei nefunkciniai reikalavimai centrinei bei klientinei sistemoms. Išnagrinėti 8 pagrindiniai paslaugų projektavimo principai. Remiantis dauguma šių principų bei laikantis nustatytų reikalavimų suprojektuotas išsamus sąveikaujančių sistemų modelis, įskaitant ir tinklo paslaugų modelį, kuris išplėstas kokybinėmis tinklo paslaugų charakteristikomis. Toks išplėstas modelis ne tik leidžia aprašyti paslaugų sąsają, bet ir suteikia galimybę parodyti teikiamų paslaugų kokybę. Pagal

centralizuotos sistemos modelį realizuotas programinis prototipas, testuojant jį visos realizacijos metu bei po jos. Pagal jį sukurta keletas centrinių bei klientinių sistemų pavyzdžių, besiskiriančių naudojamų fragmentų kiekiu bei dydžiu. Atliktas eksperimentinis realizuotų pavyzdžių tyrimas, kurio metu išmatuoti kokybinių charakteristikų įverčiai. Taip pat eksperimento metu nustatyta, kiek padidėja puslapių formavimo trukmė informacijos sistemose, priklausomai nuo naudojamų probleminių kodo fragmentų kiekio bei dydžio.

Darbo rezultatas – remiantis paslaugų projektavimo principais bei išplečiant tinklo paslaugų aprašą kokybinėmis charakteristikomis sukurtas centralizuoto užsakovų informacijos sistemų atnaujinimo paslaugų modelis bei programinis prototipas, kuris yra patogus galutiniam vartotojui, leidžia sumažinti atnaujinimo laiko sąnaudas bei nepertraukia klientinių sistemų darbo.

Darbo struktūra:

- Skyriuje *Informacijos sistemų atnaujinimo galimybių analizė* nagrinėjama informacijos sistemų atnaujinimo problema, pateikiamas tokias sistemas kuriančios bei diegiančios organizacijos veiklos modelis. Taip pat pateikiama glausta egzistuojančių atnaujinimo sistemų analizė. Šiame skyriuje taip pat analizuojama standartinė paslaugų architektūra bei nusprendžiama, jog reikalinga centralizuota atnaujinimo paslaugų architektūra.
- Skyriuje *Centralizuoto atnaujinimo sistemos reikalavimai* pateikiami funkciniai centrinės bei klientinės sistemų reikalavimai panaudojimo atvejų pavidalu. Taip pat išvardijami nefunkciniai reikalavimai sistemoms.
- Skyriuje *Tinklo paslaugų modelis ir kūrimo principai* analizuojami 8 pagrindiniai paslaugų projektavimo principai: *standartizuotas paslaugos kontraktas*, *laisvas paslaugos susiejimas*, *paslaugos abstrakcija*, *pakartotinis paslaugos panaudojamumas*, *paslaugos autonomija*, *paslaugos būsenų neturėjimas*, *paslaugos atrandamumas* ir *paslaugos komponuojamumas*. Taip pat sudaromas darbe modeliuojamų tinklo paslaugų WSDL aprašo modelis bei jo išplėtimas kokybinėmis charakteristikomis.
- Skyriuje *Centralizuoto atnaujinimo sistemos modelis* parodoma loginė centrinės bei klientinės sistemų architektūra, reikalavimų analizės modeliai, centrinės bei klientinės dalies klasių modeliai. Taip pat pateikiami duomenų, vartotojo sąsajos bei sistemos elgsenos modeliai. Galiausiai aprašoma idėja, atspindinti darbo naujumą.
- Skyriuje *Sistemos prototipo realizacija* pirmiausiai pateikiami centrinės bei klientinės sistemų realizacijos modeliai. Toliau aprašomas sistemų autentifikavimas bei perduodamų duomenų šifravimas, taip pat transakcijų valdymas. Pateikiamas sistemų testavimo planas bei rezultatai. Galiausiai aprašomos sukurtos atnaujinimo sistemos realizacijos priemonės bei diegimo nurodymai.

- Skyriuje *Eksperimentinis paslaugų sistemos tyrimas* aprašomas realizuotą prototipą atitinkančiuose centrinės bei klientinės sistemų pavyzdžiuose atliktas eksperimentas. Pirmiausiai aprašomas atnaujinimo pranešimų siuntimo iš centrinės į klientines sistemas trukmės tyrimas, kurio metu nustatyta siuntimo trukmės priklausomybė nuo pranešimų kiekio. Toliau aprašomas atnaujinimų vykdymo klientinėse sistemose trukmės tyrimas, kurio metu nustatoma atnaujinimų vykdymo trukmės priklausomybė nuo atnaujinamų kodo fragmentų kiekio bei dydžio. Po to aprašomas informacijos sistemų puslapių formavimo trukmės pokyčių tyrimas, kurio metu nustatoma, kokiems kodo fragmentų dydžiams bei kiekiams esant puslapių formavimo trukmė padidėja ne daugiau kaip 5 %. Galiausiai nustatytos charakteristikos įtaukiamos į išplėstą tinklo paslaugų aprašo modelį.
- Skyriuje *Darbo apibendrinimas* pateikiami darbo rezultatai, sukurtos sistemos atitikimo paslaugų projektavimo principams įvertinimas bei darbo išvados.

Darbo tematika parengtas straipsnis (pateikiamas *prieduose*) bei pristatytas konferencijai „Informacinės Technologijos 2008“.

2. INFORMACIJOS SISTEMŲ ATNAUJINIMO GALIMYBIŲ ANALIZĖ

Analizės metu siekiama atskleisti užsakovų informacijos sistemų atnaujinimo problemą, lemiančią dideles įdiegtų interneto informacijos sistemų palaikymo darbo sąnaudas. Taip pat siekiama išanalizuoti, kokiais būdais sprendžiamos panašios problemos, pasirinkti vieną jų, sukurti naują ar modifikuoti esamą sprendimo metodą. Galutinis analizės tikslas – suformuluoti užduotį tyrimui, kuris padės sumažinti įdiegtų informacijos sistemų palaikymo laiko sąnaudas ir bus patogus galutiniam vartotojui.

2.1. Informacijos sistemų atnaujinimo problema

Šio tyrimo sritis – informacijos sistemų kūrimas paslaugų architektūroje (Service Oriented Architecture (SOA)), tinklo paslaugų architektūra. Tyrimo objektas – automatizuoto informacijos sistemų atnaujinimo procesas paslaugų architektūros pagrindu.

Organizacija, kuriai skirta modeliuojama sistema, yra interneto informacijos sistemų kūrimo bei diegimo paslaugas teikianti įmonė. Daugumai šios įmonės sukurtų bei įdiegtų sistemų turi būti užtikrintas palaikymas. Dažnai būna kritinių atvejų, kai jau įdiegtoje sistemoje reikia atlikti esminių pakeitimų. Tokių pakeitimų poreikis gali atsirasti dėl toliau išvardintų priežasčių:

- po sistemos įdiegimo pastebima, kad joje yra likusių esminių klaidų, dėl kurių sukurta sistema netinkamai atlieka numatytas funkcijas;
- atsiranda sistemose naudojamų standartų pokyčių, kurie iššaukia tam tikrus pokyčius pačiose sistemose;
- išleidžiamos naujos programinės įrangos, glaudžiai susijusios su įdiegtomis sistemomis, versijos (pavyzdžiui, interneto naršyklės), dėl kurių būtina atlikti tam tikrus pakeitimus tose sistemose.

Paprastai informacijos sistemas diegianti įmonė būna susikūrusi savo programinio kodo bibliotekas, šablonus ar komponentus. Nemažai uždavinių kartojasi ne viename projekte. Panašiems uždaviniams spręsti skirtinguose projektuose naudojamos tos pačios ar panašios priemonės. Taigi jeigu dėl kurios nors iš aukščiau išvardintų priežasčių būtina atlikti pakeitimus vienoje iš įmonės įdiegtų sistemų, didelė tikimybė, kad tokius pat pakeitimus reikės atlikti ir daugelyje kitų tos pačios įmonės realizuotų sistemų. Jeigu tokių sistemų būtų tik keletas, tuomet tokių pakeitimų darbo sąnaudos įmonei didelės žalos neatneštų. Tačiau kai tokių koreguotinų sistemų yra kelios dešimtys, akivaizdu, kad greitai atlikti pakeitimų visose sistemose nepavyks. Tokiu atveju tiek įmonė patirs nuostolių dėl perteklinio darbo, tiek užsakovams gali būti padaryta žala dėl užsitęsusio netinkamo sistemų veikimo.

Realus pavyzdys – Internet Explorer 7-osios versijos pasirodymo iššaukti pakeitimai įmonės sukurtose interneto svetainėse. Probleminis objektas buvo iššokantys langai, kurie su senesnėmis naršyklių versijomis veikė taip, kaip numatyta. 7-ojoje versijoje iššokantys langai tinkamai nebefunkcionavo, todėl teko koreguoti keliolikos sistemų programinį kodą. Tai realus pavyzdys, rodantis, kad problema yra ne tik teorinė, bet egzistuoja ir praktikoje.

Galima daryti išvadą, kad kuriamų sistemų dalis (komponentus, funkcijas), kurioms yra tikimybė ateityje keistis, būtina realizuoti taip, kad atlikus pakeitimus vienoje sistemoje, jie atsispindėtų ir kitose to reikalaujančiose sistemose. Kitaip sakant, būtina centralizuota architektūra. 1 lentelėje pateikti centralizuotos architektūros privalumai. Tinklo paslaugų galimybės leidžia suprojektuoti tokią architektūrą, kuri tinkamai išspręstų šią problemą.

1 lentelė. Siekiamos sistemos privalumai, lyginant su senais informacijos sistemų palaikymo metodais

Siekiamos sistemos privalumai	**Palyginimas su ankstesniais metodais**
Sistemos palaikymas atliekamas centralizuotai. Atlikti pakeitimai atsispindi visose to reikalaujančiose sistemose.	Anksčiau kiekvienoje sistemoje pakeitimus reikėjo atlikti atskirai.
Pakeitimai gali būti atlikti net nematant konkrečių sistemų ir neprisijungus prie jų.	Anksčiau norint atlikti pakeitimus reikėjo tiesiogiai dirbti prie konkrečių sistemų.

Darant prielaidą, jog informacijos sistemos puslapių formavimas susideda iš daugelio programinio kodo fragmentų, tokius fragmentus galima atskirti. Kiekvieną probleminį kodo fragmentą, kuriam yra bent menkiausia tikimybė ateityje keistis, galima realizuoti atskirame faile. Atskirtas kodo fragmentas turi būti lengvai panaudojamas daugelyje sistemų (pavyzdžiui, atskirta klasė ar klasių biblioteka), kad būtų prasminga jį atnaujinti centralizuotai. Įprastai formuojant puslapį tokie fragmentai įtraukiami (angl. include) statiškai, t. y. tą patį fragmentą kiekvieną kartą realizuoja tas pats failas. Šio darbo idėja yra įtraukti kodo fragmentus dinamiškai, t. y. kiekvieną kartą prieš juos įtraukiant kreiptis į duomenų bazę bei iš jos išgauti naujausių fragmentų versijų failų vardus. Tokiu būdu užtikrinama, kad kiekvieną kartą formuojant informacijos sistemos puslapį bus naudojami atnaujinti programinio kodo failai.

Iš viso to išplaukia tokie tyrimo uždaviniai:

- atlikti egzistuojančių atnaujinimo sistemų, tinklo paslaugų standartų ir architektūros analizę bei nustatyti, kaip būtų galima efektyviai atlikti užsakovų informacijos sistemų atnaujinimą;
- išnagrinėti paslaugų projektavimo principus bei siekti, kad projektavimo bei realizavimo metu jų būtų laikomasi;
- išplėsti tinklo paslaugų aprašymo modelį kokybinėmis charakteristikomis, kad jį būtų galima naudoti ne tik funkciniais tikslais, bet ir įvertinant paslaugų kokybę;

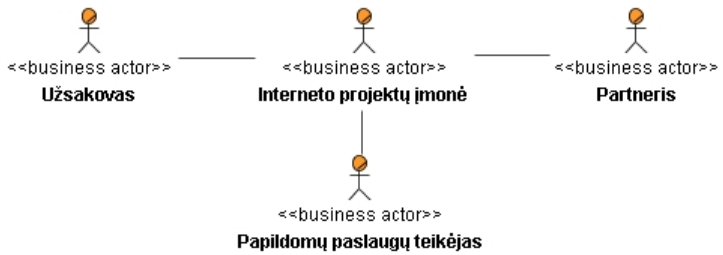
- nustatyti atnaujinimo sistemos reikalavimus ir pagal juos sukurti atnaujinimo sistemos modelį;
- realizuoti sistemos prototipą, kad būtų galima atlikti eksperimentą su keletu centrinės bei klientinės sistemų pavyzdžių;
- atlikti eksperimentą, skirtą nustatyti paslaugų kokybines charakteristikas bei patikrinti, ar ženkliai sukurtas modelis padidina klientinių sistemų puslapių formavimo trukmę.

Šiuos uždavinius galima apibendrinti vienu tyrimo tikslu: sukurti centralizuoto informacijos sistemų atnaujinimo paslaugų modelį ir programinį prototipą, kuris užtikrintų mažesnes atnaujinimo laiko sąnaudas, būtų patogus galutiniam vartotojui ir turėtų patikimas kokybines charakteristikas.

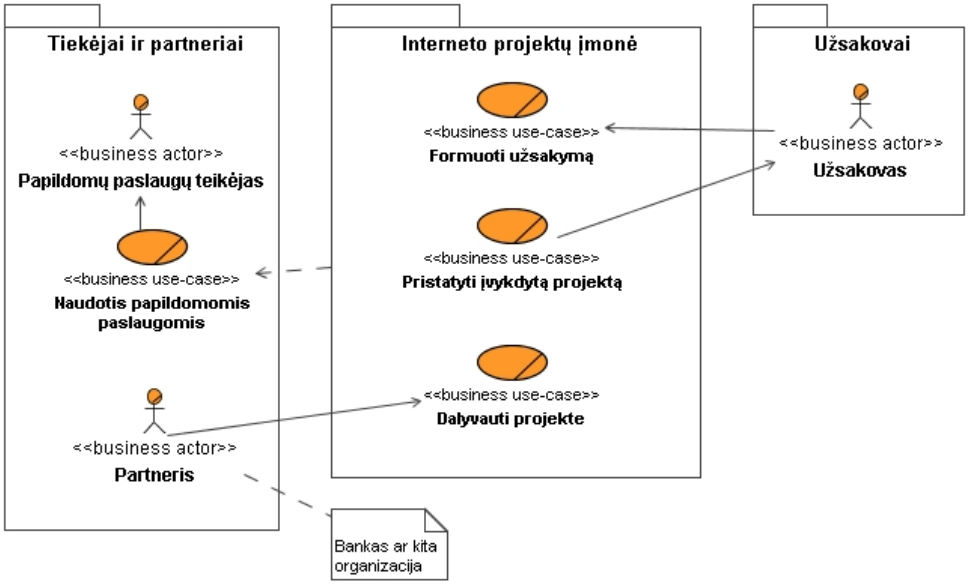
2.2. Informacijos sistemų kūrimo paslaugas teikiančios organizacijos veiklos modelis

Kaip minėta 2.1 poskyryje, įmonė, kuriai skirta šiame darbe projektuojama sistema, yra informacijos sistemas kurianti ir diegianti įmonė. Šiame poskyryje analizuojama organizacijos veikla. Organizacijos veiklos modelyje siekiama parodyti, kuriai įmonės veiklos daliai kuriamas modelis turi daryti įtaką.

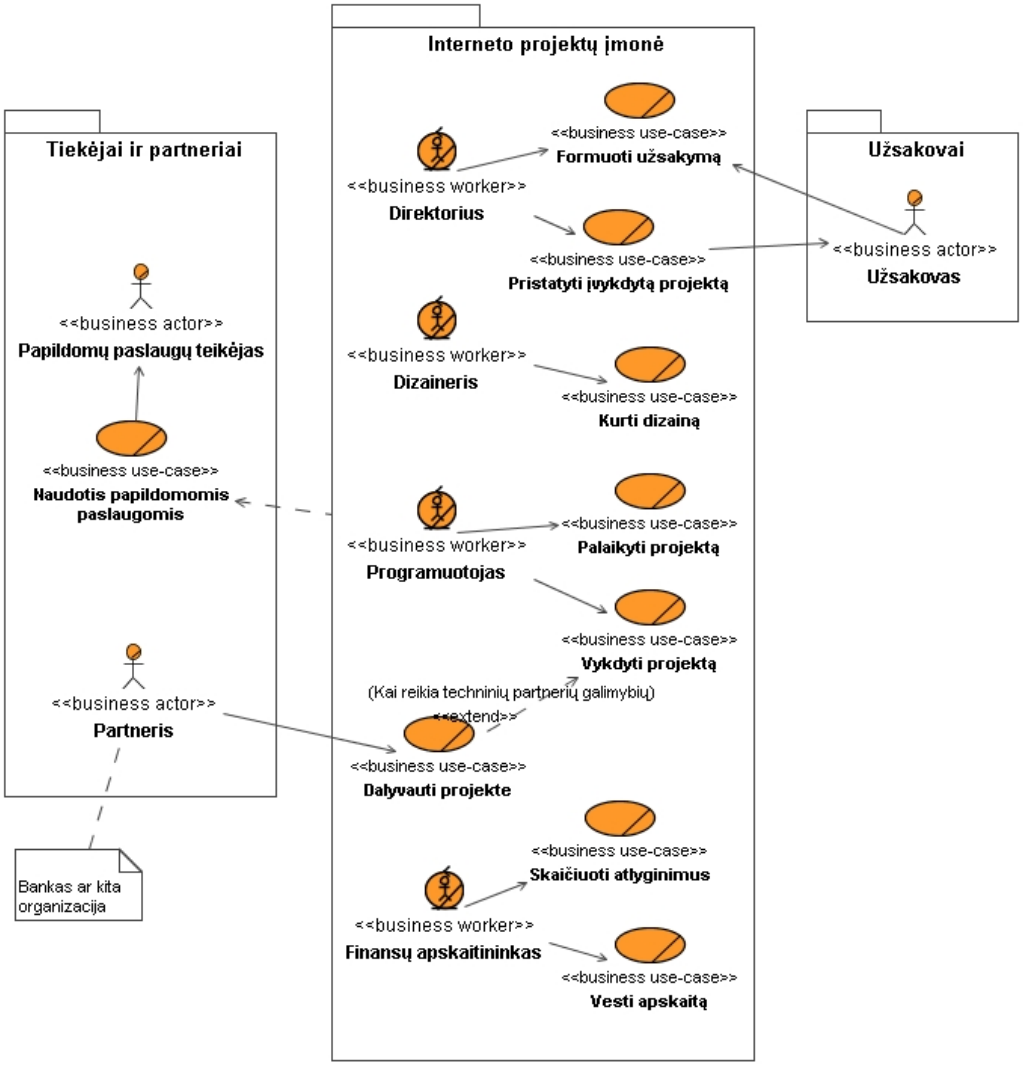
1 pav. pateiktoje kontekstinėje diagramoje parodoma, su kokiais pagrindiniais išoriniais veiklos aktoriais įmonė sąveikauja. Užsakovas – pagrindinis išorinis veiklos aktorius, nes įmonė kiekvieną sistemą kuria užsakovui pagal jo reikalavimus. Partneris – tai kita organizacija, kurios veikla įtraukiama kuriant informacijos sistemą užsakovui (pavyzdžiui, kuriant elektroninio verslo sistemą tai gali būti bankas). Papildomų paslaugų teikėjas – fizinis ar juridinis asmuo, teikiantis vienkartines arba ilgalaikes, tačiau nesusijusias su konkrečiu projektu, paslaugas įmonei (pavyzdžiui, fotografas, reklamos agentas, vertėjas).



1 pav. Organizacijos kontekstinė diagrama



2 pav. Organizacijos veiklos sąveikų modelis



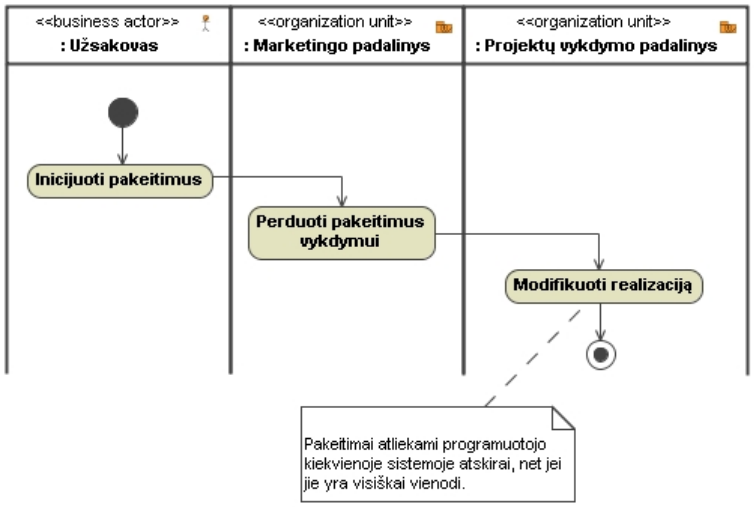
3 pav. Organizacijos veiklos sąveikų modelis (detalus)

2 pav. pateiktame organizacijos veiklos sąveikų modelyje parodoma ne tik su kuo įmonė sąveikauja, bet ir kokie veiksmai ją sieja su išoriniais aktoriais. 3 pav. pateiktas detalizuotas modelis, parodant ir įmonės vidinius darbuotojus, dalyvaujančius sąveikose.

4 pav. parodytas pagrindinio įmonės veiklos proceso modelis. Jame matoma, iš kokių sudėtinių procesų susideda kiekvienos informacijos sistemos kūrimas ir diegimas.

4 pav. Pagrindinio organizacijos veiklos proceso modelis

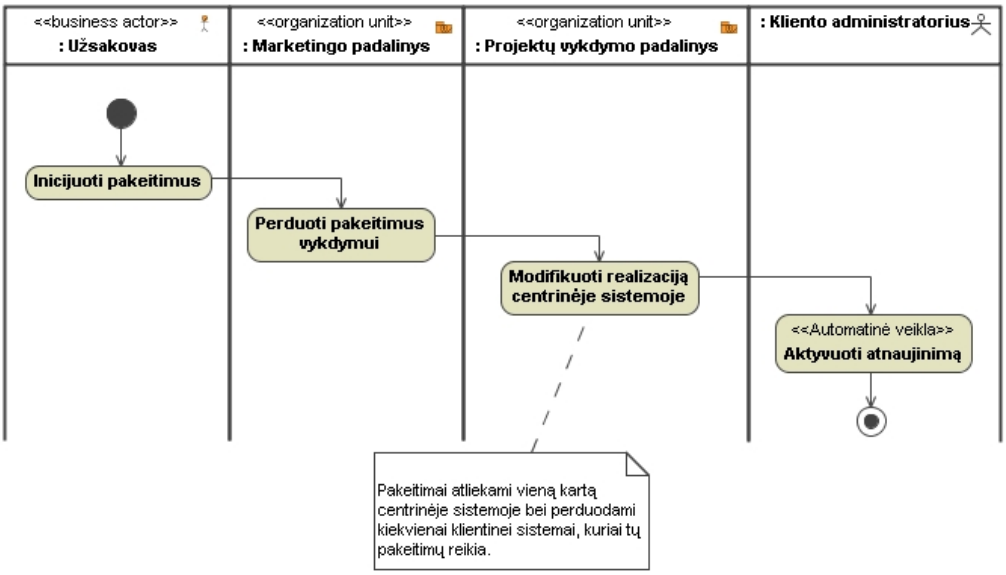
5 pav. pateiktas programinio kodo atnaujinimo veiklos proceso modelis. Programinio kodo atnaujinimas gali būti inicijuotas užsakovo arba pačios įmonės, jei, pavyzdžiui, sename kode randama klaidų. Tokiu atveju įmonė turi vykdyti tokį procesą kiekvienoje įdiegtoje sistemoje, kurioje naudojamas tas atnaujinamas kodas. Tam įmonė turi skirti daug laiko, tuo pačiu ir papildomų lėšų. Būtent šių sąnaudų sumažinimui yra skirtas šiame darbe kuriamas modelis.



5 pav. Programinio kodo atnaujinimo (pakeitimų) veiklos proceso modelis

6 pav. pateiktas patobulintas programinio kodo atnaujinimo veiklos proceso modelis. Jame parodoma, kokią įtaką kuriamas modelis daro organizacijos veiklos procesui. Atnaujinimas vykdomas centrinėje sistemoje, o klientinėms sistemoms, naudojančioms atnaujinamą programinį

kodą, išsiunčiami pranešimai apie atnaujinimą. Tuomet kiekvienos klientinės sistemos administratoriui tereikia aktyvuoti atnaujinimą. Tai galima realizuoti ir automatiškai, t. y. neįtraukiant klientinių sistemų administratorių, tačiau toks sprendimas pasirinktas tam, kad klientinių sistemų administratoriai galėtų patys nuspręsti, kada atlikti pakeitimus.



6 pav. Patobulintas programinio kodo atnaujinimo (pakeitimų) proceso modelis

2.3. Egzistuojančių atnaujinimo sistemų analizė

Prieš projektuojant naują sistemą verta išanalizuoti panašias problemas sprendžiančių kitų sistemų veikimą, kad būtų galima panaudoti atskiras jų idėjas arba atmesti jose naudojamus sprendimus kaip netinkamus. Analogiškų paslaugų, kurios būtų skirtos informacijos sistemų atnaujinimui, nėra. Šiame poskyryje nagrinėjama, kaip atnaujinimai atliekami taikomosiose programose bei lietuviškame produkte E-pasas.

Atnaujinimo problema, kurią spręs šiame darbe kuriama sistema, taikomosiose programose sprendžiama specialių programinių agentų ar posistemių pagalba. Centriniame serveryje reguliariai būna atliekami programinės įrangos pakeitimai. Startuojant taikomajai programai ar net jos naudojimo metu tikrinama, ar nėra atnaujinimų centriniame serveryje. Jei jų yra, tolesnis scenarijus gali būti įvairus. Toliau aprašomi du pavyzdiniai scenarijai.

Pats paprasčiausias scenarijus – naudojant programinę įrangą ekrane pateikiamas atnaujinimo pranešimas bei nuoroda į interneto svetainę, kur patalpinta nauja programos versija. Vartotojui reikia ją atsisiųsti, o atsisiuntus ji įdiegiama taip pat, kaip ir įdiegiant pirmą kartą. Tokiu būdu atnaujinama, pavyzdžiui, Adobe Reader (bent jau iki 8 versijos). Programavimo prasme toks atnaujinimas yra paprasčiausias. Jo patikimumas yra didelis, kadangi įdiegiama visa stabili programa iš karto. Be to, neiškyla saugumo problemų, nebent pats vartotojas neatkreipia dėmesio,

kad naująją versiją siunčiasi iš nepatikimo šaltinio. Tačiau vartotojui toks atnaujinimo scenarijus nėra patogus.

Vartotojui labiau priimtinas toks atnaujinimo būdas, kai atnaujinimai įdiegiami automatiškai arba gavus jo sutikimą. Šis būdas paplitęs daugelyje internetu atnaujinamų programinių produktų (pvz., daugelis Microsoft paketų, Mozilla Firefox naršyklė ir kt.). To paties aukščiau minėto Adobe Reader tarpiniai atnaujinimai tarp naujų versijų vykdomi šiuo būdu. Tačiau tokio atnaujinimo metu gali kilti saugumo problemų, jeigu atnaujinimų serveris nėra autentifikuojamas. Patikimumas taip pat gali būti mažesnis nei pirmuoju atveju, kadangi gali būti atvejų, kai atnaujinimas nebus sėkmingai įvykdytas (dėl nesuteiktų teisių ar pan.). Toliau kaip pavyzdys trumpai aprašomas Mozilla Firefox naršyklės atnaujinimas.

Mozilla Firefox paleidimo bei naudojimo metu yra periodiškai tikrinama, ar nėra pačios naršyklės arba jos priedų (Add-ons) atnaujinimų. Programos konfigūracijoje leidžiama nustatyti, ar atnaujinimus vykdyti automatiškai, ar būtina atsiklausti vartotojo, kuris tokiu atveju gali atidėti atnaujinimo vykdymą. Vykdant atnaujinimą naršyklė turi startuoti iš naujo, tačiau prieš tai yra išsaugoma jos būsena, t. y. visų atidarytų tinklalapių adresai. Įdiegus atnaujinimą (-us), naršyklė grįžta į išsaugotą būseną, t. y. atidaro visus prieš atnaujinimą atidarytus tinklalapius. Toks atnaujinimas yra patogus vartotojui, kadangi tik minimaliai pertraukia jo darbą.

Aukščiau aprašyti du scenarijai yra skirti taikomosioms programoms atnaujinti, tačiau informacijos sistemų atnaujinimui jie gali būti tik kaip pavyzdžiai (kuriamai sistemai galima pritaikyti tai, kad vartotojui leidžiama pasirinkti, ar vykdyti atnaujinimą, ar jį kuriam laikui atidėti). Šiek tiek panašumų į šiame darbe kuriamą atnaujinimo sistemą informacijos sistemų terpėje turi lietuviškos E-paso prisijungimo sistemos (www.e-pasas.lt) atnaujinimas. E-pasas yra integruojamas produktas, skirtas prisijungimui su tais pačiais duomenimis keliuose tinklalapiuose. Integravus jį tinklalapyje, laikas nuo laiko gali būti vykdomas pagrindinių E-paso failų atnaujinimas. Centriniam E-paso serveriui išsiuntus atnaujinimo komandą į tinklalapį, šiame paleidžiamas atnaujinimo mechanizmas, susidedantis iš tokių žingsnių:

- patikrinama, ar failai tikrai turi būti atnaujinami (jei ne, atnaujinimas nėra tęsiamas);
- atsiunčiamos naujos atnaujinamų failų versijos;
- padaromos senų failų atsarginės kopijos;
- E-pasas atjungiamas nuo tinklalapio;
- atnaujinami reikiami failai;
- patikrinama, ar atnaujinimas įvykdytas teisingai (jei ne, atstatomi senieji failai);
- E-pasas vėl prijungiamas prie tinklalapio;
- išsiunčiama ataskaita apie atnaujinimą į centrinį E-paso serverį.

E-paso atnaujinimo patikimumas gana didelis, nes nesėkmės atveju yra atstatomi seni failai. Saugumas taip pat pakankamas, kadangi siunčiami duomenys yra šifruojami slaptu raktu. Tačiau dar saugiau būtų šifravimui naudoti privataus ir viešo rakto derinį.

Toks atnaujinimo scenarijus šiame darbe kuriamai sistemai netinka, nes jis pritaikytas būtent tik E-pasui. Kad ir koks tinklalapių kiekis bebūtų, visuose juose E-paso failai bus vienodi. Šiame darbe priešingai – kuriama sistema skirta skirtingoms informacijos sistemoms, kuriose dalis atnaujinamų programinio kodo fragmentų sutampa, atnaujinti.

Didelis tokio scenarijaus trūkumas yra tai, kad atnaujinimo metu E-pasas yra atjungiamas. Tai reiškia, kad tuo metu tinklalapyje naršantys vartotojai neturi galimybės prisijungti, o prisijungusieji yra automatiškai atjungiami. Žinoma, tas atjungimo laiko tarpas yra mažas, tačiau gali užtekti ir vieno ar dviejų tokių nesėkmingo prisijungimo atvejų, kad vartotojas suabejotų tinklalapio patikimumu. Todėl kuriamos sistemos schema turėtų būti tokia, kad atnaujinimas vyktų nepastebimai vartotojui (bent jau iš blogosios pusės). Taigi nors E-paso atnaujinimas yra artimas kuriamai sistemai pagal taikymo sritį (informacijos sistemos), jo scenarijus nėra tinkamas.

2 lentelėje pateikiamas analizuotų scenarijų apibendrinimas.

2 lentelė. Analizuotų egzistuojančių atnaujinimo scenarijų apibendrinimas

	Naujos versijos atsisiuntimas ir įdiegimas iš naujo	**Atnaujinimų įdiegimas minimaliai trikdant vartotojo darbą**	**E-paso failų atnaujinimo tinklalapiuose scenarijus**
Paskirtis	Taikomoji programinė įranga.	Taikomoji programinė įranga.	Interneto informacijos sistemos.
Patikimumas	Patikimiausias.	Gali būti nepatikimas.	Pakankamai patikimas.
Saugumas	Saugus, nebent vartotojas naują versiją atsisiunčia iš nepatikimo šaltinio.	Gali būti labai nesaugus, jeigu atnaujinimų serveris nėra autentifikuojamas.	Pakankamai saugus (šifravimas slaptuoju raktu).
Privalumai	Kūrėjams nereikia realizuoti atnaujinimų apdorojimo.	Vartotojo darbas pertraukiamas minimaliai. Vartotojui suteikiama galimybė atidėti atnaujinimą.	Nesėkmės atveju atstatomi seni failai.
Trūkumai	Nepatogu vartotojui.	Kūrėjams būtina realizuoti atnaujinimų apdorojimą.	Atnaujinimo metu E-pasas atjungiamas nuo tinklalapio.

2.4. Tinklo paslaugų architektūros analizė ir įgyvendinimo varianto pagrindimas

2.1 poskyryje aprašytos problemos sprendimui kuriamoje sistemoje bus naudojamos tinklo paslaugos. Tinklo paslauga (Web servisas) yra programinė įranga ar jos dalis, kuri gali būti pasiekiama konkrečiu URI (Uniform Resource Identifier) adresu, jos sąsaja, įvedimo bei išvedimo parametrai yra tinkamai aprašyti naudojant XML (eXtensible Markup Language) kalba paremtus

standartus [1]. Toliau glaustai aprašomas kiekvienas iš jų, o po to analizuojama pati tinklo paslaugų architektūra.

SOAP (Simple Object Access Protocol) yra protokolas, skirtas tinklo paslaugų užklausų perdavimui [1, 2]. Tai nėra transporto lygmens protokolas. SOAP užklausa yra įterpiama į HTTP arba SMTP protokolų užklausą, o pats SOAP pranešimas yra XML dokumentas. Jį sudaro antraštinė bei turinio dalys. Antraštėse aprašoma, kuriuo protokolu pranešimas turi būti perduodamas, nurodomos taisyklės, kaip elgtis su pranešimu. Turinyje perduodamas pranešimas XML formatu.

WSDL (Web Services Description Language) – tai tinklo paslaugų aprašymui skirta kalba [1, 3, 4]. WSDL dokumentas yra XML dokumentas, sudarytas pagal specialią schemą. Jame aprašomos paslaugų sąsajos bei gaunamų užklausų (pranešimų) tipai, pateikiami adresai, kuriais konkrečias tinklo paslaugas galima pasiekti. WSDL dokumentai paprastai naudojami ieškant tam tikrus įvedimo bei išvedimo parametrus atitinkančios paslaugos tinklo paslaugų registre.

UDDI (Universal Description Discovery and Integration) registras yra skirtas globalių tinklo paslaugų aprašymų paskelbimui bei paieškai [1]. Registre saugoma informacija apie tinklo paslaugas teikiančias įmones bei apie kiekvieną paslaugą. Taip pat saugomi WSDL dokumentai arba nuorodos į juos.

Problemą spręsti tinklo paslaugų pagrindu pasirinkta dėl standartizavimo ir plečiamumo – tai yra pagrindiniai paslaugų pranašumai, lyginant, pavyzdžiui, su duomenų perdavimu paprastomis HTTP užklausomis. Tinklo paslaugomis paremtai sistemai neturi įtakos naudojamos technologijos, t. y. sąveikaujančios sistemos gali būti realizuotos naudojant skirtingas technologijas. Nauda, gaunama remiantis tinklo paslaugomis šiame darbe, yra tokia [7]:

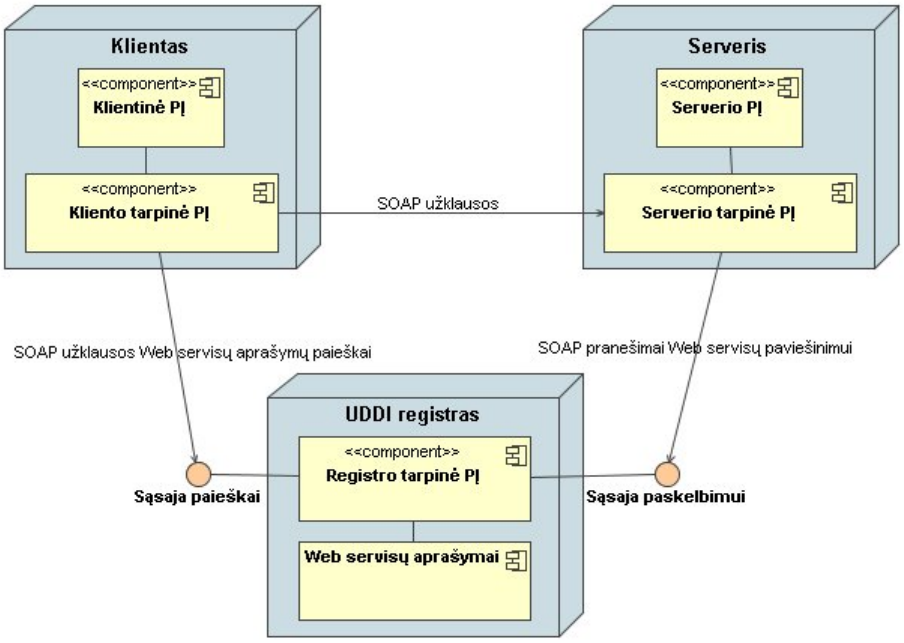
- didesnis operacinis suderinamumas (Increased Intrinsic Interoperability) – paslaugos yra savaime suderinamos, nėra papildomo integravimo poreikio;
- didesnė investicijų grąža (Increased Return of Investment) – sistema gali plėstis, tačiau paslaugų sąsajos bus kuriamos pagal vienodus standartus, todėl kiekvienas tolesnis išplėtimas kainuos vis mažiau;
- greitesnė organizacijos reakcija (Increased Organizational Agility) į aplinkos pokyčius, kadangi naudojami vienodi standartai.

Kadangi pasirinkta realizuoti sistemą tinklo paslaugų pagrindu, reikia pasirinkti arba standartinę tinklo paslaugų architektūrą, arba ją modifikuoti, kad būtų tenkinami sistemai keliami reikalavimai. Standartinės tinklo paslaugų architektūros [1] esmė yra tokia. Tinklo paslaugų aprašymai WSDL kalba yra skelbiami UDDI registruose. Tam registrai turi išorinę sąsają, per kurią iš serverio gali būti prisijungta bei patalpintas ar atnaujintas norimos tinklo paslaugos aprašymas.

Kita UDDI registro sąsaja skirta klientinėms sistemoms tam, kad šios galėtų vykdyti tam tikrus parametrus atitinkančių registre skelbiamų tinklo paslaugų paiešką. Radus tinkamas paslaugas, kliento sistema išsirenka vieną jų bei kreipiasi registre skelbiamu adresu su tinklo paslaugos užklausa. Standartinė schema pateikta 7 pav.

UDDI registras schemoje reikalingas tam, kad klientas visada į tinklo paslaugą kreiptųsi egzistuojančiu adresu. Pavyzdžiui, gali būti, kad tinklo paslaugos adresas pasikeitė, todėl klientas, kreipdamasis senu adresu, paslaugos nepasieks. Tačiau pakeitimus paslaugos teikėjai paprastai nedelsiant paskelbia registre. Dėl to klientai, pirma kreipdamiesi į registrą, gali būti užtikrinti, kad gautu adresu paslauga bus pasiekiama.

UDDI registro sąsajos realizuojamos kaip tinklo paslaugų sąsajos. Taigi tiek serverio, tiek kliento užklausos, nukreiptos į UDDI registrą, yra ne kas kita, o tinklo paslaugų užklausos. Todėl klientas, norėdamas kreiptis į tam tikrą paslaugą, turi atlikti dvi tinklo paslaugų užklausas: į UDDI registrą, o po to ir į serverį.



7 pav. Standartinės tinklo paslaugų architektūros schema

Jei šiame darbe kuriama sistema būtų realizuojama pagal standartinę schemą, tuomet tiek centrinės, tiek klientinės sistemos kiekvieną kartą pačios UDDI registre turėtų susirasti tinklo paslaugą, o tada jau kreiptųsi į ją. Tektų kiekvieną kartą be užklausos į serverį siųsti papildomą užklausą į registrą. Kadangi tiek viena, tiek kita užklausa yra SOAP pranešimas, galima daryti prielaidą, jog duomenų transformavimo bei užklausų apdorojimo trukmė apytiksliai bus dvigubai didesnė nei vienos užklausos atveju. Kuriamos sistemos atveju didelis vėlinimas nebūtų labai didelė problema, tačiau projektuoti atskirą registrą būtų neracionalu, kadangi centrinių bei klientinių

sistemų ratas yra uždaras. Taigi standartinė schema kuriamai sistemai netinka, būtina ją modifikuoti.

Paprasčiausias modifikavimas – UDDI registro atsisakymas. Vietoje jo kiekvienoje klientinėje sistemoje gali būti saugomas centrinių, o centrinėje – klientinių sistemų sąrašas. Taip išvengiama tarpinio mazgo bei papildomų užklausų.

2.5. Analizės išvados

Informacijos sistemas kuriančios organizacijos veiklos analizė parodė, kad užsakovų informacijos sistemų atnaujinimas reikalauja didelių laiko sąnaudų ir yra nepatogus galutiniams vartotojams, nes atnaujinimas sutrikdo jų darbą.

Informacijos sistemas kuriancios organizacijos veiklos analizė taip pat atskleidė, kad geriausiai tokius atnaujinimus būtų vykdyti centralizuotai.

Egzistuojančių atnaujinimo būdų bei tinklo paslaugų architektūros analizė parodė, kad geriausia atnaujinimą organizuoti tinklo paslaugų architektūros pagrindu. Tačiau sprendžiamam uždaviniui standartinė architektūra nėra racionali, todėl atsisakyta UDDI registro.

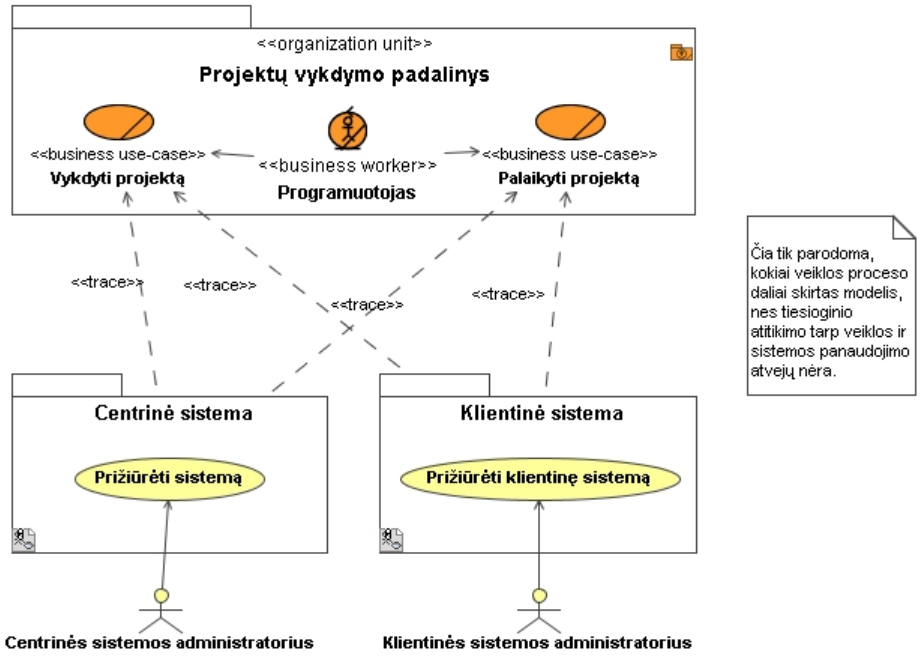
Atnaujinimo saugumo analizės pagrindu nustatyta, kad saugumui užtikrinti būtinai reikia įgyvendinti sistemų, su kuriomis bendraujama, autentifikaciją bei svarbiausių perduodamų duomenų šifravimą. Saugiausias būdas tiek autentifikavimui, tiek šifravimui – privataus ir viešojo raktų deriniu paremti algoritmai.

Analizės metu suformuluoti tyrimo uždaviniai, kurių tikslas – sukurti centralizuoto informacijos sistemų atnaujinimo paslaugų modelį ir programinį prototipą, kuris užtikrintų mažesnes atnaujinimo laiko sąnaudas, būtų patogus galutiniam vartotojui ir turėtų patikimas kokybines charakteristikas.

3. CENTRALIZUOTO ATNAUJINIMO SISTEMOS REIKALAVIMAI

Reikalavimų specifikacija yra skirta formaliai aprašyti sistemos elgseną bei pradinį dalykinės srities modelį, t. y. funkcinius sistemos reikalavimus. Taip pat joje aprašomos kokybinės ir kiekybinės savybės, t. y. nefunkciniai reikalavimai, kuriuos turi tenkinti sistema.

Kadangi šiame darbe modeliuojamos dvi – centrinė bei klientinė – sistemos ir jų tarpusavio sąveika, reikalavimai joms pateikiami atskirai. Prieš tai 8 pav. pateikiama perėjimo nuo veiklos prie reikalavimų diagrama. Joje parodoma ne tiesioginis atitikimas tarp veiklos panaudojimo atvejų ir sistemos panaudojimo atvejų, o tik pavaizduojama, kuriems įmonės veiklos proceso etapams modelis yra skirtas. Kuriamas modelis paveikia projektų vykdymą bei palaikymą.



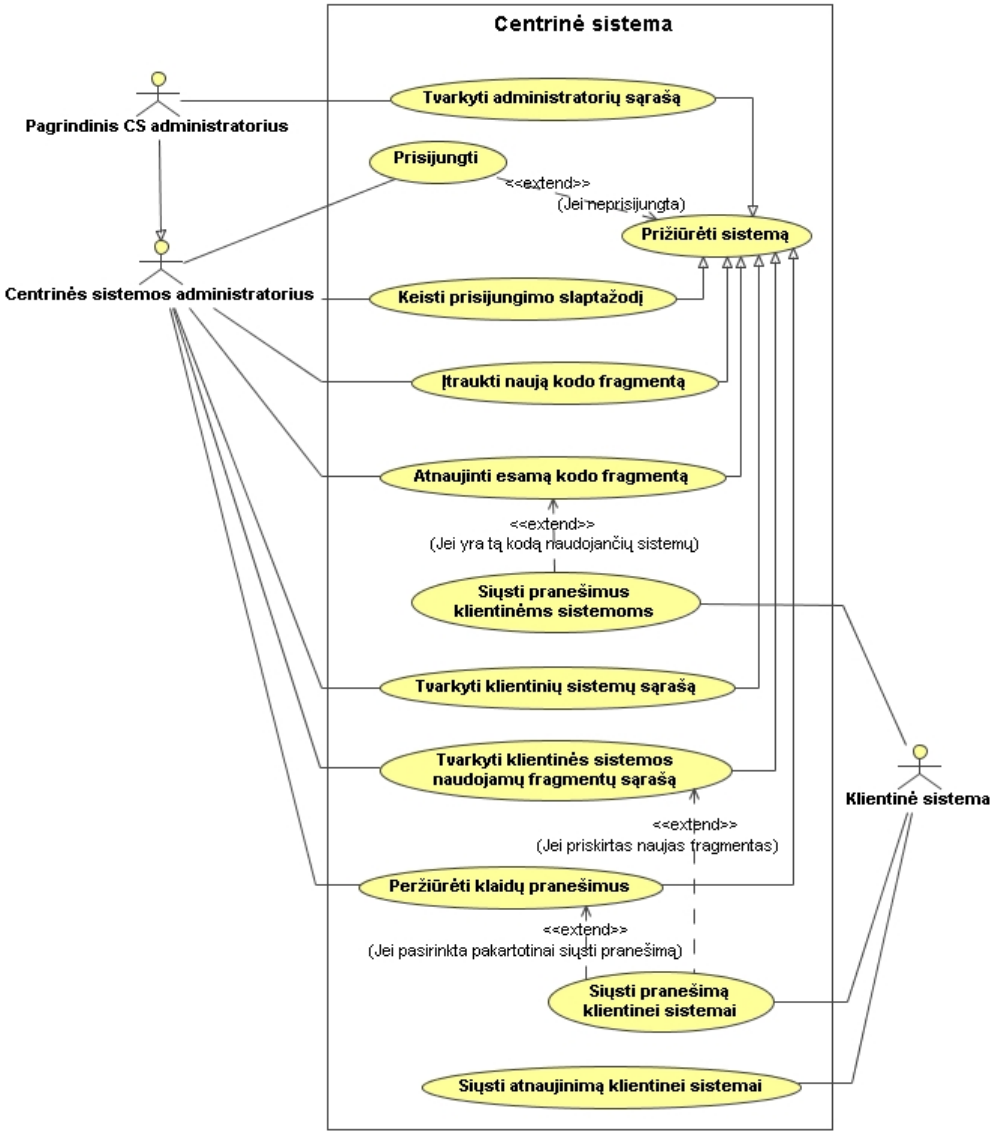
8 pav. Perėjimo nuo veiklos modelio prie reikalavimų modelio diagrama

3.1. Funkciniai reikalavimai

Šiame poskyryje atskirai aprašomi centrinės ir klientinės sistemų funkciniai reikalavimai panaudojimo atvejų diagramų pavidalu. Kiekvienas panaudojimo atvejis yra specifikuojamas struktūrizuota lentele. Jose ne specialistui suprantama kalba išdėstoma, kokia yra kiekvieno panaudojimo atvejo įvykių seka, su kokiais kitais panaudojimo atvejais jis siejasi, kokie yra alternatyvūs žingsniai. Esminiai panaudojimo atvejai taip pat specifikuojami veiklos diagramomis, o sudėtingesni panaudojimo atvejai – ir sekų diagramomis.

3.1.1. Centrinės sistemos funkciniai reikalavimai

9 pav. pateikta centrinės sistemos panaudojimo atvejų diagrama. Joje pateikiami visi pagrindiniai veiksmai, kuriuos gali atlikti centrinės sistemos administratorius. Klientinė sistema čia laikoma aktoriumi.



9 pav. Centrinės sistemos panaudojimo atvejų modelis

3 lentelėje pateikiama panaudojimo atvejo „Prisijungti“ specifikacija struktūrizuota lentele. Šis panaudojimo atvejis skirtas administratoriui prisijungti prie centrinės sistemos, kad būtų galima vykdyti kitus panaudojimo atvejus.

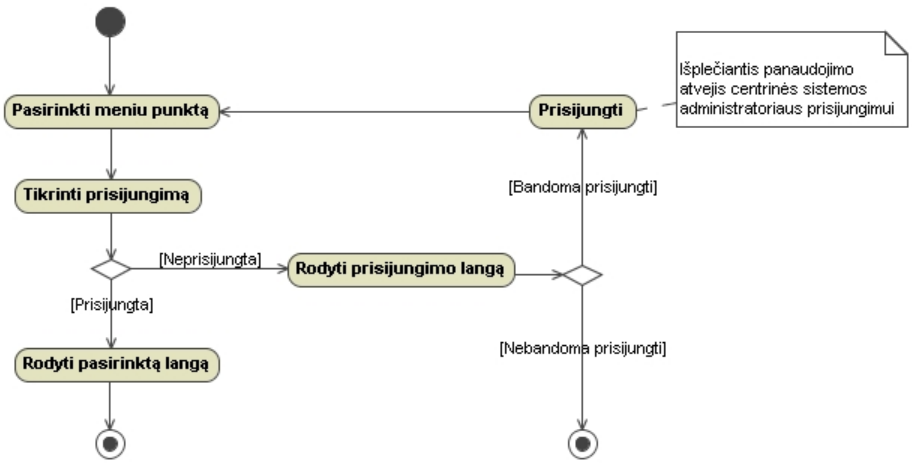
3 lentelė. Centrinės sistemos panaudojimo atvejo „Prisijungti“ specifikacija

PA „**Prisijungti**“	
Prieš sąlyga	Administratorius nėra prisijungęs.
Sužadinimo sąlyga	Administratorius nori prisijungti.
Susiję panaudojimo atvejai	Šis PA išplečia PA „Prižiūrėti sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius įveda prisijungimo vardą bei slaptažodį ir patvirtina.	**1.1.** Sistema tikrina, ar įvesti duomenys teisingi. Jei taip, administratorius laikomas prisijungusiu. Kitu atveju laikoma, kad prisijungimas nepavyko.
Po sąlyga:	Administratorius yra prisijungęs (arba ne).

4 lentelėje pateikiama panaudojimo atvejo „Prižiūrėti sistemą“ specifikacija struktūrizuota lentele, o 10 pav. – veiklos diagramos pavidalu. Tai yra apibendrinantis panaudojimo atvejis, kurio veiksmai vykdomi visuose jį specializuojančiuose panaudojimo atvejuose. Kiekviename jų vietoje veiklos „Rodyti pasirinktą langą“ vykdoma specifinė veikla.

4 lentelė. Centrinės sistemos panaudojimo atvejo „Prižiūrėti sistemą“ specifikacija

PA „**Prižiūrėti sistemą**“	
Prieš sąlyga	Administratorius įrašytas sistemos duomenų bazėje.
Sužadinimo sąlyga	Administratorius nori vykdyti kokius nors veiksmus centrinėje sistemoje.
Susiję panaudojimo atvejai	Šis PA turi išplečiantį PA „Prisijungti“ (išplėtimo taškas – „Jei neprisijungta“) bei apibendrina šiuos PA: „Įtraukti naują kodo fragmentą“, „Atnaujinti esamą kodo fragmentą“, „Tvarkyti klientinių sistemų sąrašą“, „Peržiūrėti klaidų pranešimus“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius atidaro kurį nors sistemos langą (įrašo naršyklės adreso lauke arba paspaudžia atitinkamą nuorodą).	**1.1.** Sistema tikrina, ar administratorius prisijungęs. Jei taip, šio PA vykdymas baigiamas ir iš karto vykdomas vienas iš specializuojančių PA. Priešingu atveju pasiūloma prisijungti.
2. Administratorius pasirenka, ar prisijungti, ar ne.	**2.1.** Jei pasirinkta prisijungti, vykdomas išplečiantis PA „Prisijungti“ ir grįžtama į **1.1** žingsnį. Priešingu atveju PA vykdymas baigiamas.
Po sąlyga:	Pereita prie specializuojančio PA arba nevykdomi jokie veiksmai.
Alternatyvūs scenarijai	
2. Jei administratorius nėra prisijungęs ir nori prisijungti, **išplėtimo taškas** „Jei neprisijungta“.	**2.1.** Vykdomas išplėtimo PA „**Prisijungti**“. **Po sąlyga:** administratorius yra prisijungęs.

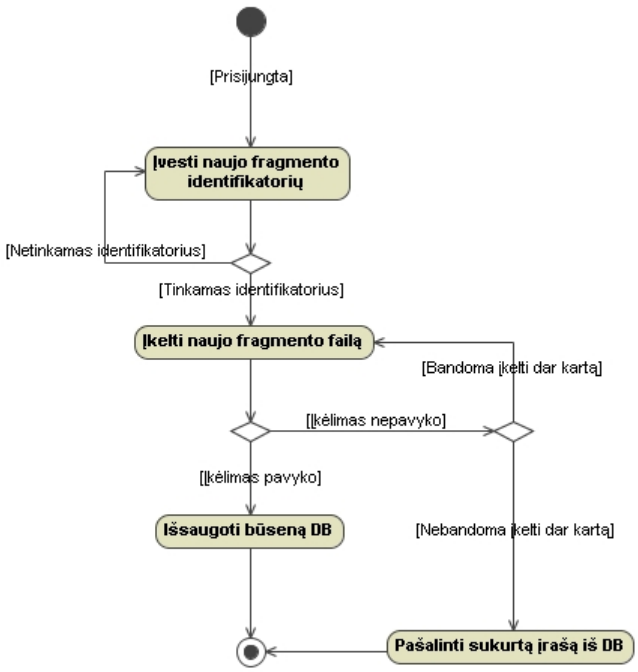


10 pav. Panaudojimo atvejo „Prižiūrėti sistemą“ veiklos modelis

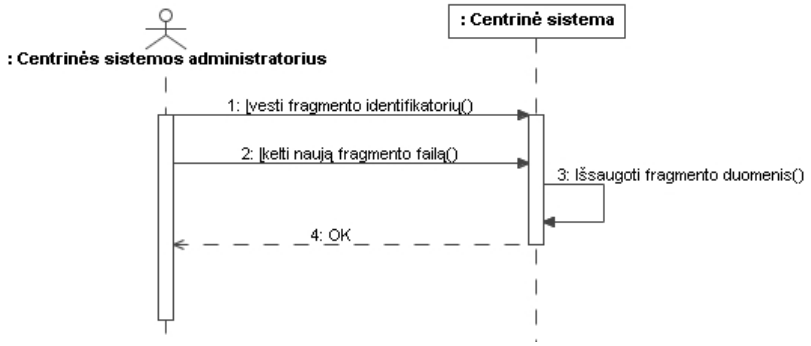
5 lentelėje pateikiama panaudojimo atvejo „Įtraukti naują kodo fragmentą“ specifikacija struktūrizuota lentele, o 11 pav. – veiklos diagramos pavidalu. 12 pav. parodyta to paties panaudojimo atvejo sekų diagrama. Šis panaudojimo atvejis skirtas įvesti naujo programinio kodo fragmento tekstinį identifikatorių ir įkelti bei išsaugoti tekstinį kodo failą.

5 lentelė. Centrinės sistemos panaudojimo atvejo „Įtraukti naują kodo fragmentą“ specifikacija

PA „**Įtraukti naują kodo fragmentą**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori įtraukti naują kodo fragmentą.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius įveda naujo fragmento identifikatorių.	**1.1.** Sistema tikrina, ar identifikatorius tinkamas, ar tokio dar nėra sistemoje. Jei identifikatorius netinkamas arba jau egzistuoja, grįžtama į **1** žingsnį. Kitu atveju identifikatorius išsaugomas DB.
2. Administratorius įkelia naujo fragmento failą.	**2.1.** Jei įkėlimas pavyko, DB išsaugoma nuoroda į įkeltą failą ir PA vykdymas baigiamas. Kitu atveju leidžiama bandyti įkelti dar kartą (vykdomas **3** žingsnis).
3. Administratorius pasirenka, ar bandyti dar kartą, ar ne.	**3.1.** Jei pasirinkta bandyti dar kartą, grįžtama prie **2** žingsnio. Kitu atveju iš DB pašalinamas įvestas identifikatorius ir PA vykdymas baigiamas.
Po sąlyga:	Įtrauktas naujas kodo fragmentas.



11 pav. Panaudojimo atvejo „Įtraukti naują kodo fragmentą“ veiklos modelis

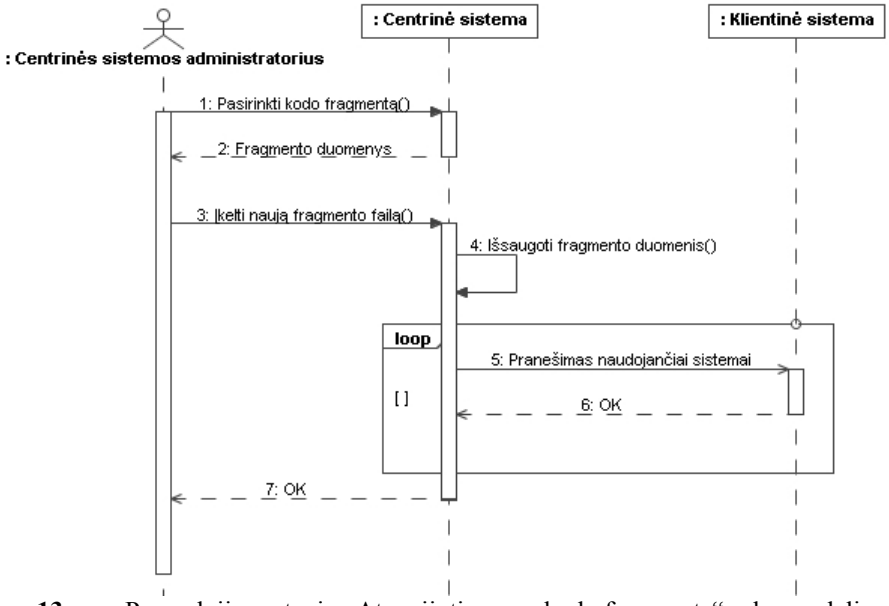


12 pav. Panaudojimo atvejo „Įtraukti naują kodo fragmentą“ sekų modelis

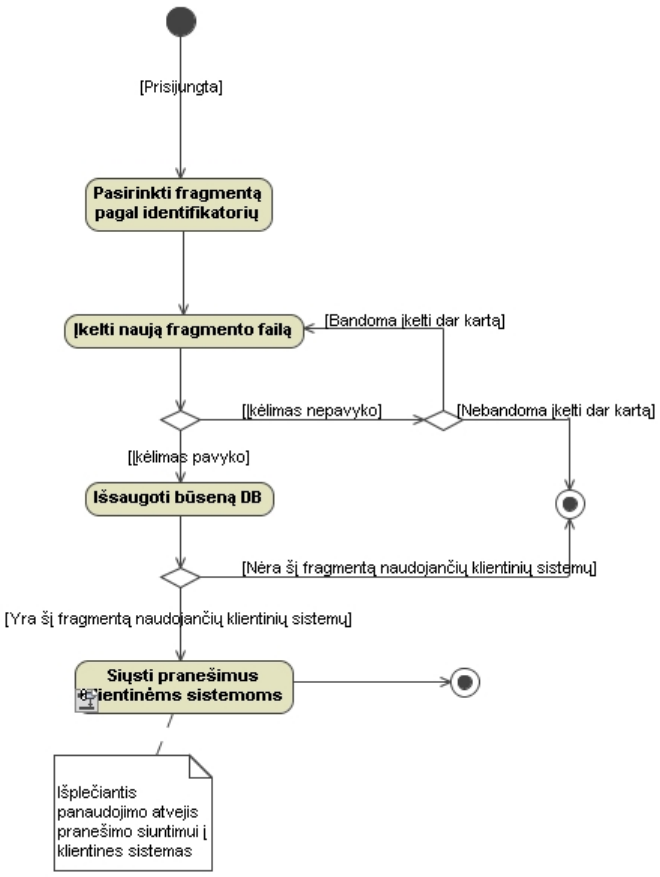
6 lentelėje pateikiama panaudojimo atvejo „Atnaujinti esamą kodo fragmentą“ specifikacija struktūrizuota lentele. 13 pav. parodyta to paties panaudojimo atvejo sekų diagrama, o 14 pav. – veiklos diagrama. Šio panaudojimo atvejo vykdymo metu pasirenkamas vienas iš įvestų programinio kodo fragmentų bei įkeliamas naujas failas. Atnaujinus fragmentą, jei yra jį naudojančių klientinių sistemų, įtraukiamas išplečiantis panaudojimo atvejis „Siųsti pranešimus klientinėms sistemoms“.

6 lentelė. Centrinės sistemos panaudojimo atvejo „Atnaujinti esamą kodo fragmentą“ specifikacija

PA „**Atnaujinti esamą kodo fragmentą**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori atnaujinti vieną iš esamų kodo fragmentų.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“ bei turi išplečiantį PA „Siųsti pranešimus klientinėms sistemoms“ (išplėtimo taškas – „Jei yra tą kodą naudojančių sistemų“).
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pasirinkti fragmentą pagal identifikatorių iš pateikto sąrašo.	**1.1.** Sistema pateikia naujo failo įkėlimo formą.
2. Administratorius įkelia naują fragmento failą.	**2.1.** Jei įkėlimas pavyko, DB išsaugoma nuoroda į įkeltą failą ir pereinama prie **2.2** žingsnio. Kitu atveju leidžiama bandyti įkelti dar kartą (vykdomas **3** žingsnis).
	2.2. Sistema tikrina, ar yra šį fragmentą naudojančių klientinių sistemų. Jei taip, tuomet vykdomas išplėtimo PA „Siųsti pranešimus klientinėms sistemoms“.
3. Administratorius pasirenka, ar bandyti dar kartą, ar ne.	**3.1.** Jei pasirinkta bandyti dar kartą, grįžtama prie **2** žingsnio. Kitu atveju PA vykdymas baigiamas.
Po sąlyga:	Atnaujintas kodo fragmentas bei išsiųsti (jei reikia) pranešimai klientinėms sistemoms.
Alternatyvūs scenarijai	
2. Jei yra fragmentą naudojančių sistemų, **išplėtimo taškas** „Jei yra tą kodą naudojančių sistemų“.	**2.1.** Vykdomas išplėtimo PA „**Siųsti pranešimus klientinėms sistemoms**“. **Po sąlyga:** pranešimai klientinėms sistemoms išsiųsti.



13 pav. Panaudojimo atvejo „Atnaujinti esamą kodo fragmentą“ sekų modelis



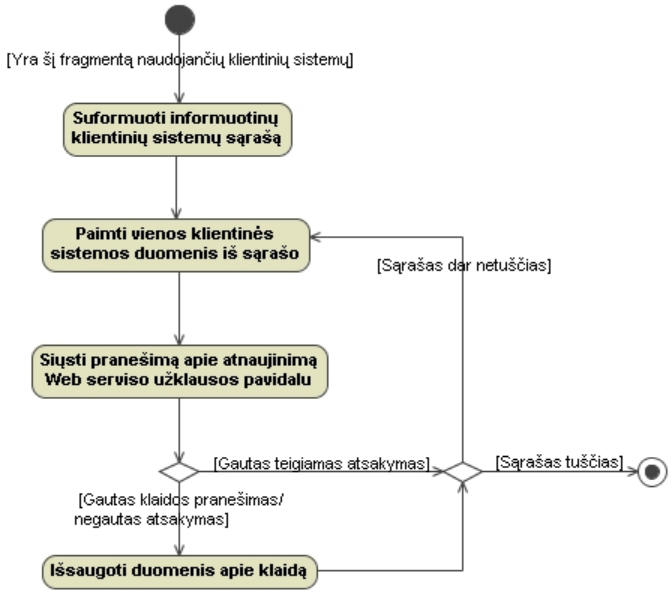
14 pav. Panaudojimo atvejo „Atnaujinti esamą kodo fragmentą“ veiklos modelis

7 lentelė. Centrinės sistemos panaudojimo atvejo „Siųsti pranešimus klientinėms sistemoms“ specifikacija

PA „**Siųsti pranešimus klientinėms sistemoms**“	
Prieš sąlyga	Administratorius yra prisijungęs. Yra kodo fragmentą naudojančių sistemų.
Sužadinimo sąlyga	Administratorius atnaujino kodo fragmentą.
Susiję panaudojimo atvejai	Šis PA išplečia PA „Atnaujinti esamą kodo fragmentą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Šis PA pilnai automatinis, tad administratoriaus veiksmų jame nėra.	**1.1.** Sudaromas klientinių sistemų, kurioms reikia siųsti pranešimą, sąrašas.
	1.2. Suformuojamas tinklo paslaugos užklausos pranešimas vienai iš sąraše esančių sistemų.
	1.3. Siunčiamas pranešimas tinklo paslaugos užklausos pavidalu ir laukiama atsakymo.
	1.4. Jei atsakymas neigiamas arba visai negautas, išsaugomi duomenys apie klaidą.
	1.5. Jei visas sąrašas pereitas, PA vykdymas baigiamas. Kitu atveju grįžtama prie **1.2** žingsnio.
Po sąlyga:	Išsiųsti pranešimai apie atnaujinimą klientinėms sistemoms.

7 lentelėje pateikiama panaudojimo atvejo „Siųsti pranešimus klientinėms sistemoms“ specifikacija struktūrizuota lentele, o 15 pav. – veiklos diagramos pavidalu. Šis panaudojimo atvejis

vykdomas tik tuomet, jei yra atnaujinamas vienas iš kodo fragmentų, ir yra bent viena jį naudojanti klientinė sistema. Pranešimai siunčiami tinklo paslaugų užklausų pavidalu. Nepavykus išsiųsti pranešimo, kiekvienas toks atvejis užregistruojamas klaidų sąraše.



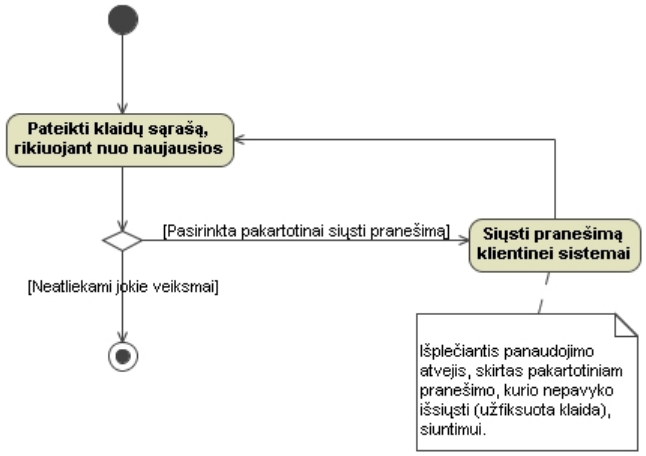
15 pav. Panaudojimo atvejo „Siųsti pranešimus klientinėms sistemoms“ veiklos modelis

8 lentelė. Centrinės sistemos panaudojimo atvejo „Peržiūrėti klaidų pranešimus“ specifikacija

PA „**Peržiūrėti klaidų pranešimus**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori peržiūrėti informaciją apie užfiksuotas klaidas.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“ bei turi išplečiantį PA „Siųsti pranešimą klientinei sistemai“ (išplėtimo taškas – „Jei pasirinkta pakartotinai siųsti pranešimą“).
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius peržiūri pateiktą užfiksuotų klaidų (nepavykusių pranešimų siuntimų) sąrašą.	**1.1.** Jei administratorius pasirinko pakartotinai siųsti pranešimą klientinei sistemai, vykdomas išplėtimo PA „Siųsti pranešimą klientinei sistemai“ ir grįžtama į **1** žingsnį. Kitu atveju PA vykdymas baigiamas.
Po sąlyga:	Peržiūrėtos klaidos ir (nebūtinai) išsiųsti pakartotiniai pranešimai.
Alternatyvūs scenarijai	
1. Jei pasirinkta siųsti nepavykusį pranešimą dar kartą, **išplėtimo taškas** „Jei pasirinkta pakartotinai siųsti pranešimą“.	**1.1.** Vykdomas išplėtimo PA „**Siųsti pranešimą klientinei sistemai**“. **Po sąlyga:** pranešimas klientinei sistemai išsiųstas.

8 lentelėje pateikiama panaudojimo atvejo „Peržiūrėti klaidų pranešimus“ specifikacija struktūrizuota lentele, o 16 pav. – veiklos diagramos pavidalu. Šis panaudojimo atvejis skirtas

siunčiant pranešimus klientinėms sistemoms užfiksuotų klaidų sąrašo peržiūrai. Galima pasirinktai klientinei sistemai pranešimą siųsti pakartotinai, tokiu būdu ištaisant klaidą.

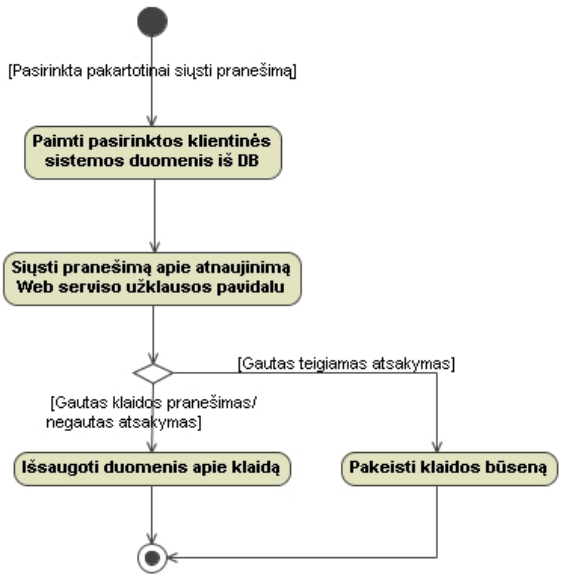


16 pav. Panaudojimo atvejo „Peržiūrėti klaidų pranešimus“ veiklos modelis

9 lentelėje pateikiama panaudojimo atvejo „Siųsti pranešimą klientinei sistemai“ specifikacija struktūrizuota lentele, o 17 pav. – veiklos diagramos pavidalu. Šis panaudojimo atvejis vykdomas tuomet, jei peržiūrint klaidų sąrašą pasirenkama pakartotinai siųsti pranešimą pasirinktai klientinei sistemai. Siuntimas vykdomas analogiškai tam, kuris vykdomas atnaujinus kodo fragmentą. Įvykdžius siuntimą atnaujinami klaidos duomenys, priklausomai nuo to, ar pranešimas nusiųstas sėkmingai, ar ne.

9 lentelė. Centrinės sistemos panaudojimo atvejo „Siųsti pranešimą klientinei sistemai“ specifikacija

PA „**Siųsti pranešimą klientinei sistemai**“	
Prieš sąlyga	Administratorius yra prisijungęs. Užfiksuota klaida dėl nepavykusio pranešimo siuntimo klientinei sistemai.
Sužadinimo sąlyga	Administratorius nori pakartotinai siųsti pranešimą.
Susiję panaudojimo atvejai	Šis PA išplečia PA „Peržiūrėti klaidų pranešimus“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Šis PA pilnai automatinis, tad administratoriaus veiksmų jame nėra.	**1.1.** Paimami klientinės sistemos duomenys iš DB.
	1.2. Suformuojamas tinklo paslaugos užklausos pranešimas tai sistemai.
	1.3. Siunčiamas pranešimas tinklo paslaugos užklausos pavidalu ir laukiama atsakymo.
	1.4. Jei atsakymas neigiamas arba visai negautas, išsaugomi duomenys apie klaidą. Kitu atveju klaidos būsena pakeičiama į „ištaisytą“.
Po sąlyga:	Išsiųstas pranešimas apie atnaujinimą klientinei sistemai.



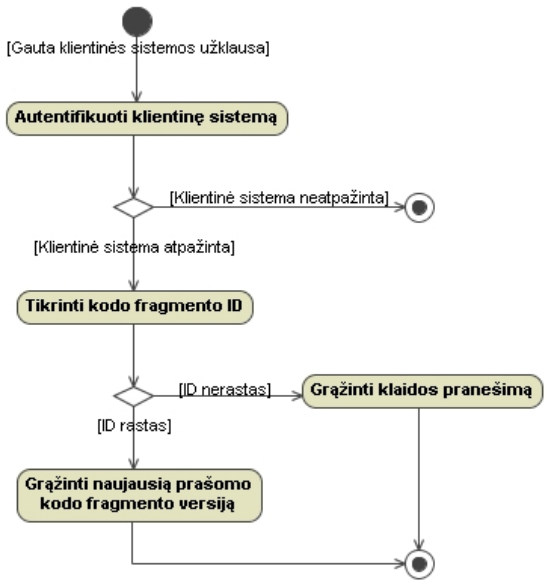
17 pav. Panaudojimo atvejo „Siųsti pranešimą klientinei sistemai“ veiklos modelis

10 lentelėje pateikiama panaudojimo atvejo „Siųsti atnaujinimą klientinei sistemai“ specifikacija struktūrizuota lentele, o 18 pav. – veiklos diagramos pavidalu. Šis panaudojimo atvejis vykdomas tiktai tuomet, kai gaunama atitinkama klientinės sistemos tinklo paslaugos užklausa. Pirmiausiai klientinė sistema yra autentifikuojama, kad būtų išvengta kodo siuntimo neatpažintoms sistemoms. Tuomet tikrinama, ar atnaujinimo reikalaujanti klientinė sistema naudoja prašomą kodo fragmentą, ir, jei naudoja, grąžinama naujausia kodo fragmento versija. Priešingu atveju klientinei sistemai grąžinamas klaidos pranešimas.

Veikla „Autentifikuoti klientinę sistemą“ – tai privataus-viešojo raktų deriniu paremtas užklausos siuntėjo atpažinimas. Analogiškai klientinėse sistemose autentifikuojama centrinė sistema.

10 lentelė. Centrinės sistemos panaudojimo atvejo „Siųsti atnaujinimą klientinei sistemai“ specifikacija

PA „**Siųsti atnaujinimą klientinei sistemai**“	
Prieš sąlyga	Gauta kodo fragmento atnaujinimo užklausa iš klientinės sistemos.
Sužadinimo sąlyga	Gauta kodo fragmento atnaujinimo užklausa iš klientinės sistemos.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Gauta klientinės sistemos užklausa.	**1.1.** Autentifikuojamas užklausos siuntėjas. Jei sistema neatpažinta, PA vykdymas baigiamas.
	1.2. Tikrinama, ar egzistuoja kodo fragmentas su pageidaujamu identifikatoriumi. Jei taip, tinklo paslaugos užklausos atsakyme grąžinama naujausia kodo fragmento versija. Kitu atveju grąžinamas klaidos pranešimas.
Po sąlyga:	Išsiųstas atnaujintas kodo fragmentas klientinei sistemai.



18 pav. Panaudojimo atvejo „Siųsti atnaujinimą klientinei sistemai“ veiklos modelis

11 lentelėje pateikiama panaudojimo atvejo „Tvarkyti klientinių sistemų sąrašą“ specifikacija struktūrizuota lentele. Šis panaudojimo atvejis skirtas redaguoti anksčiau įvestų bei įvesti naujų klientinių sistemų duomenis, tokius kaip adresas pranešimų siuntimui ir viešas raktas autentifikavimui.

11 lentelė. Centrinės sistemos panaudojimo atvejo „Tvarkyti klientinių sistemų sąrašą“ specifikacija

PA „**Tvarkyti klientinių sistemų sąrašą**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori tvarkyti klientinių sistemų sąrašą.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pateiktame klientinių sistemų sąraše administratorius gali redaguoti įvedimo laukuose pateiktus duomenis (viešą raktą, adresą ir pan.).	
2. Sąrašo gale pateiktuose tuščiuose įvedimo laukuose įveda (nebūtinai) naujų sistemų duomenis.	
3. Patvirtina pakeitimus.	**3.1.** Sistema išsaugo kiekvieną įrašą. Jei yra įvesta naujų įrašų, jie irgi išsaugomi.
Po sąlyga:	Išsaugoti klientinių sistemų sąrašo pakeitimai.

12 lentelėje pateikiama panaudojimo atvejo „Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą“ specifikacija struktūrizuota lentele. Šio panaudojimo atvejo vykdymo metu pasirenkama klientinė sistema bei pateiktame fragmentų sąraše sužymimi tie fragmentai, kuriuos ji naudoja. Apie kiekvieną naujai priskirtą fragmentą klientinei sistemai yra siunčiamas pranešimas.

12 lentelė. Centrinės sistemos panaudojimo atvejo „Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą“ specifikacija

PA „**Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori pakoreguoti klientinės sistemos naudojamų kodo fragmentų sąrašą.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“ bei turi išplečiantį PA „Siųsti pranešimą klientinei sistemai“ (išplėtimo taškas – „Jei priskirtas naujas fragmentas“).
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Iš duoto klientinių sistemų sąrašo administratorius išsirenka vieną.	**1.1.** Sistema suformuoja fragmentų sąrašą su žymėjimo laukeliais (angl. checkbox).
2. Gali pažymėti naujus, išvalyti senus ar palikti nekeistus laukelius.	
3. Patvirtina pakeitimus.	**3.1.** Sistema išsaugo pakeitimus. Kiekvienam naujai priskirtam fragmentui yra vykdomas išplėtimo PA „Siųsti pranešimą klientinei sistemai“. PA vykdymas baigiamas.
Po sąlyga:	Išsaugoti klientinės sistemos naudojamų fragmentų sąrašo pakeitimai.
Alternatyvūs scenarijai	
1. Kiekvienam naujai priskirtam fragmentui **išplėtimo taškas** „Jei priskirtas naujas fragmentas“.	**1.1.** Vykdomas išplėtimo PA „**Siųsti pranešimą klientinei sistemai**“. **Po sąlyga:** pranešimas klientinei sistemai išsiųstas.

13 lentelė. Centrinės sistemos panaudojimo atvejo „Tvarkyti administratorių sąrašą“ specifikacija

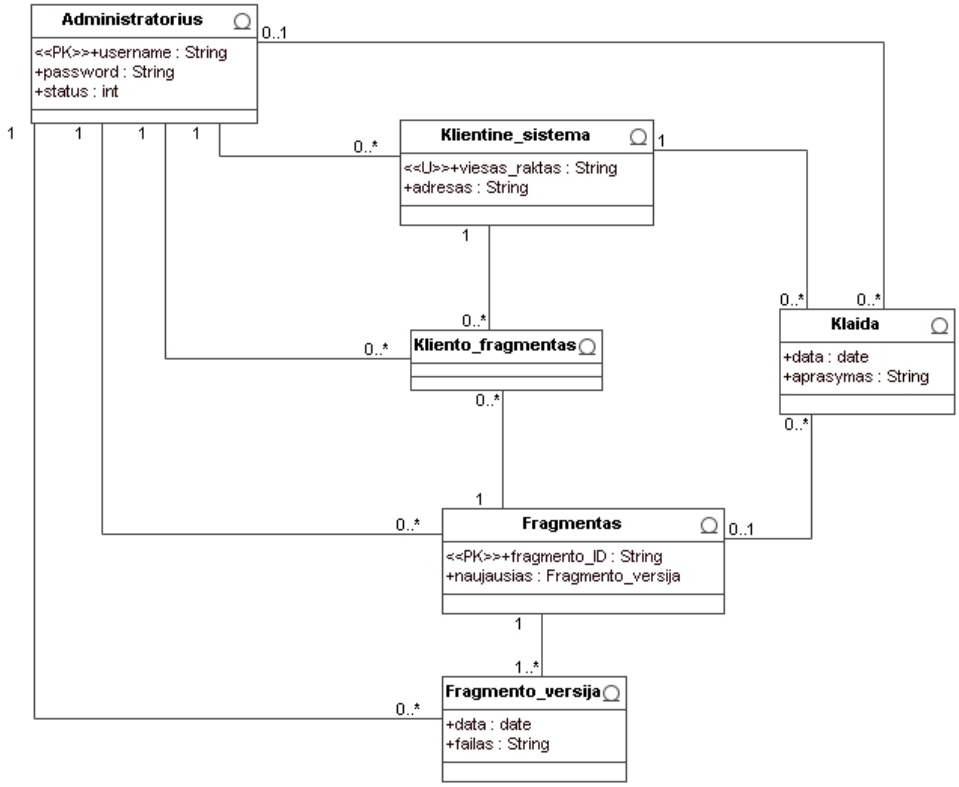
PA „**Tvarkyti administratorių sąrašą**“	
Prieš sąlyga	Administratorius yra prisijungęs ir jis yra pagrindinis administratorius.
Sužadinimo sąlyga	Administratorius nori tvarkyti administratorių sąrašą.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pateiktame administratorių sąraše administratorius gali redaguoti įvedimo laukuose pateiktus duomenis (prisijungimo slaptažodžius) ARBA paspausti vieną šalinimo nuorodą.	**1.1.** Jei paspausta šalinimo nuoroda, ją atitinkančio administratoriaus duomenys pašalinami iš sistemos ir grįžtama į **1** žingsnį. Kitu atveju einama į **2** žingsnį.
2. Sąrašo gale pateiktuose tuščiuose įvedimo laukuose įveda (nebūtinai) naujų administratorių duomenis.	
3. Patvirtina pakeitimus.	**3.1.** Sistema išsaugo kiekvieną įrašą. Jei yra įvesta naujų įrašų, jie irgi išsaugomi.
Po sąlyga:	Išsaugoti administratorių sąrašo pakeitimai.

13 lentelėje pateikiama panaudojimo atvejo „Tvarkyti administratorių sąrašą“ specifikacija struktūrizuota lentele. Šis panaudojimo atvejis skirtas redaguoti ar šalinti esamų bei įvesti naujų administratorių duomenis, tokius kaip prisijungimo vardas (tik įvedant naujus) bei slaptažodis.

14 lentelėje pateikiama panaudojimo atvejo „Keisti prisijungimo slaptažodį“ specifikacija struktūrizuota lentele. Jis skirtas prisijungusiam administratoriui pakeisti savo slaptažodį.

14 lentelė. Centrinės sistemos panaudojimo atvejo „Keisti prisijungimo slaptažodį“ specifikacija

PA „**Keisti prisijungimo slaptažodį**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori pakeisti prisijungimo slaptažodį.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius įveda naują slaptažodį.	
2. Įveda naują slaptažodį dar kartą.	
3. Patvirtina pakeitimus.	**3.1.** Jei įvesti slaptažodžiai sutampa bei yra tinkami, slaptažodis išsaugomas ir PA baigiamas. Kitu atveju rodomas pranešimas apie klaidą ir grįžtama į **1** žingsnį.
Po sąlyga:	Pakeistas prisijungusio administratoriaus slaptažodis.



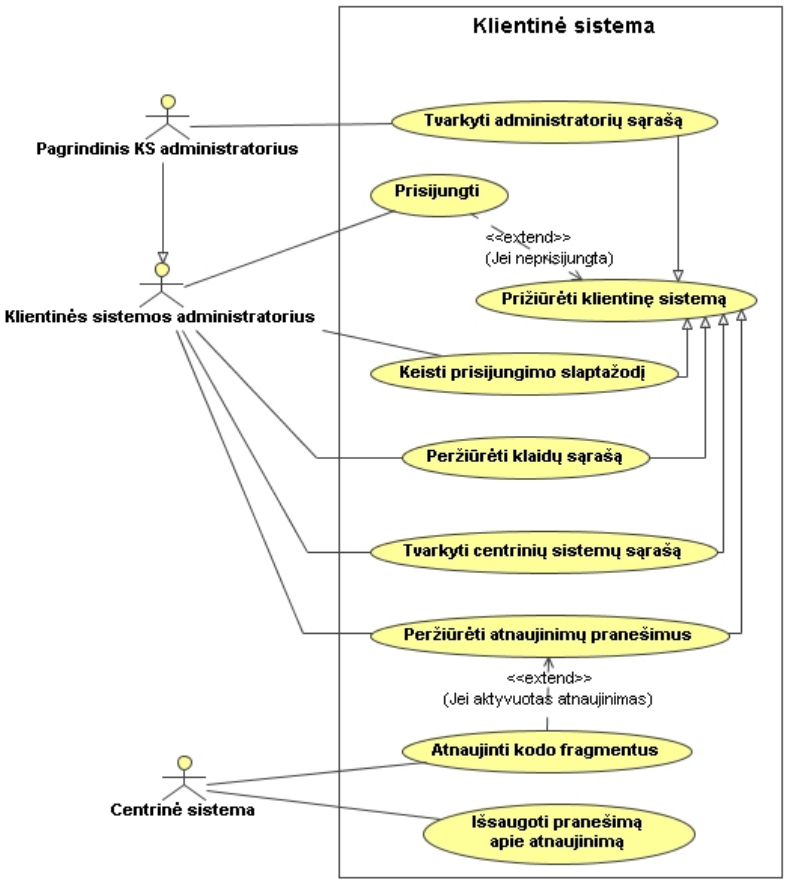
19 pav. Centrinės sistemos klasių modelis

19 pav. parodytas centrinės sistemos dalykinės srities modelis. Bendru atveju centrinėje sistemoje gali būti ne vienas administratorius, todėl išskirta administratoriaus esybė (klasė). Kiekviena klientinė sistema gali turėti daug kodo fragmentų, o kiekvienas fragmentas gali būti naudojamas daugelyje klientinių sistemų. Tam įvesta papildoma klasė „Kliento_fragmentas“. Kiekvienas kodo fragmentas turi turėti vieną (pradinę) ar daugiau versijų. Siunčiant konkrečiai klientinei sistemai konkretaus fragmento atnaujinimo pranešimą, siuntimas gali nepavykti. Dėl to sukurta klaidos esybė, kuria remiantis vėliau pranešimą galima išsiųsti pakartotinai.

3.1.2. Klientinės sistemos funkciniai reikalavimai

Klientinės sistemos panaudojimo atvejų modelis pateiktas 20 pav. Čia (analogiškai centrinės sistemos panaudojimo atvejų modeliui) parodomi pagrindiniai veiksmai, kuriuos gali atlikti klientinės sistemos administratorius. Šioje diagramoje centrinė sistema laikoma aktoriumi.

Pastaba: klientinės sistemos panaudojimo atvejų bei kituose modeliuose nenagrinėjama, kokius specifinius sistemos turinio valdymo veiksmus gali atlikti administratorius, kadangi kiekviena užsakovo informacijos sistema skiriasi nuo kitų. Nagrinėjami tik esminiai kuriamo modelio elementai.



20 pav. Klientinės sistemos panaudojimo atvejų modelis

15 lentelėje pateikiama klientinės sistemos panaudojimo atvejo „Prisijungti“ specifikacija struktūrizuota lentele. Šis panaudojimo atvejis skirtas administratoriui prisijungti prie klientinės sistemos, kad būtų galima vykdyti kitus panaudojimo atvejus.

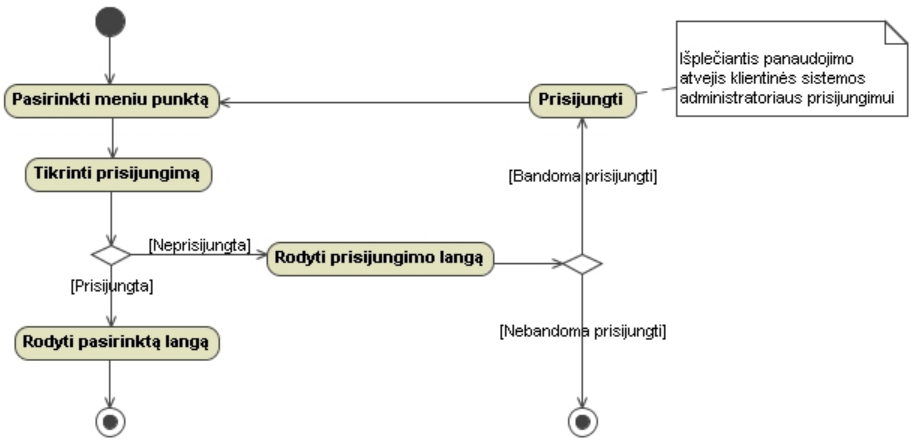
15 lentelė. Klientinės sistemos panaudojimo atvejo „Prisijungti“ specifikacija

PA „**Prisijungti**“	
Prieš sąlyga	Administratorius nėra prisijungęs.
Sužadinimo sąlyga	Administratorius nori prisijungti.
Susiję panaudojimo atvejai	Šis PA išplečia PA „Prižiūrėti klientinę sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius įveda prisijungimo vardą bei slaptažodį ir patvirtina.	**1.1.** Sistema tikrina, ar įvesti duomenys teisingi. Jei taip, administratorius laikomas prisijungusiu. Kitu atveju laikoma, kad prisijungimas nepavyko.
Po sąlyga:	Administratorius yra prisijungęs (arba ne).

16 lentelėje pateikiama panaudojimo atvejo „Prižiūrėti klientinę sistemą“ specifikacija struktūrizuota lentele, o 21 pav. – veiklos diagramos pavidalu. Tai yra analogiškas panaudojimo atvejis centrinės sistemos panaudojimo atvejui „Prižiūrėti sistemą“.

16 lentelė. Klientinės sistemos panaudojimo atvejo „Prižiūrėti klientinę sistemą“ specifikacija

PA „**Prižiūrėti klientinę sistemą**“	
Prieš sąlyga	Administratorius įrašytas sistemos duomenų bazėje.
Sužadinimo sąlyga	Administratorius nori vykdyti kokius nors veiksmus klientinėje sistemoje.
Susiję panaudojimo atvejai	Šis PA turi išplečiantį PA „Prisijungti“ (išplėtimo taškas – „Jei neprisijungta“) bei apibendrina PA „Aktyvuoti atnaujinimą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius atidaro kurį nors sistemos langą (įrašo naršyklės adreso lauke arba paspaudžia atitinkamą nuorodą).	**1.1.** Sistema tikrina, ar administratorius prisijungęs. Jei taip, šio PA vykdymas baigiamas ir iš karto vykdomas vienas iš specializuojančių PA. Priešingu atveju pasiūloma prisijungti.
2. Administratorius pasirenka, ar prisijungti, ar ne.	**2.1.** Jei pasirinkta prisijungti, vykdomas išplečiantis PA „Prisijungti“ ir grįžtama į **1.1** žingsnį. Priešingu atveju PA vykdymas baigiamas.
Po sąlyga:	Pereita prie specializuojančio PA arba nevykdomi jokie veiksmai.
Alternatyvūs scenarijai	
2. Jei administratorius nėra prisijungęs ir nori prisijungti, **išplėtimo taškas** „Jei neprisijungta“.	**2.1.** Vykdomas išplėtimo PA „**Prisijungti**“. **Po sąlyga:** administratorius yra prisijungęs.

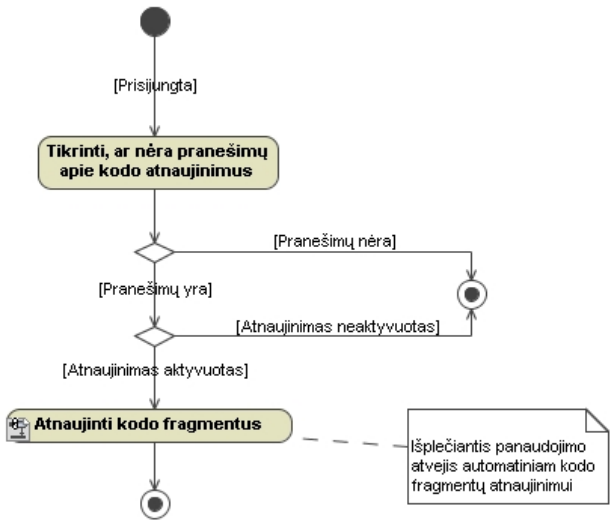


21 pav. Panaudojimo atvejo „Prižiūrėti klientinę sistemą“ veiklos modelis

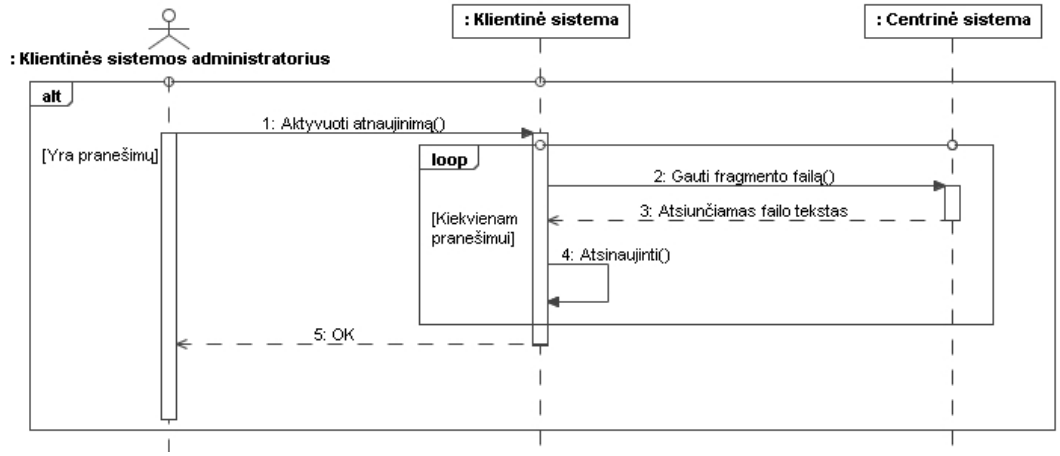
17 lentelėje pateikiama panaudojimo atvejo „Peržiūrėti atnaujinimo pranešimus“ specifikacija struktūrizuota lentele, o 22 pav. – veiklos diagramos pavidalu. 23 pav. parodyta to paties panaudojimo atvejo sekų diagrama. Šis panaudojimo atvejis skirtas peržiūrėti atsiųstų atnaujinimo pranešimų sąrašą ir, jei yra atnaujinimų, vieno mygtuko paspaudimu aktyvuoti atnaujinimų siuntimą. Tokiu atveju įtraukiamas išplečiantis panaudojimo atvejis „Atnaujinti kodo fragmentus“.

17 lentelė. Klientinės sistemos panaudojimo atvejo „Peržiūrėti atnaujinimų pranešimus“ specifikacija

PA „**Peržiūrėti atnaujinimų pranešimus**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori patikrinti, ar nėra pranešimų apie atnaujinimus. Tikrinimas taip pat vyksta ir automatiškai, vos tik administratoriui prisijungus.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti klientinę sistemą“ bei turi išplečiantį PA „Atnaujinti kodo fragmentus“ (išplėtimo taškas – „Jei aktyvuotas atnaujinimas“).
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Administratorius laukia, kol bus patikrinta, ar nėra pranešimų apie kodo atnaujinimus.	**1.1.** Sistema tikrina, ar nėra pranešimų apie kodo atnaujinimus. Jei nėra, PA vykdymas baigiamas. Kitu atveju pateikiamas atnaujinimų sąrašas bei galimybė pasirinkti, ar juos aktyvuoti, ar ne.
2. Administratorius pasirenka, ar aktyvuoti atnaujinimus, ar ne.	**2.1.** Jei pasirinkta aktyvuoti, tuomet vykdomas išplėtimo PA „Atnaujinti kodo fragmentus“. Kitu atveju PA vykdymas baigiamas.
Po sąlyga:	Atnaujinti kodo fragmentai.
Alternatyvūs scenarijai	
2. Jei yra atnaujinimo pranešimų bei atnaujinimai aktyvuoti, **išplėtimo taškas** „Jei aktyvuotas atnaujinimas“.	**2.1.** Vykdomas išplėtimo PA „**Atnaujinti kodo fragmentus**“. **Po sąlyga:** atsiųstos ir išsaugotos naujos kodo fragmentų versijos.



22 pav. Panaudojimo atvejo „Peržiūrėti atnaujinimų pranešimus“ veiklos modelis

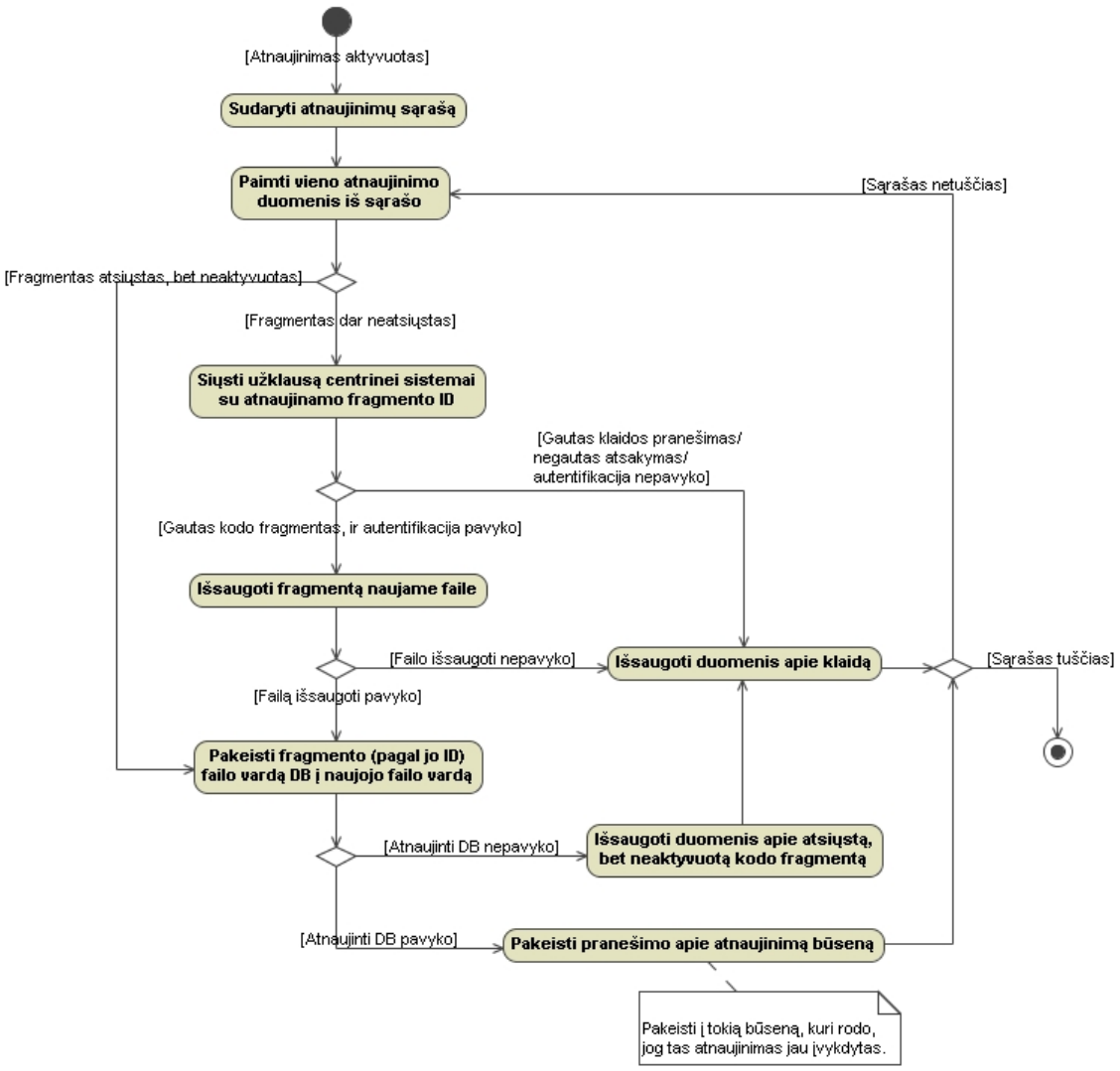


23 pav. Panaudojimo atvejo „Peržiūrėti atnaujinimų pranešimus“ sekų modelis

18 lentelėje pateikiama panaudojimo atvejo „Atnaujinti kodo fragmentus“ specifikacija struktūrizuota lentele, o 24 pav. – veiklos diagramos pavidalu. Šis panaudojimo atvejis vykdomas tada, kai yra naujų pranešimų apie atnaujinimus, o klientinės sistemos administratorius juos aktyvuoja.

18 lentelė. Klientinės sistemos panaudojimo atvejo „Atnaujinti kodo fragmentus“ specifikacija

PA „**Atnaujinti kodo fragmentus**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius aktyvavo atnaujinimus.
Susiję panaudojimo atvejai	Šis PA išplečia PA „Aktyvuoti atnaujinimą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Šis PA yra pilnai automatinis, tad administratoriaus veiklos jame nėra.	**1.1.** Sistema suformuoja atsiųstų atnaujinimo pranešimų sąrašą.
	1.2. Imamas vienas pranešimas iš sąrašo ir tikrinama, ar jo kodo failas dar nėra atsiųstas. Jei nėra, vykdomas **1.2** žingsnis. Jei yra, pereinama prie **1.6** žingsnio.
	1.3. Pranešimą atsiuntusiai centrinei sistemai siunčiama tinklo paslaugos užklausa su kodo fragmento identifikatoriumi ir laukiama atsakymo.
	1.4. Jei atsakyme yra klaidos pranešimas arba atsakymas visai negautas, išsaugomi duomenys apie klaidą bei pereinama prie **1.8** žingsnio.
	1.5. Bandoma išsaugoti atsiųstą kodą į failą. Jei nepavyksta, išsaugomi duomenys apie klaidą bei pereinama prie **1.8** žingsnio.
	1.6. Sistema bando pakeisti duomenų bazėje esančio kodo fragmento įrašo nuorodą į naująjį failą. Jei nepavyksta, tuomet išsaugomi duomenys apie atsiųstą, bet dar neaktyvuotą kodo fragmentą, išsaugomi duomenys apie klaidą bei pereinama prie **1.8** žingsnio.
	1.7. Pakeičiama pranešimo būsena į tokią, kuri rodo, jog tas atnaujinimas jau įvykdytas.
	1.8. Jei sąrašas dar nepereitas, grįžtama prie **1.2** žingsnio. Kitu atveju PA vykdymas baigiamas.
Po sąlyga:	Atnaujinti kodo fragmentai.

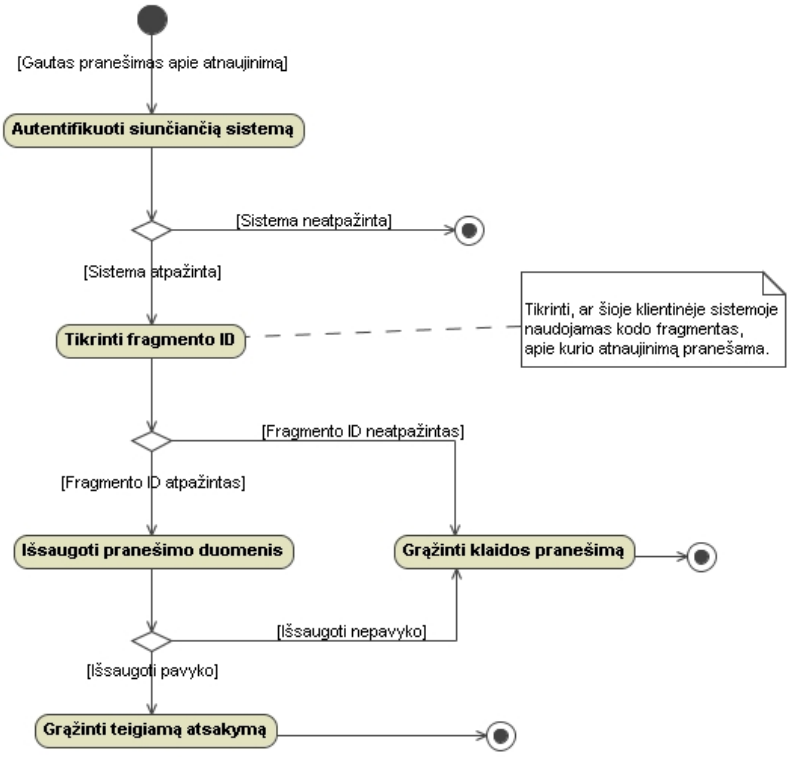


24 pav. Panaudojimo atvejo „Atnaujinti kodo fragmentus“ veiklos modelis

19 lentelėje pateikiama panaudojimo atvejo „Išsaugoti pranešimą apie atnaujinimą“ specifikacija struktūrizuota lentele, o 25 pav. – veiklos diagramos pavidalu. Šis panaudojimo atvejis vykdomas tik tada, kai centrinė sistema siunčia klientinei sistemai pranešimą apie vieno iš jos naudojamų kodo fragmentų atnaujinimą tinklo paslaugos užklausos pavidalu. Pirmiausiai autentifikuojama siunčianti sistema, kad būtų išvengta kenkėjiškų sistemų užklausų. Tuomet, jei pranešimą atitinkantis kodo fragmentas klientinėje sistemoje yra naudojamas (gali būti ir taip, kad nenaudojamas, jei centrinėje sistemoje klaidingai sužymėti klientinės sistemos naudojami fragmentai), pranešimas išsaugomas. Priešingu atveju centrinei sistemai grąžinamas pranešimas apie klaidą.

19 lentelė. Klientinės sistemos panaudojimo atvejo „Išsaugoti pranešimą apie atnaujinimą“ specifikacija

PA „**Išsaugoti pranešimą apie atnaujinimą**“		
Prieš sąlyga		Iš centrinės sistemos gautas pranešimas apie atnaujinimą tinklo paslaugos užklausos pavidalu.
Sužadinimo sąlyga		Iš centrinės sistemos gautas pranešimas apie atnaujinimą tinklo paslaugos užklausos pavidalu.
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Gauta centrinės sistemos užklausa.		**1.1.** Autentifikuojamas užklausos siuntėjas. Jei sistema neatpažinta, PA vykdymas baigiamas.
		1.2. Tikrinama, ar šioje klientinėje sistemoje naudojamas fragmentas su atsiųstu identifikatoriumi. Jei ne, centrinei sistemai grąžinamas klaidos pranešimas ir PA baigiamas vykdyti.
		1.3. Bandoma išsaugoti pranešimo duomenis. Jei nepavyksta, centrinei sistemai grąžinamas klaidos pranešimas. Kitu atveju grąžinamas teigiamas atsakymas.
Po sąlyga:		Išsaugotas atsiųstas pranešimas apie atnaujinimą.



25 pav. Panaudojimo atvejo „Išsaugoti pranešimą apie atnaujinimą“ veiklos modelis

20 lentelėje pateikiama panaudojimo atvejo „Tvarkyti centrinių sistemų sąrašą“ specifikacija struktūrizuota lentele. Šis panaudojimo atvejis skirtas redaguoti anksčiau įvestų bei įvesti naujų centrinių sistemų duomenis, tokius kaip adresas bei viešas raktas autentifikavimui.

20 lentelė. Klientinės sistemos panaudojimo atvejo „Tvarkyti centrinių sistemų sąrašą“ specifikacija

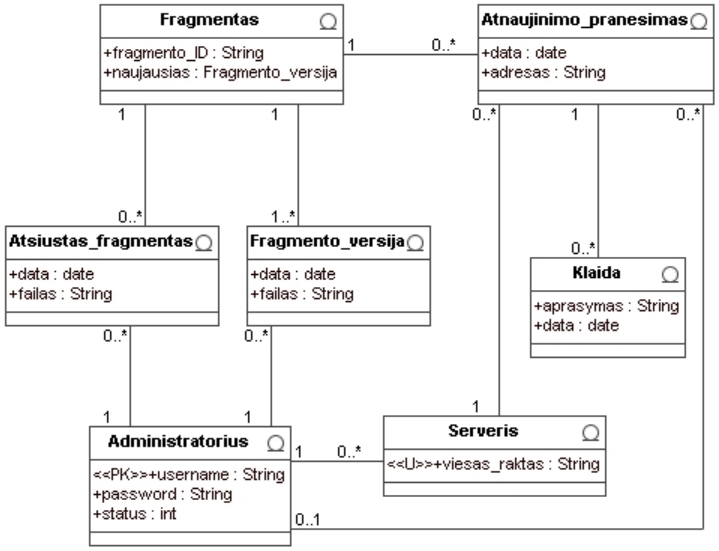
PA „**Tvarkyti centrinių sistemų sąrašą**“	
Prieš sąlyga	Administratorius yra prisijungęs.
Sužadinimo sąlyga	Administratorius nori tvarkyti centrinių sistemų sąrašą.
Susiję panaudojimo atvejai	Šis PA specializuoja PA „Prižiūrėti sistemą“.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Pateiktame centrinių sistemų sąraše administratorius gali redaguoti įvedimo laukuose pateiktus duomenis.	
2. Sąrašo gale pateiktuose tuščiuose įvedimo laukuose įveda (nebūtinai) naujų centrinių sistemų duomenis.	
3. Patvirtina pakeitimus.	**3.1.** Sistema išsaugo kiekvieną įrašą. Jei yra įvesta naujų įrašų, jie irgi išsaugomi.
Po sąlyga:	Išsaugoti centrinių sistemų sąrašo pakeitimai.

Panaudojimo atvejai „Tvarkyti administratorių sąrašą“ bei „Keisti prisijungimo slaptažodį“ yra identiški atitinkamiems centrinės sistemos panaudojimo atvejams (atitinkamai 13 lentelė ir 14 lentelė). O panaudojimo atvejis „Peržiūrėti klaidų sąrašą“ skirtas tik informacijai apie užfiksuotas klaidas pateikti, t. y. administratoriaus veiksmai nereikalingi.

Klientinės sistemos dalykinės srities modelis pateiktas 26 pav. Klasė „Serveris“ saugo viešą raktą centrinės sistemos autentifikavimui, o adresas saugomas kiekviename atnaujinimo pranešime. Taip modelis nereikalaus pakeitimų klientinėse sistemose, jei centrinės sistemos adresas pasikeis. Be to, modelis leidžia naudoti daugiau nei vieną centrinę sistemą – tokiu būdu dalis kodo fragmentų gali būti laikoma viename serveryje, kita dalis – kitame.

Kiekvienas fragmentas turi turėti bent vieną (pradinę) versiją. Jei aktyvavus atnaujinimą fragmentas atsiunčiamas, bet nepavyksta sukurti naujos fragmento versijos, tuomet duomenys apie atsiųstą fragmentą laikinai iki kito atnaujinimo aktyvavimo saugomi kaip „Atsiustas_fragmentas“. Nepavykus atsiųsti atnaujinto fragmento išsaugomas įrašas apie klaidą.

Kadangi sistemoje gali būti keletas administratorių, juos atitinkanti klasė yra susieta su kai kuriomis redaguojamomis esybėmis. Taip išsaugoma informacija apie tai, kuris administratorius kurį esybės egzempliorių sukūrė arba paskutinis pakoregavo.



26 pav. Klientinės sistemos klasių modelis

3.2. Nefunkciniai reikalavimai

Šiame poskyryje pateikiami pagrindiniai centrinės (21 lentelė) bei klientinės (22 lentelė) sistemų nefunkciniai reikalavimai.

21 lentelė. Centrinės sistemos nefunkciniai reikalavimai

Paprastas sistemos įdiegimas	
Pagrindimas:	Sistemos įdiegimas turi būti nesudėtingas.
Tinkamumo kriterijus:	Tinkamai naudojantis įdiegimo instrukcija įdiegimas pavyksta iš pirmo karto.
Sistemos perkeliamumas	
Pagrindimas:	Esant būtinybei, turi būti galimybė perkelti centrinę sistemą su visais duomenimis į kitą Web serverį.
Tinkamumo kriterijus:	Perkėlus centrinę sistemą į kitą serverį atnaujinimai neturi sutrikti.
Greitas išmokstamumas	
Pagrindimas:	Centrinės sistemos administravimas turi būti greitai išmokstamas, intuityvus.
Tinkamumo kriterijus:	Nepatyręs administratorius greitai išmoksta atlikti visus veiksmus.

22 lentelė. Klientinės sistemos nefunkciniai reikalavimai

Greitas HTML puslapių formavimas	
Pagrindimas:	Realizavus klientinę sistemą šiame darbe sukurtu modeliu, HTML puslapiai turi būti formuojami beveik tokiu pačiu greičiu kaip ir kuriant įprastais metodais.
Tinkamumo kriterijus:	HTML puslapių formavimo trukmė neturi padidėti daugiau kaip 5 %.
Kodo fragmentų išsaugojimo tikslumas	
Pagrindimas:	Kiekvienas atsiųstas atnaujintas kodo fragmentas turi būti išsaugotas tiksliai toks, koks atsiųstas.
Tinkamumo kriterijus:	100 % atsiųstų fragmentų išsaugomi tokie, kokie atsiųsti.
Paprastas sistemos įdiegimas	
Pagrindimas:	Sistemos įdiegimas turi būti nesudėtingas.
Tinkamumo kriterijus:	Tinkamai naudojantis įdiegimo instrukcija įdiegimas pavyksta iš pirmo karto.
Greitas išmokstamumas	
Pagrindimas:	Klientinės sistemos administravimas turi būti greitai išmokstamas, intuityvus.
Tinkamumo kriterijus:	Bet koks vartotojas, sugebantis prisijungti prie sistemos, sėkmingai randa, kaip aktyvuoti atnaujinimus.

4. TINKLO PASLAUGŲ MODELIS IR KŪRIMO PRINCIPAI

Šiame skyriuje pirmiausiai pateikiami paslaugų projektavimo principai. Po jų – atnaujinimo paslaugų specifikacija pagal WSDL standartą. Taip pat pateikiamas jos išplėtimas tinklo paslaugų kokybinėmis charakteristikomis.

4.1. Reikalavimai paslaugų kūrimui

Projektuojant plačiai naudojamas tinklo paslaugas, rekomenduojama laikytis pagrindinių paslaugų projektavimo principų [7]. Šiame darbe projektuojamos paslaugos uždaram naudojimui, tačiau net ir tokiu atveju galima laikytis daugumos šių principų. Toliau aprašomi pagrindiniai 8 paslaugų projektavimo principai.

Standartizuotas paslaugos kontraktas (Standardized Service Contract). Šio principo esmė yra tokia, kad kiekviena paslauga viename paslaugų rinkinyje turi būti aprašyta pagal vienodus standartus. Vienodas formatas reikalingas tam, kad būtų galima vienareikšmiškai nustatyti, kokias operacijas galima vykdyti, kokie jų įeinantys bei išeinantys parametrai, kaip jas iškviesti. Tokiu būdu galima užtikrintai formuoti tikslias paslaugų užklausas.

Laisvas paslaugos susiejimas (Service Loose Coupling). Šio principo esmė – paslaugą reikia projektuoti taip, kad jos aprašas (kontraktas) būtų kuo mažiau priklausomas nuo veiklos logikos, nuo realizavimo, nuo išorinių komponentų, taip pat ir nuo paslaugos naudotojo. Rekomenduojama, kad tik paslaugos naudotojas būtų priklausomas nuo paslaugos aprašo (kadangi būtent aprašu remiantis turi būti projektuojamos paslaugos užklausos iš naudotojo), o ne atvirkščiai. Taip pat rekomenduojama, kad paslaugos veiklos logika būtų projektuojama remiantis paslaugos aprašu, o ne atvirkščiai. Šiuo principu užtikrinama, kad prireikus pakoreguoti paslaugos veiklos logiką ar realizaciją nereikės keisti paslaugos kontrakto, tuo pačiu išvengiant pakeitimų ir naudotojų užklausose.

Paslaugos abstrakcija (Service Abstraction). Šiuo principu teigiama, jog paslaugas reikia projektuoti taip, kad naudotojui būtų prieinama tik ta informacija, kurios jam tikrai reikia. Rekomenduojama paslaugos aprašyme neatskleisti informacijos apie realizaciją bei vidinę veiklos logiką, kadangi priešingu atveju paslaugos naudotojas taptų susietas ne tik su aprašymu, ir taip būtų pažeistas *laisvo paslaugos susiejimo* principas. Būtina atskleisti informaciją tik apie tai, *kam* paslauga skirta, *ką* jos pagalba galima atlikti, bet ne apie tai, *kaip* ji realizuota.

Pakartotinis paslaugos panaudojamumas (Service Reusability). Šio principo esmė tame, kad siūloma paslaugas projektuoti taip, jog jas būtų galima panaudoti ne tik tam vieninteliam tikslui, kurio siekiama kūrimo metu, bet ir kitiems tikslams. Tokiu principu suprojektuota paslauga tampa universalesnė, jos paklausa didėja.

Paslaugos autonomija (Service Autonomy). Šio principo esmė – paslauga turi būti projektuojama taip, kad ji būtų kuo mažiau priklausoma nuo išorinių aplinkybių. Yra skiriami du autonomijos tipai: *vykdymo autonomija (Runtime Autonomy)* bei *projektavimo autonomija (Design-Time Autonomy)*. Jei paslauga yra autonomiška vykdymo metu, t. y. ji pati save kontroliuoja ir nėra priklausoma nuo išorinių veiksnių, tuomet yra stabilesnis jos vykdymo greitis. Tai leidžia lengviau prognozuoti paslaugos vykdymo trukmę. Projektavimo autonomija rodo, kiek paslaugos teikėjas turi galimybių ją keisti. Jei yra laikomasi *laisvo paslaugos susiejimo* principo, tuomet galima pakeisti ir vidinę paslaugos logiką, ir net technologijas, kuriomis paslauga realizuota. Negalima keisti tiktai kontrakto, kadangi su juo yra susieti paslaugos naudotojai.

Paslaugos būsenų neturėjimas (Service Statelessness). Šis principas teigia, jog paslauga turi būti suprojektuota taip, kad būtų kuo mažesnis poreikis saugoti vykdymo būsenas, kadangi būsenų saugojimas užima papildomą atminties kiekį. Kadangi paslaugos užklausą vienu metu gali pateikti daug naudotojų, atminties poreikis gali smarkiai išaugti. Jeigu nėra galimybės išvengti būsenų saugojimo, rekomenduojama tai daryti išorinėje duomenų saugykloje, o ne pagrindinėje atmintinėje. Tačiau tuomet atminties kiekis pagrindinėje atmintinėje padidėja sumažėjusios spartos sąskaita.

Paslaugos atrandamumas (Service Discoverability). Šio principo esmė tokia, kad būtina paslaugos aprašą pateikti bendroje standartizuotoje saugykloje, pavyzdžiui, UDDI registre. Tokiu būdu potencialūs paslaugos naudotojai turėtų informaciją apie tai, ką paslauga gali atlikti, kaip reikia formuoti jos užklausas.

Paslaugos komponuojamumas (Service Composability). Principo esmė – didelės apimties paslauga projektavimo metu išskaidoma į mažesnes atomines paslaugas. Tokiu būdu supaprastinama paslaugų logika bei padidinamos galimybės tenkinti *pakartotino panaudojamumo* principą. Tačiau laikantis šio principo rizikuojama *paslaugos autonomijos* principo pažeidimu, kadangi viena paslauga gali priklausyti nuo daugelio kitų vietinių ar galimai nutolusių paslaugų.

Taigi šiame darbe paslaugos turi būti suprojektuotos taip, kad būtų tenkinama kuo daugiau išvardintų principų. 8.2 poskyryje pateikiamas sukurtų paslaugų atitikimo šiems principams įvertinimas.

4.2. Atnaujinimo paslaugų apibrėžimas pagal WSDL standartą

Šiame darbe modeliuojama dviejų sistemų – centrinės ir klientinės – sąveika, todėl reikia mechanizmo duomenų perdavimui tarp jų. Kadangi pasirinkta duomenų perdavimą realizuoti tinklo paslaugomis, reikalingas tokių paslaugų WSDL (Web Service Definition Language) [3, 4] aprašas. Tiek centrinėje, tiek klientinėje sistemose yra po vieną tinklo paslaugą.

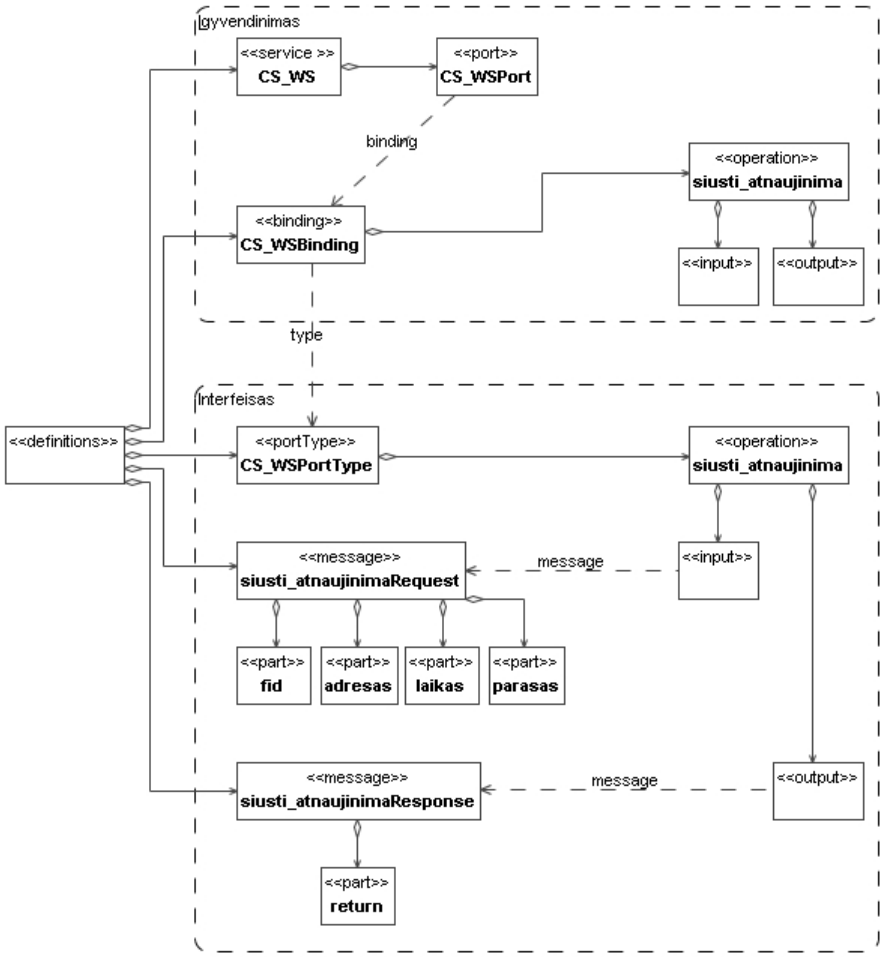
Centrinės sistemos tinklo paslauga skirta nusiųsti programinio kodo fragmentą užklaususiai klientinei sistemai. Siuntimas vykdomas kreipiantis į operaciją „siusti_atnaujinima“. Jos įeinantys parametrai yra tokie:

- fragmento identifikatorius (*fid*);
- užklaususios sistemos adresas (*adresas*);
- užklausos išsiuntimo laikas (*laikas*);
- skaitmeninis parašas (*parasas*) užklausėjo autentifikavimui.

Laiko parametras naudojamas tik tam, kad kiekvieną kartą būtų sugeneruotas skirtingas parašas. Operacijos išeinantis parametras yra *return*, kuriuo grąžinamas serializuotas (paverstas į tekstinę eilutę) masyvas. Jo turinys – vienas iš šių variantų:

- kodo fragmentas, užšifruotas sugeneruotu vienkartiniu slaptu raktu, bei pats slaptas raktas, užšifruotas viešu gavėjo raktu (6.4 posk.);
- įvykus klaidai – klaidos pranešimas.

Šios paslaugos WSDL aprašo struktūrinė schema parodyta 27 pav.

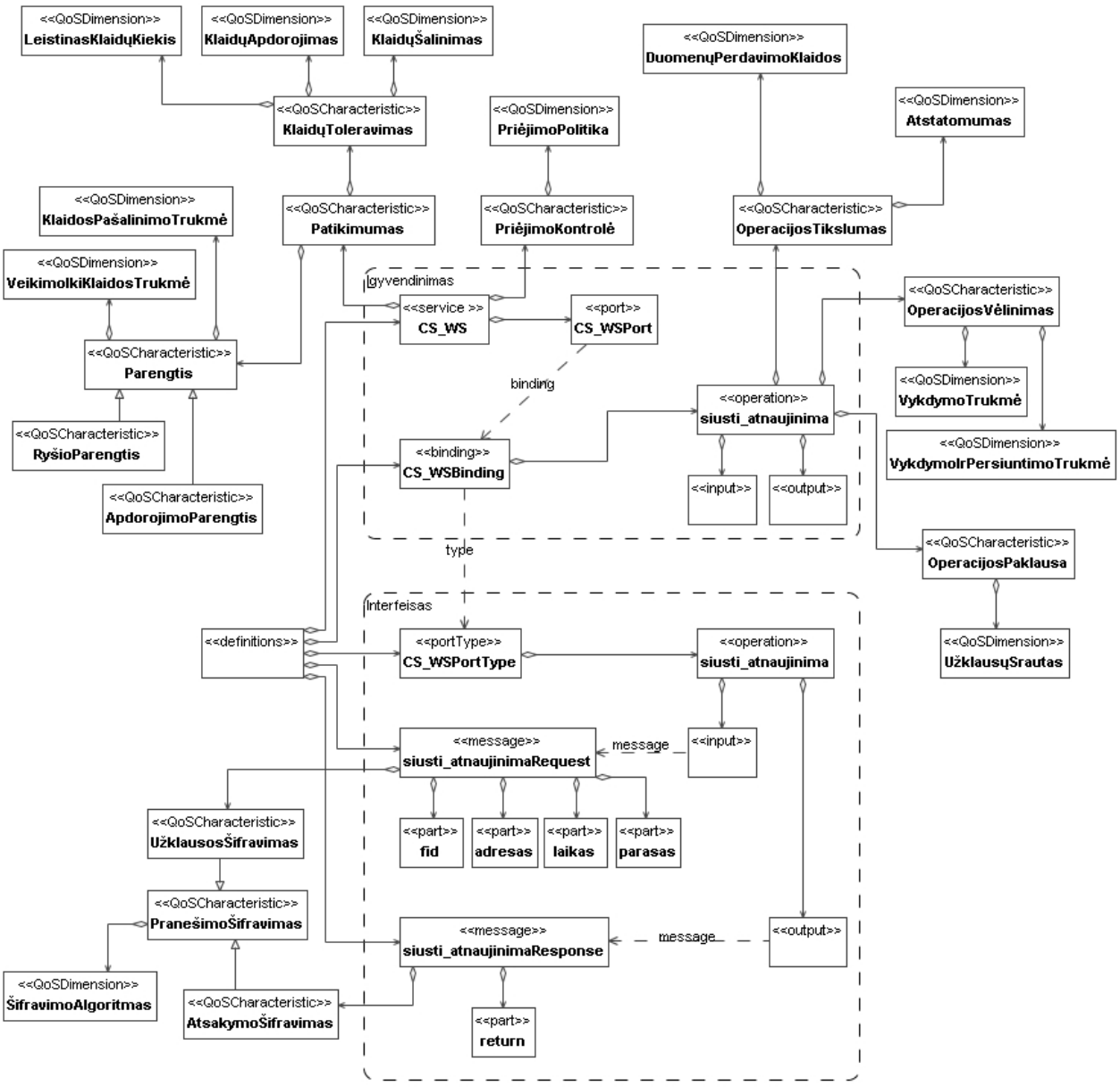


27 pav. Centrinės sistemos tinklo paslaugos aprašo struktūrinė schema

Klientinės sistemos tinklo paslauga skirta centrinės sistemos pranešimo apie kodo fragmento atnaujinimą išsaugojimui. Tai vykdoma kreipiantis į operaciją „issaugoti_pranesima“. Operacijos įeinantys ir išeinantys parametrai yra identiški centrinės sistemos operacijai „siusti_atnaujinima“. Tinklo paslaugos struktūra analogiška centrinės sistemos tinklo paslaugos struktūrai.

4.3. WSDL aprašo papildymas kokybinėmis charakteristikomis

Įprastas WSDL aprašas skirtas specifikuoti, kokias operacijas galima kviesti bei kaip jas pasiekti, tačiau jame visiškai neatspindima, kokia paslaugų kokybė. Renkantis vieną iš keleto analogiškų paslaugų, kokybė gali būti pagrindinis pasirinkimo kriterijus. Taigi natūralu, kad [5] šaltinyje buvo pasiūlyta išplėsti WSDL aprašą kokybinėmis tinklo paslaugų charakteristikomis. Remiantis šiame šaltinyje pateiktu WSDL aprašo išplėtimo metamodeliu bei paslaugų kokybės charakteristikų (angl. QoS characteristics) katalogu [13], centrinės sistemos tinklo paslaugos WSDL aprašo struktūrinė schema papildyta būtent tinklo paslaugoms svarbiomis kokybinėmis charakteristikomis (28 pav.).



28 pav. Centrinės sistemos tinklo paslaugos aprašo su kokybinėmis charakteristikomis struktūrinė schema

Kiekviena kokybinė charakteristika turi stereotipą „QoSCharacteristic“ bei yra susijusi su vienu iš WSDL struktūrinės schemos elementų. Kiekvieną atominę charakteristiką apibrėžia viena ar daugiau dimensijų („QoSDimension“ stereotipas). Sudėtinės charakteristikos dekomponuojamos iki atominių. Toliau aprašoma kiekviena kokybinė tinklo paslaugų charakteristika bei ją apibrėžiančios dimensijos.

- *Patikimumas* (reliability) – sudėtinė charakteristika. Jis skaidoma į *parengties* bei *klaidų toleravimo* charakteristikas.
- *Parengtis* (availability) – tai paslaugos prieinamumas, t. y. būsena, kada paslauga yra prieinama ir gali būti naudojama. Šią charakteristiką specializuoja *ryšio parengtis*, nusakanti ryšio su serveriu stabilumą, bei *apdorojimo parengtis*, nusakanti paslaugos vykdymo serveryje stabilumą. Parengtį apibrėžia tokios dimensijos: *veikimo iki klaidos*

trukmė (time between failure arba TBF) bei *klaidos pašalinimo trukmė* (time to repair arba TTR). Remiantis šiomis dimensijomis galima apskaičiuoti paslaugos prieinamumo santykinę vertę kuri lygi TBF / (TBF + TTR) · 100 %.

- *Klaidų toleravimas* (fault tolerance) parodo, ar atsiradus klaidoms paslauga gali toliau būti prieinama ir vykdoma. Jį apibrėžia tokios dimensijos: *leistinas klaidų kiekis* (parodo, kiek klaidų gali įvykti, kol paslauga nebegalės būti vykdoma), *klaidų apdorojimas* (kaip sistemoje elgiamasi įvykus klaidoms) bei *klaidų šalinimas* (kokių priemonių imamasi klaidoms panaikinti).
- *Priėjimo kontrolė* (access control) nusako, kokiu būdu atrenkama, kas gali kviesti paslaugos operacijas. Atrinkimą apibrėžia šios charakteristikos dimensija *priėjimo politika*.
- *Operacijos tikslumas* (operation accuracy) parodo, kiek gali skirtis gautas atsakymas nuo to, kurį tikimasi gauti. Tikslumą apibrėžia tokios dimensijos: *duomenų perdavimo klaidos* (kokia tikimybė, kad gali įvykti klaida duomenų perdavimo metu), *atstatomumas* (ar yra galimybė atstatyti sistemą į paskutinę stabilią būseną įvykus klaidai).
- *Operacijos vėlinimas* (operacion latency) parodo, kokia yra operacijos vykdymo trukmė. Šią charakteristiką apibrėžia tokios dimensijos: *vykdymo trukmė* (kiek laiko operacija vykdoma serveryje) bei *vykdymo ir persiuntimo trukmė* (turn-around time – koks laiko skirtumas tarp užklausos išsiuntimo ir atsakymo gavimo momentų).
- *Operacijos paklausa* (operation demand) nusako, ar dažnai operacija yra kviečiama. Ši charakteristika apibrėžiama dimensija *užklausų srautas* (užklausų kiekis per tam tikrą laiko tarpą).
- *Pranešimo šifravimas* (message encryption) skirtas siunčiamų pranešimų saugumui nusakyti. Jį specializuoja *užklausos šifravimas* bei *atsakymo šifravimas*. Šifravimą apibrėžia *šifravimo algoritmo* dimensija.

Kokybinių charakteristikų pateikimas gali tapti papildomu privalumu pasirašant sutartis su užsakovais. Taigi tinklo paslaugų aprašo išplėtimas kokybinėmis charakteristikomis svarbus ne tik įprastoje tinklo paslaugų architektūroje, bet ir šiame darbe sukurtam informacijos sistemų atnaujinimo modeliui.

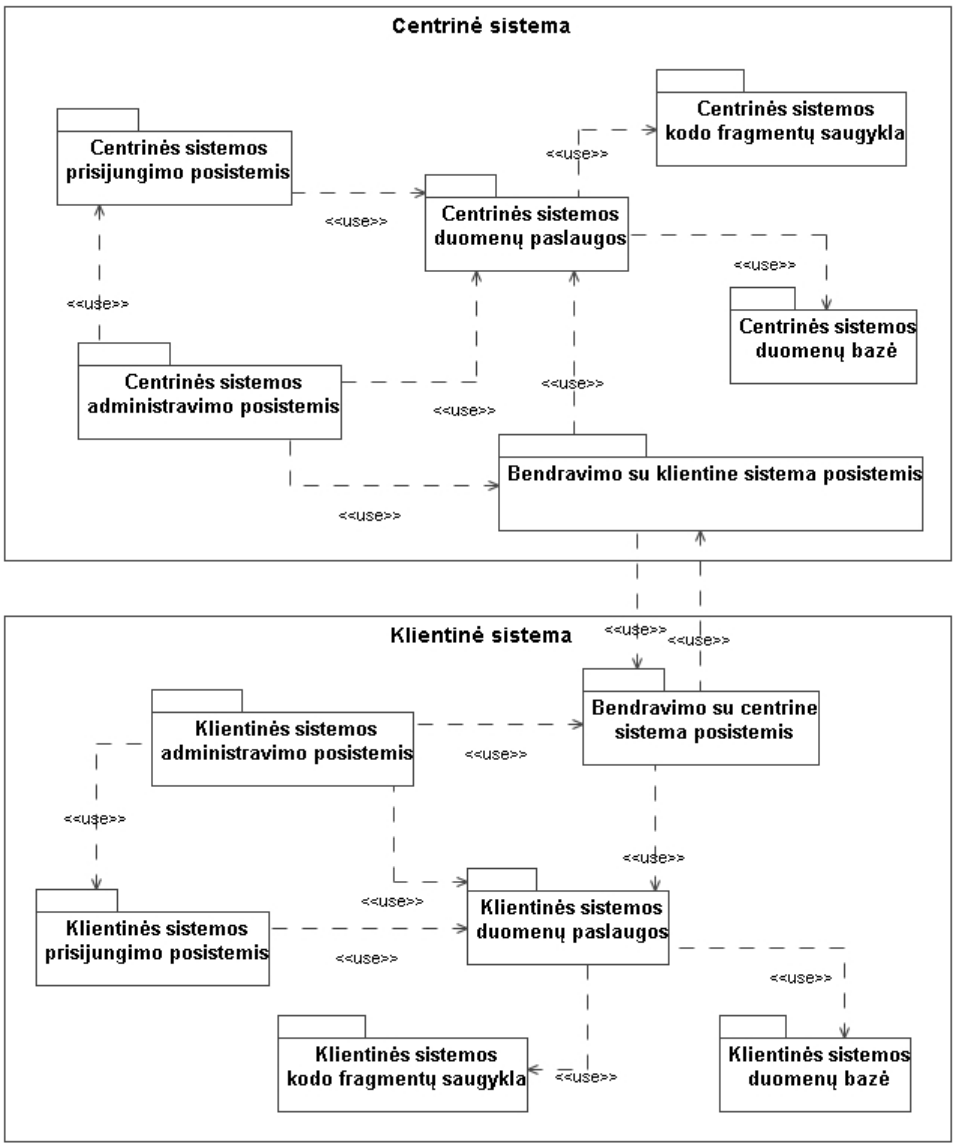
5. CENTRALIZUOTO ATNAUJINIMO SISTEMOS MODELIS

Šiame skyriuje apibrėžiamas kuriamos sistemos projekto tikslas, pateikiama loginė centrinės ir klientinės sistemų architektūra. Taip pat pateikiamas duomenų bazės modelis, detalus pagrindinių posistemių projektas, elgsenos modelis. Galiausiai aprašomi metodai, technologijos, atspindinčios darbo idėją ir naujumą.

5.1. Projekto tikslas

Projekto tikslas – suprojektuoti centrinę bei klientinę sistemas naudojant CASE priemonę, kad remiantis sukurtais modeliais būtų galima realizuoti prototipą.

5.2. Loginė centrinės ir klientinės sistemų architektūra



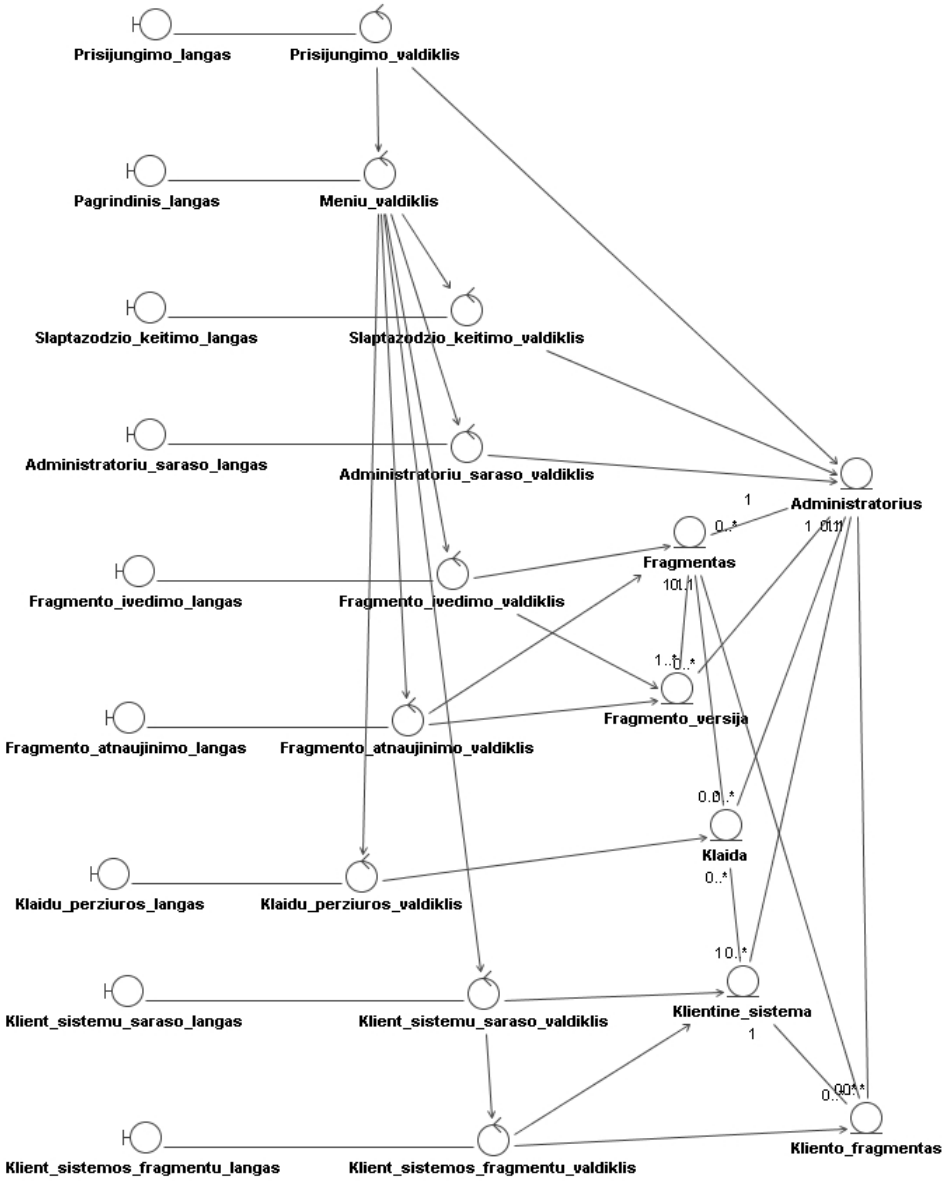
29 pav. Loginė centrinės ir klientinės sistemų architektūra

29 pav. parodyta loginė centrinės ir klientinės sistemų architektūra. Čia parodoma, į kokius loginius sąveikaujančius posistemius galima suskaidyti sistemas. Taip pat parodoma, per kuriuos posistemius vyksta centrinės ir klientinės sistemų bendravimas.

5.4.1 ir 5.4.2 poskyriuose pateiktose panaudojimo atvejų realizacijos diagramose matoma, kuriam posistemiui priskiriamas kiekvienas panaudojimo atvejis. 5.4.3 ir 5.4.4 poskyriuose pateiktas detalus loginių posistemių projektas.

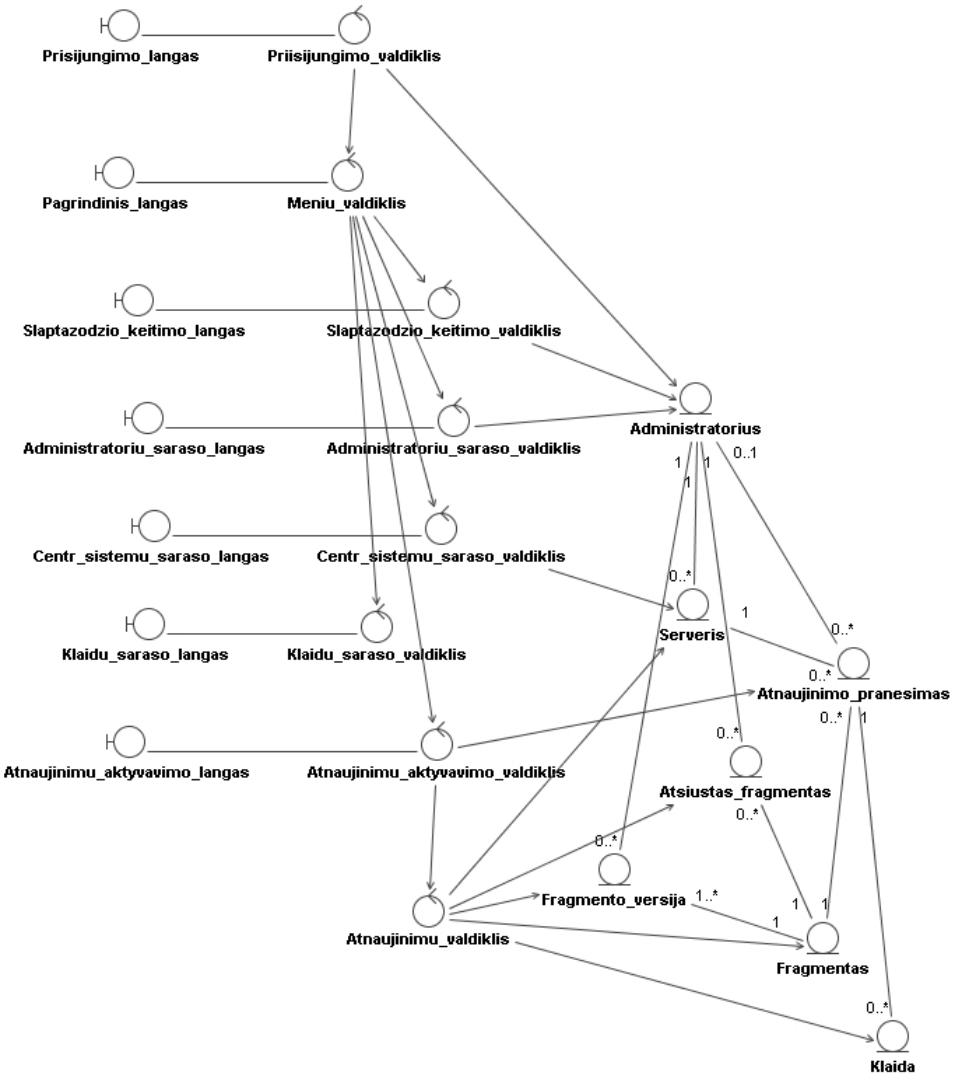
5.3. Centralizuoto atnaujinimo sistemos reikalavimų analizė

Šiame poskyryje pateikiami centrinės bei klientinės sistemų reikalavimų analizės modeliai, sukurti išanalizavus sistemai keliamus reikalavimus. 30 pav. pateiktoje diagramoje parodomos analizės metu numatytos centrinės sistemos duomenų, valdymo ir ribinės klasės bei ryšiai tarp jų.



30 pav. Centrinės sistemos analizės modelis

Prisijungęs prie sistemos administratorius nukreipiamas į pagrindinį langą, kuriame per meniu gali patekti į bet kurį kitą langą. Analogiška veiksmų seka yra ir 31 pav. pateiktame klientinės sistemos duomenų, valdymo bei ribinių klasių modelyje.



31 pav. Klientinės sistemos analizės modelis

5.4. Centrinės ir klientinės dalies klasių modeliai

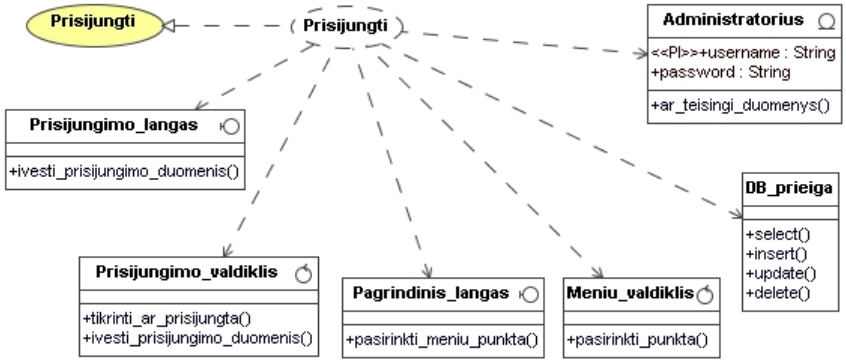
Šiame poskyryje pirmiausiai pateikiamos centrinės bei klientinės sistemų panaudojimo atvejų realizacijos diagramos. Po jų – detalūs centrinės ir klientinės sistemų projektai.

5.4.1. Centrinės sistemos panaudojimo atvejų realizacijos klasės

Šiame poskyryje pateikiamos pagrindinių centrinės sistemos posistemių panaudojimo atvejų realizacijos diagramos. Jos parodo, kokios klasės yra reikalingos realizuojant kiekvieną panaudojimo atvejį.

5.4.1.1. Centrinės sistemos prisijungimo posistemio panaudojimo atvejų realizacijos

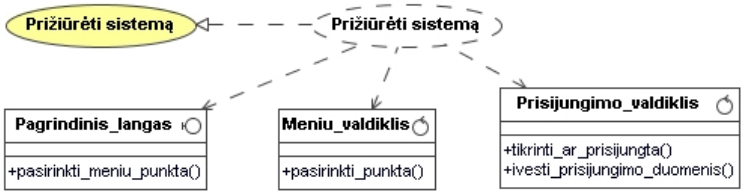
32 pav. pateikta panaudojimo atvejo „Prisijungti“ realizacijos diagrama.



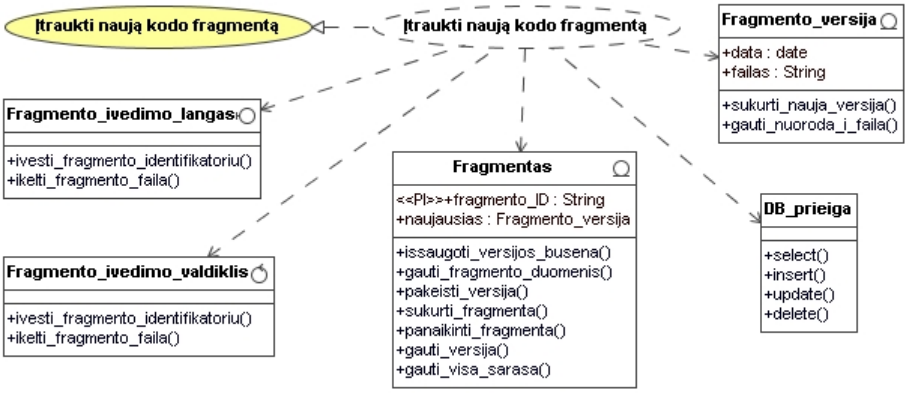
32 pav. PA „Prisijungti“ realizacijos diagrama

5.4.1.2. Centrinės sistemos administravimo posistemio panaudojimo atvejų realizacijos

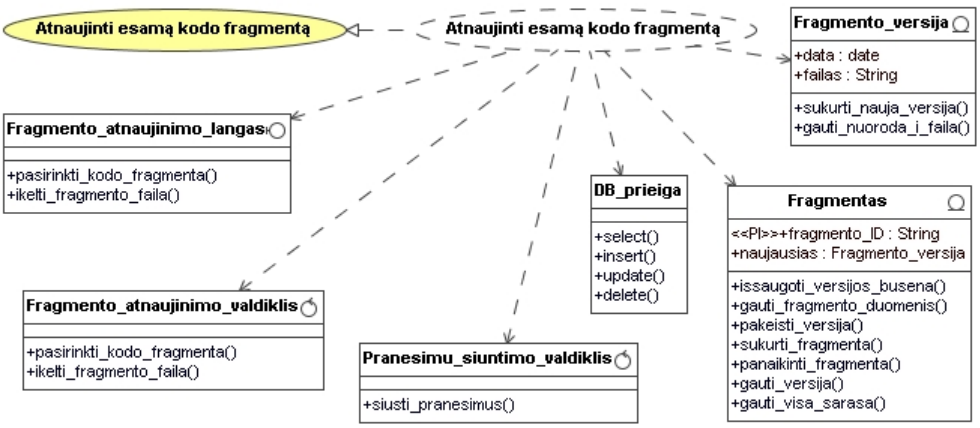
33 pav. pateikta panaudojimo atvejo „Prižiūrėti sistemą“, 34 pav. – „Įtraukti naują kodo fragmentą“, 35 pav. – „Atnaujinti esamą kodo fragmentą“, 36 pav. – „Peržiūrėti klaidų pranešimus“, 37 pav. – „Tvarkyti klientinių sistemų sąrašą“, 38 pav. – „Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą“, 39 pav. – „Tvarkyti administratorių sąrašą“, 40 pav. – „Keisti prisijungimo slaptažodį“ realizacijos diagramos.



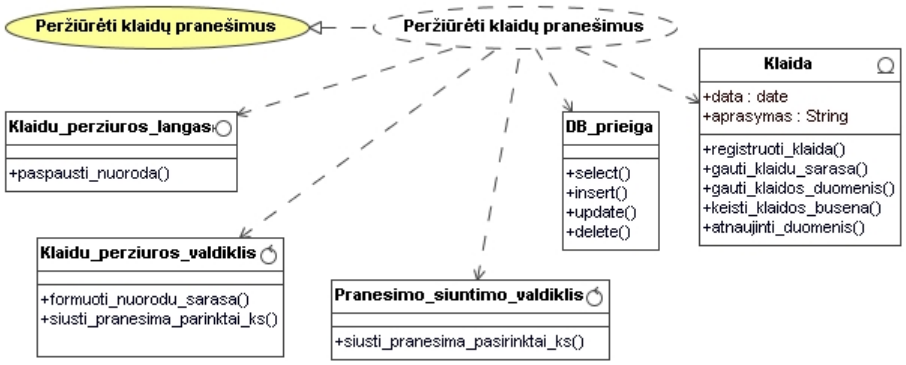
33 pav. PA „Prižiūrėti sistemą“ realizacijos diagrama



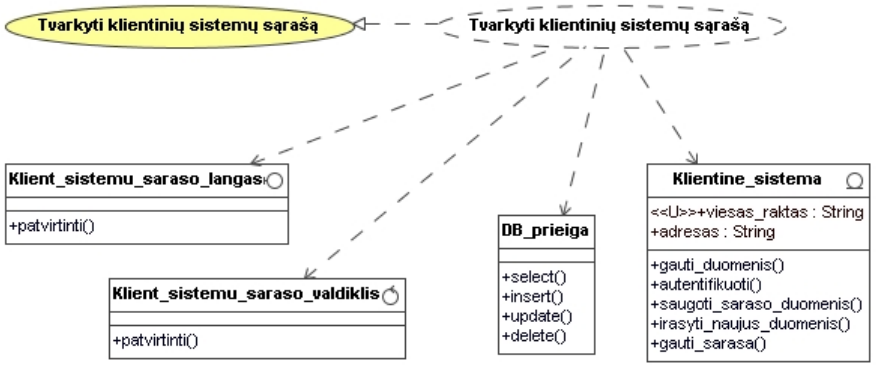
34 pav. PA „Įtraukti naują kodo fragmentą“ realizacijos diagrama



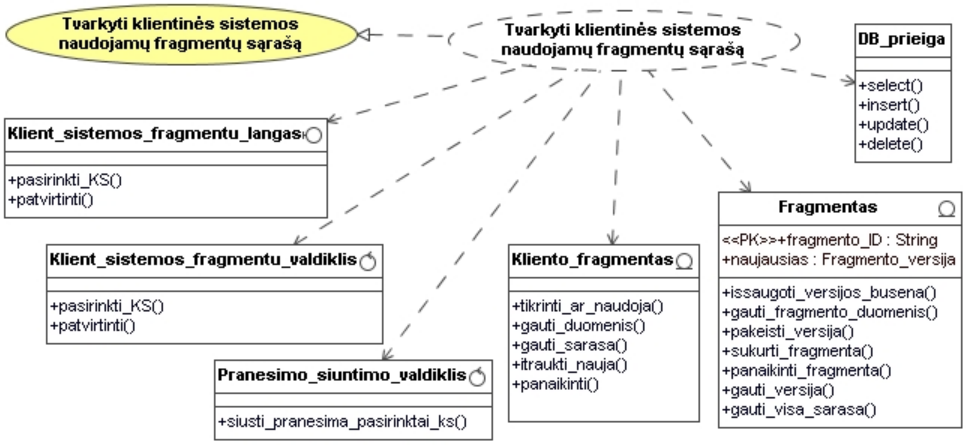
35 pav. PA „Atnaujinti esamą kodo fragmentą“ realizacijos diagrama



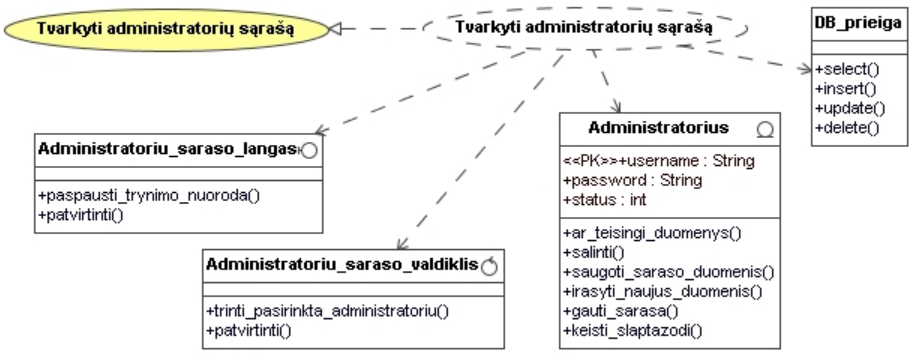
36 pav. PA „Peržiūrėti klaidų pranešimus“ realizacijos diagrama



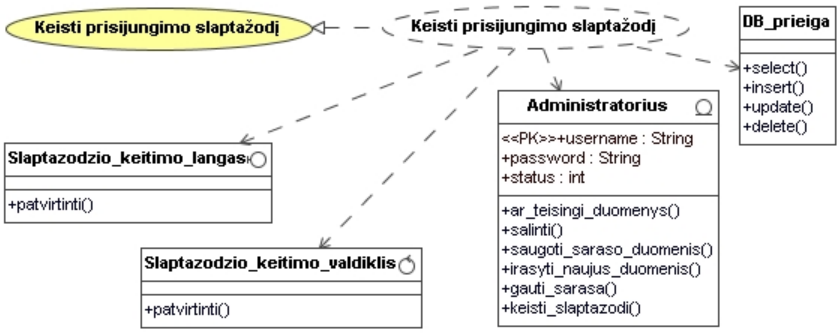
37 pav. PA „Tvarkyti klientinių sistemų sąrašą“ realizacijos diagrama



38 pav. PA „Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą“ realizacijos diagrama



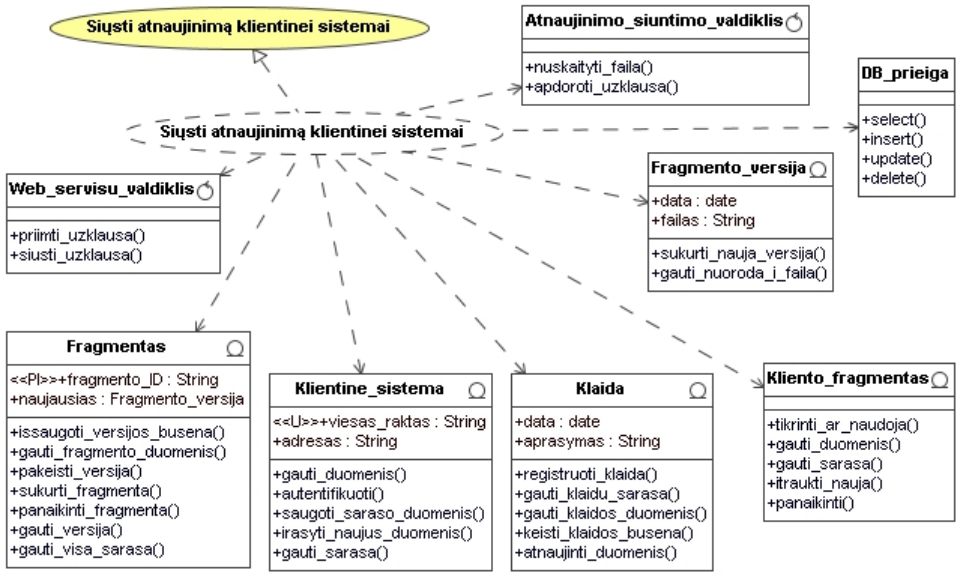
39 pav. PA „Tvarkyti administratorių sąrašą“ realizacijos diagrama



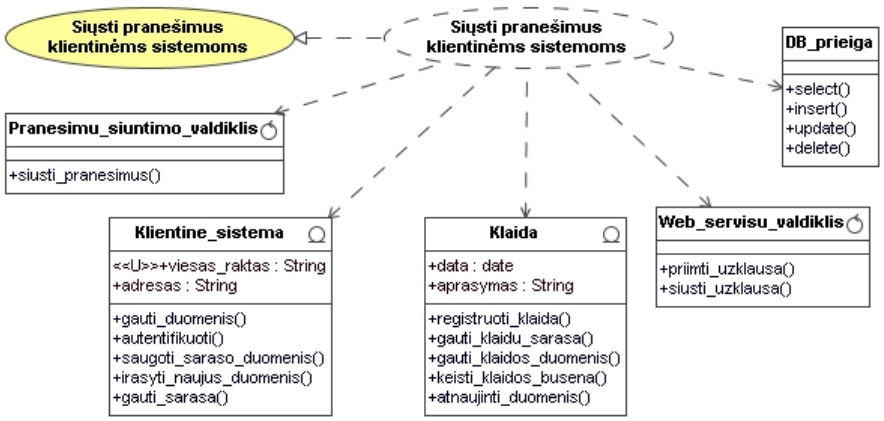
40 pav. PA „Keisti prisijungimo slaptažodį“ realizacijos diagrama

5.4.1.3. Bendravimo su klientine sistema posistemio panaudojimo atvejų realizacijos

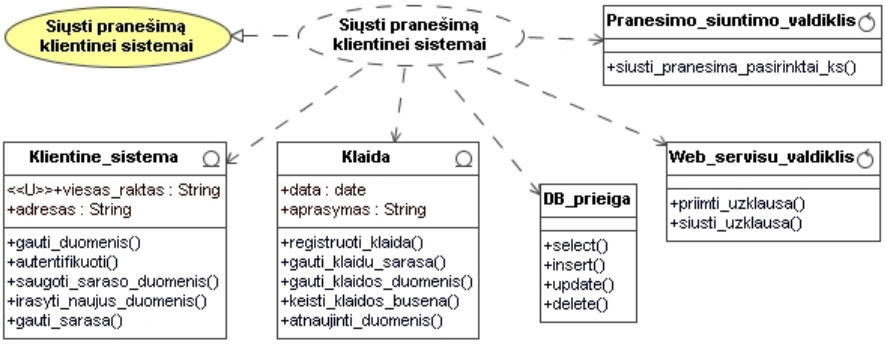
41 pav. pateikta panaudojimo atvejo „Siųsti atnaujinimą klientinei sistemai“, 42 pav. – „Siųsti pranešimus klientinėms sistemoms“, 43 pav. – „Siųsti pranešimą klientinei sistemai“ realizacijos diagramos.



41 pav. PA „Siųsti atnaujinimą klientinei sistemai“ realizacijos diagrama



42 pav. PA „Siųsti pranešimus klientinėms sistemoms“ realizacijos diagrama



43 pav. PA „Siųsti pranešimą klientinei sistemai“ realizacijos diagrama

5.4.2. Klientinės sistemos panaudojimo atvejų realizacijos klasės

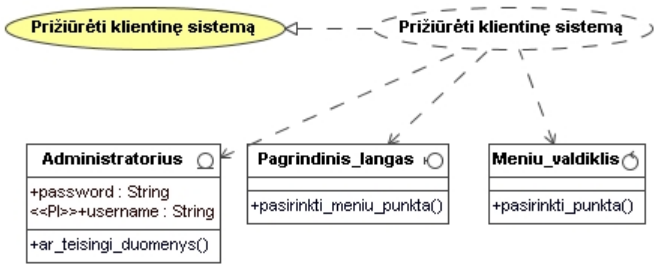
Analogiškai 5.4.1 poskyriui, šiame poskyryje pateikiamos pagrindinių klientinės sistemos posistemių panaudojimo atvejų realizacijos diagramos.

5.4.2.1. Klientinės sistemos prisijungimo posistemio panaudojimo atvejų realizacijos

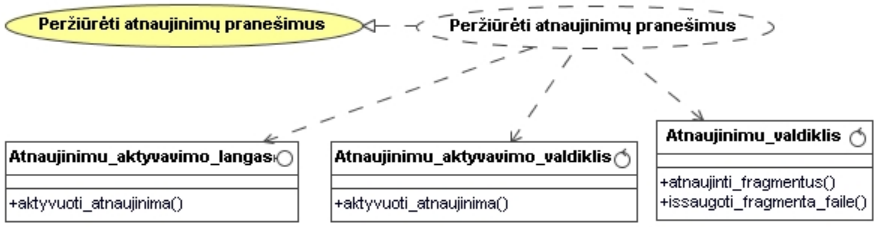
Klientinės sistemos panaudojimo atvejo „Prisijungti“ realizacijos diagrama yra identiška analogiško centrinės sistemos panaudojimo atvejo realizacijos diagramai (32 pav.).

5.4.2.2. Klientinės sistemos administravimo posistemio panaudojimo atvejų realizacijos

44 pav. pateikta panaudojimo atvejo „Prižiūrėti klientinę sistemą“, 45 pav. – „Peržiūrėti atnaujinimų pranešimus“, 48 pav. – „Peržiūrėti klaidų sąrašą“, 47 pav. – „Tvarkyti centrinių sistemų sąrašą“ realizacijos diagramos. Panaudojimo atvejų „Tvarkyti administratorių sąrašą“ bei „Keisti prisijungimo slaptažodį“ realizacijos diagramos yra identiškos atitinkamų centrinės sistemos panaudojimo atvejų realizacijos diagramoms (39 pav. ir 40 pav.).

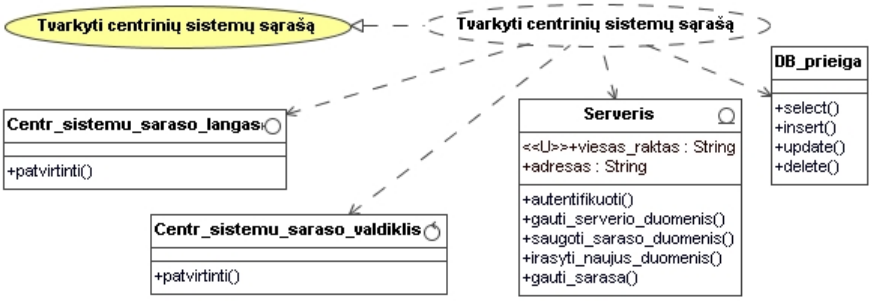


44 pav. PA „Prižiūrėti klientinę sistemą“ realizacijos diagrama



45 pav. PA „Peržiūrėti atnaujinimų pranešimus“ realizacijos diagrama

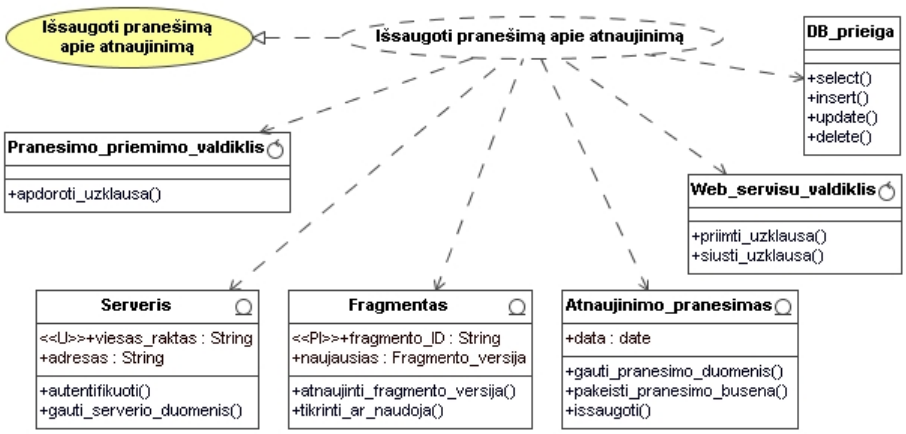
46 pav. PA „Peržiūrėti klaidų sąrašą“ realizacijos diagrama



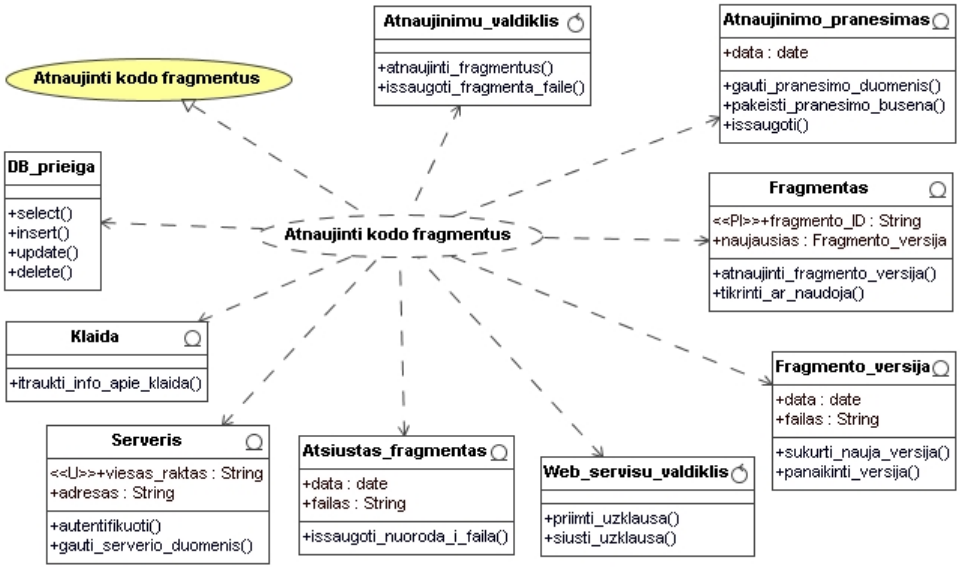
47 pav. PA „Tvarkyti centrinių sistemų sąrašą“ realizacijos diagrama

5.4.2.3. Bendravimo su centrine sistema posistemio panaudojimo atvejų realizacijos

48 pav. pateikta panaudojimo atvejo „Išsaugoti pranešimą apie atnaujinimą“, 49 pav. – „Atnaujinti kodo fragmentus“ realizacijos diagramos.



48 pav. PA „Išsaugoti pranešimą apie atnaujinimą“ realizacijos diagrama



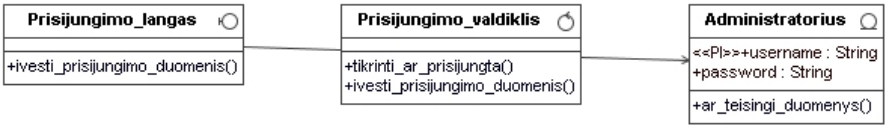
49 pav. PA „Atnaujinti kodo fragmentus“ realizacijos diagrama

5.4.3. Detalus centrinės sistemos projektas

Šiame poskyryje pateikiamos pagrindinių centrinės sistemos posistemių veiklos logikos bei vartotojo sąsajos klasių diagramos

5.4.3.1. Centrinės sistemos prisijungimo posistemio detalus projektas

50 pav. pateikta prisijungimo posistemio vartotojo sąsajos bei veiklos logikos klasių diagrama. Prisijungimo langas atitinka vartotojo sąsają administratoriaus prisijungimui prie sistemos (operacija *ivesti_prisijungimo_duomenis()*). Prisijungimo valdiklis skirtas prisijungimui (*ivesti_prisijungimo_duomenis()*) bei jo tikrinimui (*tikrinti_ar_prisijungta()*). Šios klasės naudojama klasė *Administratorius* skirta saugoti prisijungusio administratoriaus duomenis, o prisijungiant tikrinti, ar tie duomenys teisingi (*ar_teisingi_duomenys()*).



50 pav. Centrinės sistemos prisijungimo posistemio vartotojo sąsajos bei veiklos logikos klasių diagrama

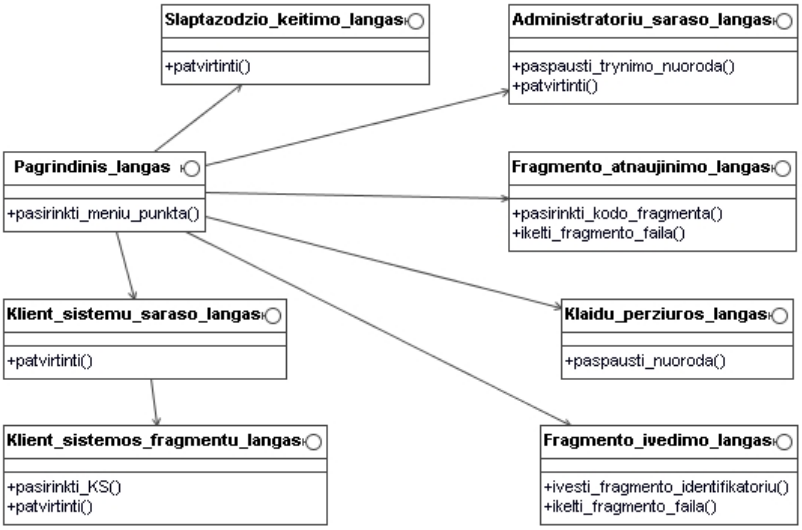
5.4.3.2. Centrinės sistemos administravimo posistemio detalus projektas

51 pav. pateiktas administravimo posistemio vartotojo sąsajos modelis. Iš pagrindinio lango, pasirinkus atitinkamą meniu punktą (*pasirinkti_meniu_punkta()*), patenkama į vieną iš kitų langų.

Fragmento_ivedimo_langas skirtas naujo kodo fragmento įvedimui. Dvi šios klasės operacijos (*ivesti_fragmento_identifikatoriu()* ir *ikelti_fragmento_faila()*) rodo, kad šiame lange administratorius turi atlikti du veiksmus.

Fragmento_atnaujinimo_langas skirtas esamo kodo fragmento naujos versijos įkėlimui. Ši klasė taip pat turi dvi operacijas: vieno iš fragmentų pasirinkimui (*pasirinkti_kodo_fragmenta()*) ir naujo failo įkėlimui (*ikelti_fragmento_faila()*).

Klaidu_perziuros_langas skirtas užregistruotų klaidų siunčiant pranešimus klientinėms sistemoms sąrašo peržiūrai. Operacija *paspausti_nuoroda()* skirta pakartotinai siųsti pranešimą vienai iš klientinių sistemų.

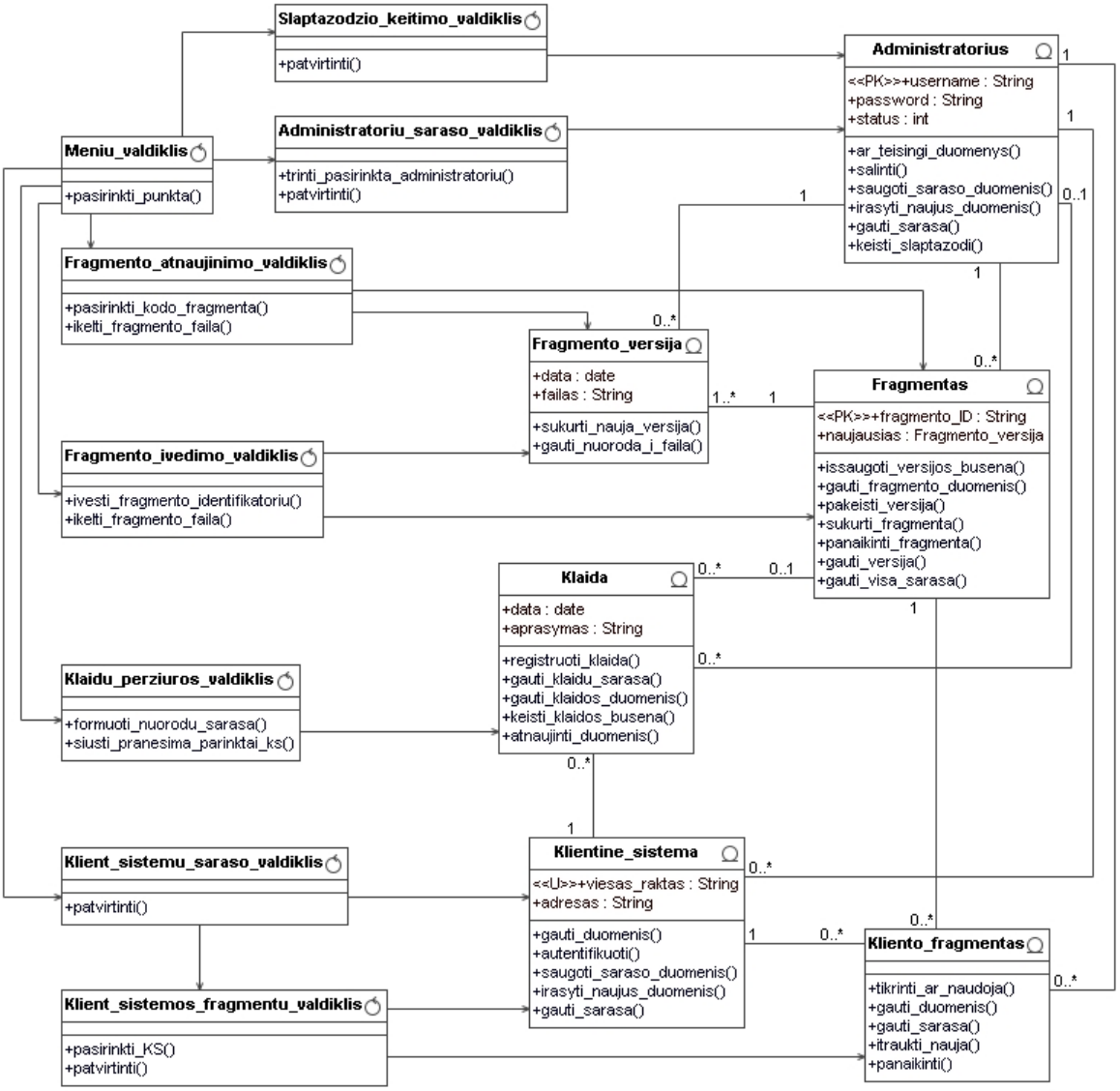


51 pav. Centrinės sistemos administravimo posistemio vartotojo sąsajos modelis

Klient_sistemu_saraso_langas skirtas esamų klientinių sistemų duomenų redagavimui bei naujų įvedimui. Operacija *patvirtinti()* skirta atliktų pakeitimų ir/arba naujų duomenų išsaugojimui. Iš šio lango galima patekti į pasirinktos klientinės sistemos naudojamų fragmentų sąrašo tvarkymo langą *Klient_sistemos_fragmentu_langas*. Jame bet kada galima pasirinkti kitą klientinę sistemą (*pasirinkti_KS()*). Sužymėjus reikiamus kodo fragmentus naudojama operacija *patvirtinti()*, kad būtų išsaugoti pakeitimai.

Administratoriu_saraso_langas skirtas esamų administratorių duomenų redagavimui bei naujų įvedimui. Operacija *patvirtinti()* skirta atliktų pakeitimų ir/arba naujų duomenų išsaugojimui. Operacija *paspausti_trynimo_nuoroda()* naudojama ištrinant vieno administratoriaus duomenis.

Slaptazodzio_keitimo_langas skirtas prisijungusiam administratoriui pakeisti savo slaptažodį. Operacija *patvirtinti()* skirta naujo slaptažodžio išsaugojimui (arba atmetimui netinkamo įvedimo atveju).



52 pav. Centrinės sistemos administravimo posistemio veiklos logikos klasių diagrama

52 pav. – šio posistemio veiklos logikos klasių diagrama. Kiekviena vartotojo sąsajos modelio klasė kreipiasi į ją atitinkančią veiklos logikos klasę, taip perduodama valdymą šio lygmens klasėms: pagrindinis langas – į meniu valdiklį, fragmento įvedimo langas – į fragmento įvedimo valdiklį ir t. t. Valdikliai, priklausomai nuo to, kokią operaciją atlieka, gali sukurti ar modifikuoti duomenų klasių duomenis (ryšiai tarp valdiklių ir esybių rodo, kuriomis esybėmis kuris valdiklis gali manipuliuoti).

Fragmentas skirtas kodo fragmento duomenų saugojimui bei apdorojimui. Jis identifikuojamas tekstiniu identifikatoriumi (pavyzdžiui, „class_mail_01“). Operacija *sukurti_fragmenta()* naujas fragmentas yra sukuriamas, o *panaikinti_fragmenta()* – panaikinamas (jei nebuvo įkeltas pradinio kodo failas). Operacija *issaugoti_versijos_busena()* skirta atributo *naujausias*, nurodančio į naujausią fragmento versiją, išsaugojimui sukūrus fragmentą. Tam pačiam atributui pakeisti, kai fragmentas atnaujinamas, skirta operacija *pakeisti_versija()*. Duomenų išgavimui skirtos operacijos *gauti_fragmento_duomenis()*, *gauti_versija()* ir *gauti_visa_sarasa()*.

Fragmento_versija skirta fragmento versijoms saugoti. Operacija *sukurti_nauja_versija()* įveda naują fragmento versiją, o *gauti_nuoroda_i_faila()* – grąžina failo, kuriame saugomas pasirinktos versijos kodas, pavadinimą.

Klientine_sistema – tai duomenims apie klientines sistemas saugoti ir apdoroti skirta klasė. Naujų klientinių sistemų duomenys įvedami naudojant operaciją *irasyti_naujus_duomenis()*, o esami duomenys redaguojami operacija *saugoti_saraso_duomenis()*. Vienos klientinės sistemos duomenų išgavimui skirta operacija *gauti_duomenis()*, o visas sąrašas gaunamas kreipiantis į operaciją *gauti_sarasa()*. Klientinės sistemos, gavus jos užklausą, atpažinimui skirta operacija *autentifikuoti()*.

Kliento_fragmentas klasė skirta klientinių sistemų naudojamų fragmentų sąrašui saugoti. Įtraukiant į sąrašą naują fragmentą naudojama operacija *itraukti_nauja()*, o pašalinant iš sąrašo – *panaikinti()*. Operacija *tikrinti_ar_naudoja()* skirta patikrinti, ar konkreti klientinė sistema naudoja konkretų fragmentą. Duomenų išgavimui yra skirtos operacijos *gauti_duomenis()* ir *gauti_sarasa()*.

Klaida – tai duomenims apie nepavykusius pranešimų klientinėms sistemoms siuntimus saugoti skirta klasė. Nauja klaida įrašoma naudojant operaciją *registruoti_klaida()*. Kai pakartotinai nepavyksta pranešimo apie to paties fragmento atnaujinimą tai pačiai klientinei sistemai siuntimas, naudojant operaciją *atnaujinti_duomenis()* redaguojamas jau egzistuojantis įrašas. Kai pakartotinis siuntimas pavyksta, naudojama operacija *keisti_klaidos_busena()*. Duomenų išgavimui skirtos operacijos *gauti_klaidos_duomenis()* bei *gauti_klaidu_sarasa()*.

Administratorius – tai duomenims apie sistemos administratorius saugoti ir apdoroti skirta klasė. Naujų administratorių duomenys įvedami naudojant operaciją *irasyti_naujus_duomenis()*, o esami duomenys redaguojami operacija *saugoti_saraso_duomenis()*. Vieno administratoriaus duomenų pašalinimui skirta operacija *salinti()*. Visas sąrašas gaunamas kreipiantis į operaciją *gauti_sarasa()*. Slaptažodžio keitimui kreipiamasi į operaciją *keisti_slaptazodi()*. Administratoriaus atpažinimui prisijungiant skirta operacija *ar_teisingi_duomenys()*.

5.4.3.3. Bendravimo su klientine sistema posistemio detalus projektas

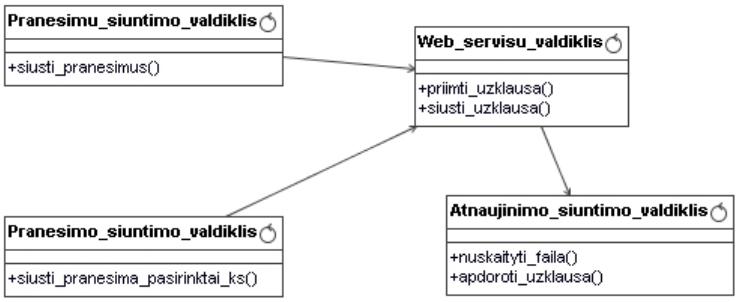
53 pav. pateikta bendravimo su klientine sistema posistemio veiklos logikos klasių diagrama. Šiame posistemyje veikla yra automatinė, t. y. administratoriaus veiksmai nereikalingi. Todėl vartotojo sąsajos elementų čia nėra.

Web_servisu_valdiklis skirtas siųsti užklausas klientinėms sistemoms (*siusti_uzklausa()*) bei priimti užklausas iš jų (*priimti_uzklausa()*). Gavus užklausą iš klientinės sistemos, šis valdiklis grąžina jai tai, ką jam grąžina atnaujinimo siuntimo valdiklio operacija *apdoroti_uzklausa()*.

Pranesimu_siuntimo_valdiklis skirtas siųsti pranešimą visoms atnaujintą fragmentą naudojančioms klientinėms sistemoms (*siusti_pranesimus()*). Kiek yra tokių sistemų, tiek kartų šis valdiklis kreipiasi į Web servisų (tinklo paslaugų) valdiklį, t. y. kviečia jo operaciją *siusti_uzklausa()*.

Pranesimo_siuntimo_valdiklis naudojamas siųsti pranešimą vienai klientinei sistemai (*siusti_pranesima_pasirinktai_ks()*). Tai gali būti pasirinkta peržiūrint klaidų sąrašą. Taip pat šis valdiklis naudojamas tvarkant pasirinktos klientinės sistemos naudojamų fragmentų sąrašą. Šiuo atveju apie kiekvieną naujai priskirtą fragmentą pranešimai siunčiami automatiškai. Šis valdiklis taip pat kviečia Web servisų valdiklio operaciją *siusti_uzklausa()*.

Atnaujinimo_siuntimo_valdiklis naudojamas Web servisų valdiklio, kai šis gauna užklausą iš klientinės sistemos. Operacija *apdoroti_uzklausa()* grąžina operacija *nuskaityti_faila()* gautą kodą, jei klientinė sistema atpažįstama bei nustatoma, kad ji naudoja fragmentą, kurio atnaujinimo reikalauja. Priešingu atveju grąžinamas pranešimas apie klaidą.



53 pav. Bendravimo su klientine sistema posistemio veiklos logikos klasių diagrama

5.4.4. Detalus klientinės sistemos projektas

Analogiškai 5.4.3 poskyriui, šiame poskyryje pateikiamos pagrindinių klientinės sistemos posistemių veiklos logikos klasių diagramos.

5.4.4.1. Klientinės sistemos prisijungimo posistemio detalus projektas

Klientinės sistemos prisijungimo posistemio vartotojo sąsajos ir veiklos logikos klasių diagrama yra identiška analogiškai centrinės sistemos diagramai (50 pav.).

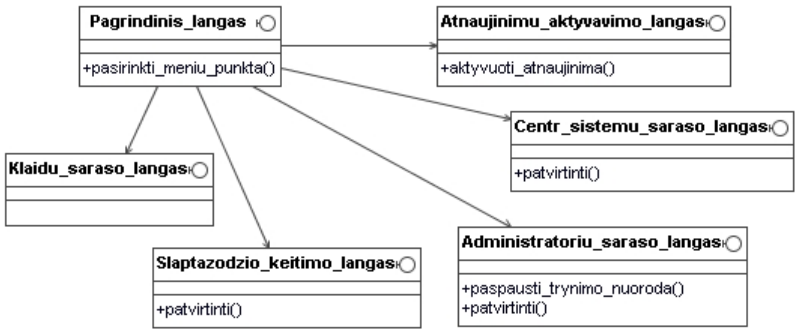
5.4.4.2. Klientinės sistemos administravimo posistemio detalus projektas

54 pav. pateiktas administravimo posistemio vartotojo sąsajos modelis. Kaip ir centrinėje sistemoje, pasirinkus meniu punktą (*pasirinkti_meniu_punkta()*) iš pagrindinio lango patenkama į bet kurį kitą langą.

Atnaujinimu_aktyvavimo_langas skirtas peržiūrėti atsiųstus atnaujinimo pranešimus. Jei yra naujų pranešimų, pateikiamas mygtukas atnaujinimui. Jį paspaudus kviečiama operacija *aktyvuoti_atnaujinima()*.

Centr_sistemu_saraso_langas skirtas esamų centrinių sistemų duomenų redagavimui bei naujų įvedimui. Operacija *patvirtinti()* skirta atliktų pakeitimų ir/arba naujų duomenų išsaugojimui.

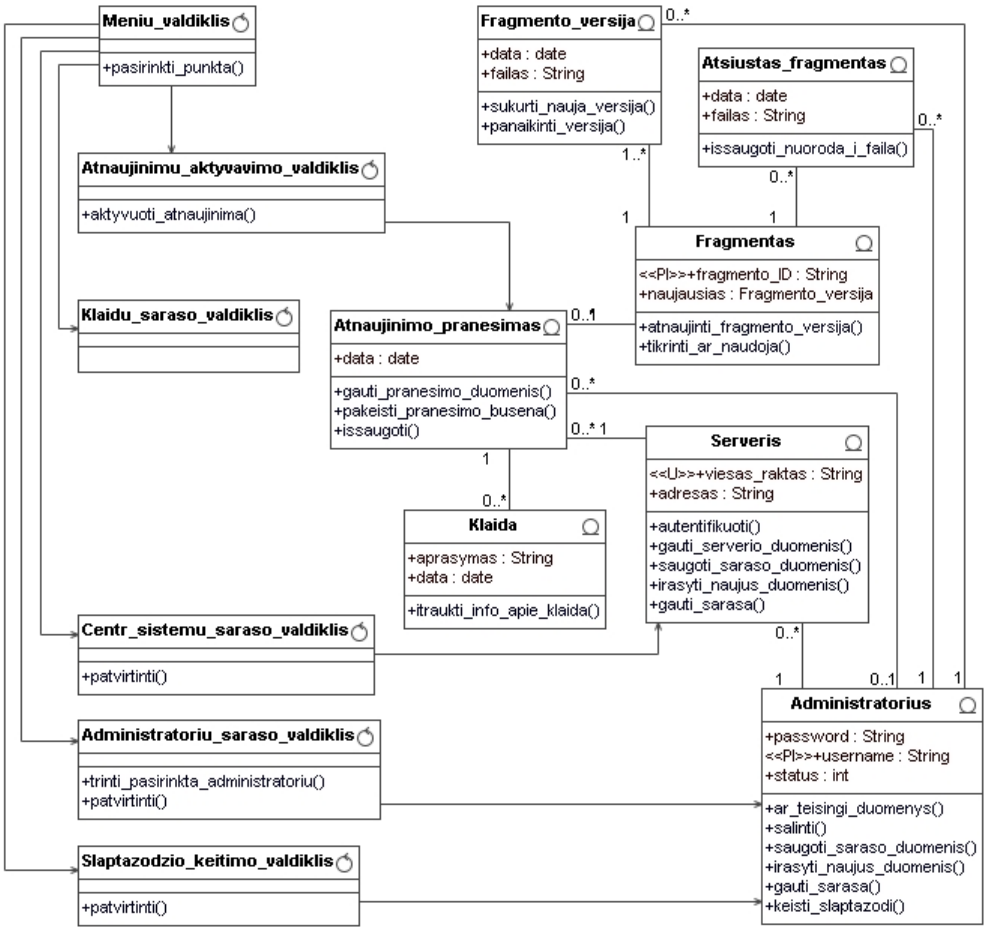
Klaidu_saraso_langas skirtas užregistruotų klaidų peržiūrai. Iš administratoriaus pusės jokie veiksmai nėra reikalingi. *Administratoriu_saraso_langas* ir *Slaptazodzio_keitimo_langas* yra identiški atitinkamiems centrinės sistemos langams.



54 pav. Klientinės sistemos administravimo posistemio vartotojo sąsajos modelis

55 pav. – šio posistemio veiklos logikos klasių diagrama. Kiekviena vartotojo sąsajos modelio klasė kreipiasi į ją atitinkančią veiklos logikos klasę, taip perduodama valdymą šio lygmens klasėms: pagrindinis langas – į meniu valdiklį, atnaujinimų aktyvavimo langas – į atnaujinimų aktyvavimo valdiklį. Šis valdiklis gali pakeisti tik atnaujinimo pranešimų duomenis.

Fragmentas skirtas kodo fragmento duomenų saugojimui bei apdorojimui. Jis atitinka analogišką centrinės sistemos klasę, tik turi mažiau operacijų. Atributui *naujausias*, rodančiam, kuri fragmento versija yra naujausia, pakeisti, kai fragmentas atnaujinamas, skirta operacija *atnaujinti_fragmento_versija()*. Operacija *tikrinti_ar_naudoja()* skirta patikrinti, ar klientinė sistema naudoja konkretų kodo fragmentą.



55 pav. Klientinės sistemos administravimo posistemio veiklos logikos klasių diagrama

Fragmento_versija skirta fragmento versijoms saugoti. Operacija *sukurti_nauja_versija()* įveda naują fragmento versiją, o *panaikinti_versija()* – pašalina sukurtą įrašą, jei klasės *Fragmentas* objektui nepavyksta pakeisti fragmento versijos.

Atsiustas_fragmentas skirtas laikinai saugoti atsiųstą kodo fragmentą, jei klasės *Fragmentas* objektui nepavyksta pakeisti fragmento versijos. Operacija *issaugoti_nuoroda_i_faila()* išsaugo atsiųsto, bet dar nenaudojamo failo pavadinimą.

Serveris – tai klasė, naudojama atpažinti centrinę sistemą (*autentifikuoti()*). Serverio duomenų išgavimui skirta operacija *gauti_serverio_duomenis()*. Esamų centrinių sistemų duomenims išsaugoti kviečiama operacija *saugoti_saraso_duomenis()*, o naujų įvedimui – *irasyti_naujus_duomenis()*. Visų centrinių sistemų sąrašui gauti skirta operacija *gauti_sarasa()*.

Atnaujinimo_pranesimas – pranešimų apie fragmentų atnaujinimus apdorojimui skirta klasė. Operacija *issaugoti()* pranešimas išsaugomas, o *pakeisti_pranesimo_busena()* – pakeičiama jo būsena į tokią, kad atnaujinimas būtų laikomas atsiųstu. Duomenų išgavimui skirta operacija *gauti_pranesimo_duomenis()*.

Klaida – tai klasė, skirta klaidų registravimui. Tam naudojama šios klasės operacija *itraukti_info_apie_klaida()*.

Administratorius – klasė, analogiška tokio pat pavadinimo centrinės sistemos klasei.

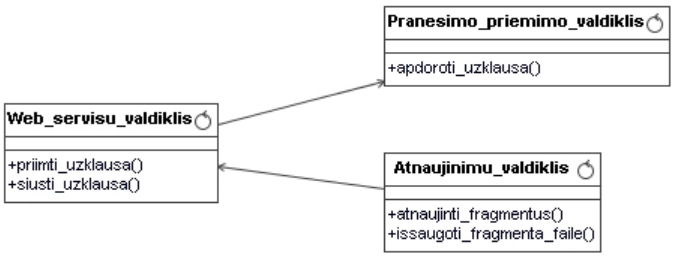
5.4.4.3. Bendravimo su centrine sistema posistemio detalus projektas

56 pav. parodyta bendravimo su centrine sistema posistemio veiklos logikos klasių diagrama. Šiame posistemyje veikla yra automatinė, todėl vartotojo sąsajos elementų čia nėra.

Web_servisu_valdiklis skirtas siųsti užklausas centrinei sistemai (*siusti_uzklausa()*) bei priimti užklausas iš jos (*priimti_uzklausa()*). Gavus užklausą iš centrinės sistemos, šis valdiklis grąžina jai tai, ką jam grąžina pranešimo priėmimo valdiklio operacija *apdoroti_uzklausa()*.

Atnaujinimu_valdiklis skirtas siųsti atnaujinimo užklausas centrinei sistemai (*atnaujinti_fragmentus()*). Siunčiama tiek kartų, kiek yra naujų pranešimų, ir reikalaujama, kad centrinė sistema grąžintų naujausią kiekvieno atnaujinto fragmento versiją. Operacija *issaugoti_fragmenta_faile()* atsiųstas atnaujintas kodo fragmentas išsaugomas tekstiniame faile.

Pranesimo_priemimo_valdiklis naudojamas Web servisų valdiklio, kai šis gauna užklausą iš centrinės sistemos. Operacija *apdoroti_uzklausa()* išsaugo pranešimą, jei centrinė sistema atpažįstama bei nustatoma, kad klientinė sistema naudoja pranešime nurodytą fragmentą. Priešingu atveju grąžinamas pranešimas apie klaidą.

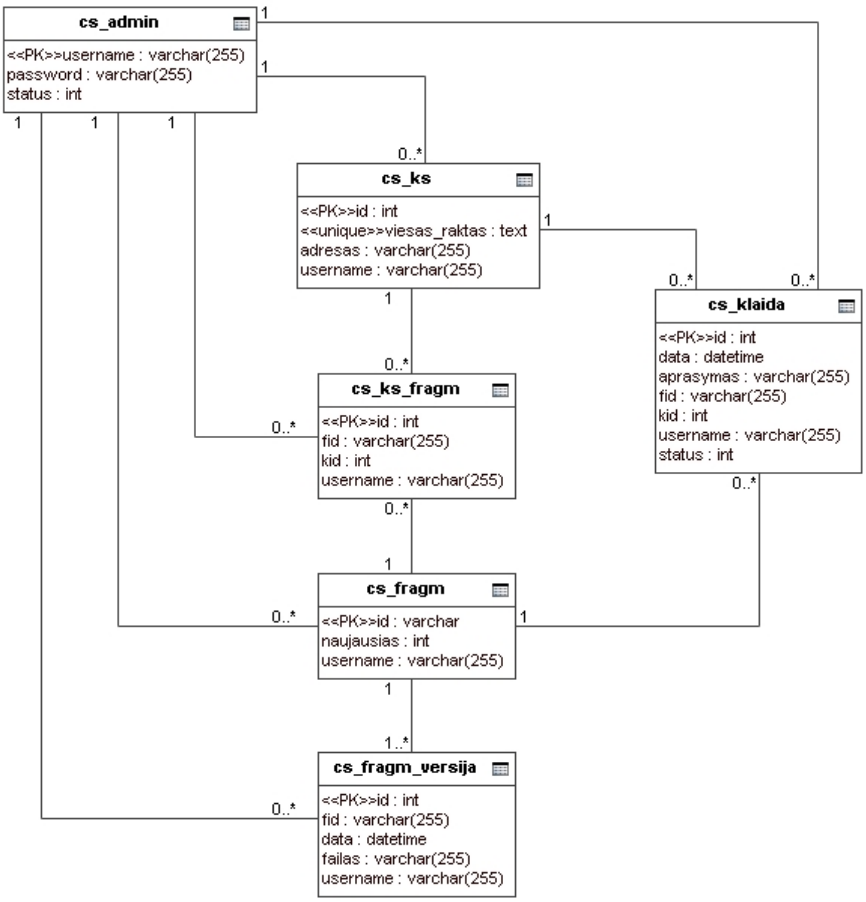


56 pav. Bendravimo su centrine sistema posistemio veiklos logikos klasių diagrama

5.5. Duomenų modelis

Šiame poskyryje pateikiamos centrinės bei klientinės sistemų duomenų bazių schemos, kurios gautos transformavus ir modifikavus reikalavimų specifikacijos klasių modelius (19 pav. ir 26 pav.). Kurioms klasėms nebuvo nurodyti pirminiai raktai, jas atitinkančiose DB lentelėse sukurti atskiri raktiniai laukai. Taip pat automatiškai sukurti išoriniai raktai iš susietų lentelių.

Centrinės sistemos duomenų bazės schema pateikta 57 pav. Jos lentelių bei jų elementų specifikacija pateikta 23 lentelėje.



57 pav. Centrinės sistemos DB schema

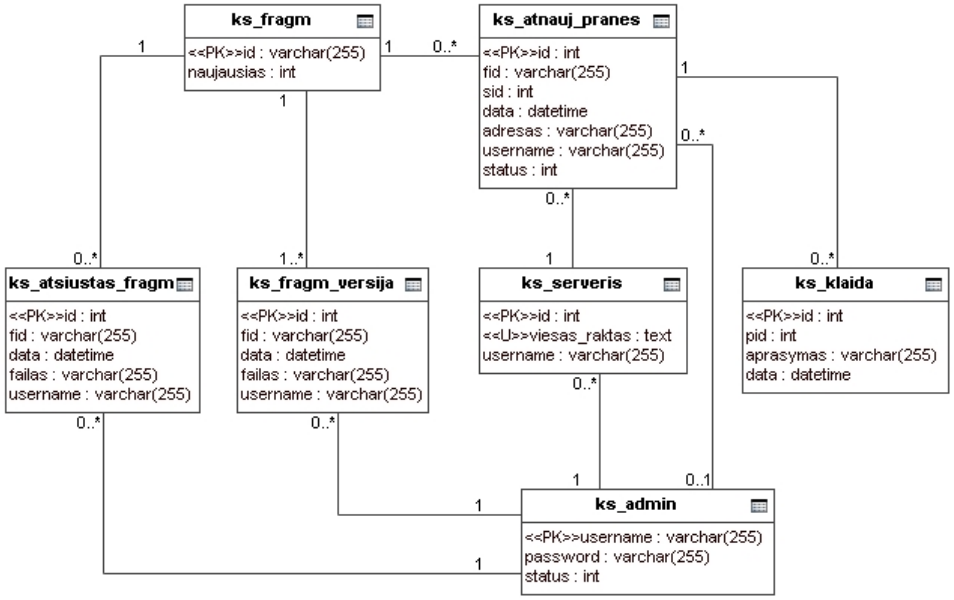
23 lentelė. Centrinės sistemos DB lentelių ir jų elementų specifikacija

Lent. / atrib.	Tipas	Paskirtis
cs_admin		Administratorių duomenys
username	varchar(255)	Pirminis raktas. Identifikuoja administratorių.
password	varchar(255)	Prisijungimo slaptažodis.
status	int	Parametras, rodantis, ar administratorius yra pagrindinis, ar ne.
cs_ks		Klientinių sistemų duomenys
id	int	Pirminis raktas. Identifikuoja klientinę sistemą.
viesas_raktas	text	Unikalus viešas raktas, skirtas klientinės sistemos autentifikavimui.
adresas	varchar(255)	URL adresas + kelias iki tinklo paslaugų variklio, skirtas pranešimų siuntimui.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, įvedusį ar redagavusį įrašą.
cs_fragm		Fragmentų duomenys
id	varchar(255)	Pirminis raktas. Identifikuoja fragmentą.
naujausias	int	Išorinis raktas. Naujausios fragmento versijos identifikatorius.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, įvedusį fragmentą.
cs_ks_fragm		Klientinių sistemų naudojamų fragmentų duomenys
id	int	Pirminis raktas. Identifikuoja lentelės įrašą.
fid	varchar(255)	Išorinis raktas. Identifikuoja fragmentą.
kid	int	Išorinis raktas. Identifikuoja klientinę sistemą.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, priskyrusį fragmentą klientinei sistemai.
cs_fragm_versija		Fragmentų versijų duomenys
id	int	Pirminis raktas. Identifikuoja lentelės įrašą.
fid	varchar(255)	Išorinis raktas. Identifikuoja fragmentą.
data	datetime	Įvedimo data ir laikas.
failas	varchar(255)	Failo, kuriame saugomas kodas, pavadinimas.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, įvedusį naują versiją.
cs_klaida		Klaidų duomenys
id	int	Pirminis raktas. Identifikuoja klaidą.
data	datetime	Klaidos užregistravimo data ir laikas.
aprasymas	varchar(255)	Klaidos aprašymas.
fid	varchar(255)	Išorinis raktas. Identifikuoja fragmentą.
kid	int	Išorinis raktas. Identifikuoja klientinę sistemą.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, pakartotinai išsiuntusį pranešimą.
status	int	Parametras, rodantis, ar klaida jau yra ištaisyta.

Klientinės sistemos duomenų bazės lentelių bei jų elementų specifikacija pateikta 24 lentelėje, o pati schema pateikta 58 pav.

24 lentelė. Klientinės sistemos DB lentelių ir jų elementų specifikacija

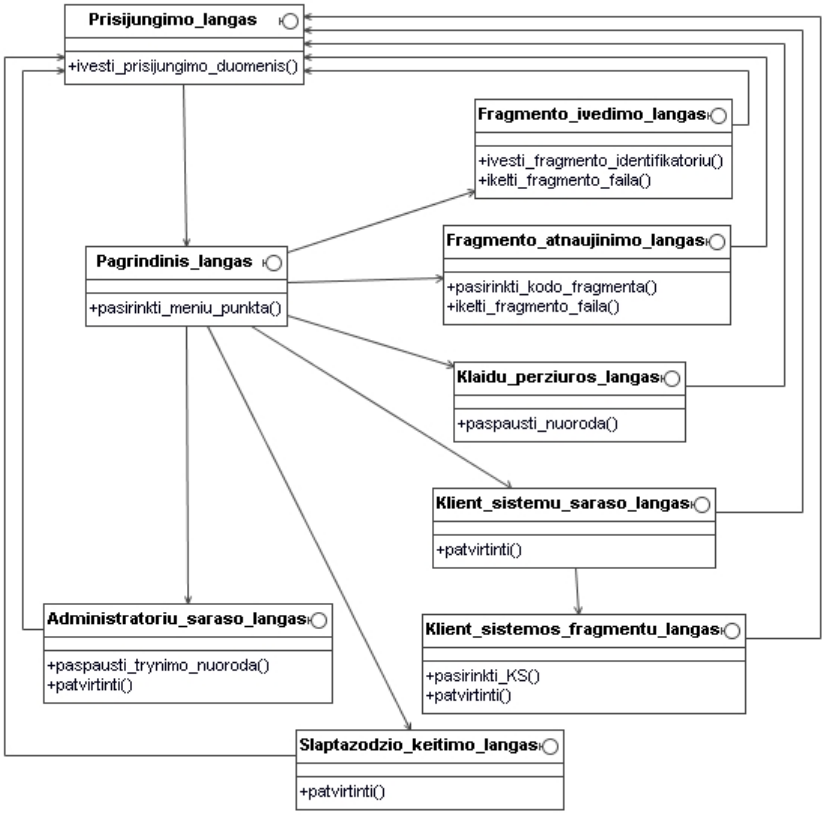
Lent. / atrib.	**Tipas**	**Paskirtis**
ks_admin		Administratorių duomenys
username	varchar(255)	Pirminis raktas. Identifikuoja administratorių.
password	varchar(255)	Prisijungimo slaptažodis.
status	int	Parametras, rodantis, ar administratorius yra pagrindinis, ar ne.
ks_atnauj_pranes		Atnaujinimo pranešimų duomenys
id	int	Pirminis raktas. Identifikuoja atnaujinimo pranešimą.
fid	varchar(255)	Išorinis raktas. Identifikuoja fragmentą.
sid	int	Išorinis raktas. Identifikuoja centrinės sistemos serverį.
data	datetime	Atsiuntimo data ir laikas.
adresas	varchar(255)	Centrinės sistemos URL adresas + kelias iki tinklo paslaugų variklio.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, aktyvavusį atnaujinimą.
status	int	Parametras, rodantis, ar atnaujinimas buvo sėkmingai aktyvuotas.
ks_fragm		Fragmentų duomenys
id	varchar(255)	Pirminis raktas. Identifikuoja fragmentą.
naujausias	int	Išorinis raktas. Naujausios fragmento versijos identifikatorius.
ks_serveris		Centrinių sistemų serverių duomenys
id	varchar(255)	Pirminis raktas. Identifikuoja fragmentą.
viesas_raktas	text	Unikalus viešas raktas, skirtas centrinės sistemos autentifikavimui.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, įvedusį arba redagavusį įrašą.
ks_atsiustas_fragm		Atsiųstų, bet dar nenaudojamų fragmentų duomenys
id	int	Pirminis raktas. Identifikuoja lentelės įrašą.
fid	varchar(255)	Išorinis raktas. Identifikuoja fragmentą.
data	datetime	Įvedimo data ir laikas.
failas	varchar(255)	Failo, kuriame saugomas kodas, pavadinimas.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, aktyvavusį atnaujinimo siuntimą.
ks_fragm_versija		Fragmentų versijų duomenys
id	int	Pirminis raktas. Identifikuoja lentelės įrašą.
fid	varchar(255)	Išorinis raktas. Identifikuoja fragmentą.
data	datetime	Įvedimo data ir laikas.
failas	varchar(255)	Failo, kuriame saugomas kodas, pavadinimas.
username	varchar(255)	Išorinis raktas. Identifikuoja administratorių, aktyvavusį atnaujinimo siuntimą.
ks_klaida		Klaidų duomenys
id	int	Pirminis raktas. Identifikuoja klaidą.
pid	int	Išorinis raktas. Identifikuoja atnaujinimo pranešimą.
aprasymas	varchar(255)	Klaidos aprašymas.
data	datetime	Nesėkmės data ir laikas.



58 pav. Klientinės sistemos DB schema

5.6. Vartotojo sąsajos modelis

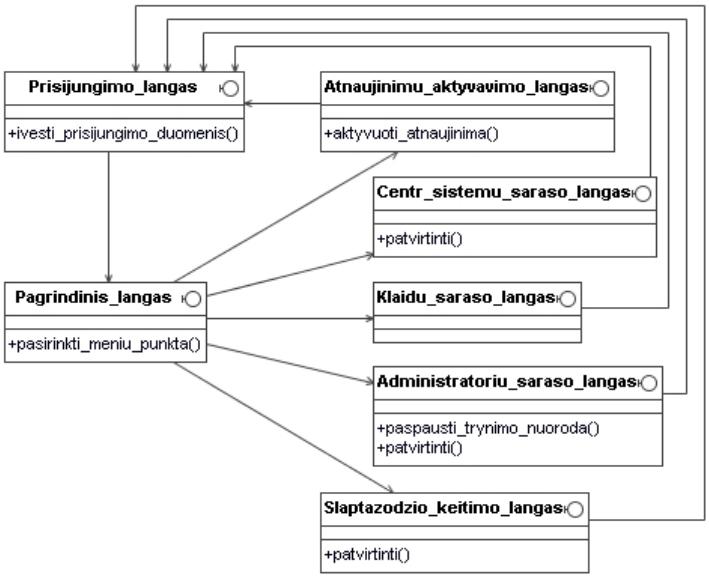
Šiame poskyryje pateikiami centrinės bei klientinės sistemų navigavimo planai. 59 pav. pateiktas centrinės sistemos navigavimo planas rodo, į kokius langus galima patekti iš kiekvieno lango. Prisijungus patenkama į pagrindinį langą, o iš jo galima pereiti į kitus langus.



59 pav. Centrinės sistemos navigavimo planas

Iš klientinių sistemų sąrašo galima patekti į vienos konkrečios klientinės sistemos naudojamų fragmentų valdymo langą. Jei esant kuriame nors lange baigiasi sesija, tuomet automatiškai yra nukreipiama į prisijungimo langą.

60 pav. pateiktas klientinės sistemos navigavimo planas rodo, į kokius langus galima patekti iš kiekvieno lango. Prisijungus patenkama į pagrindinį langą, o iš jo galima pereiti į atnaujinimų aktyvavimo ar kitą langą. Jei esant kuriame nors lange baigiasi sesija, tuomet automatiškai yra nukreipiama į prisijungimo langą.



60 pav. Klientinės sistemos navigavimo planas

5.7. Sistemos elgsenos modeliai

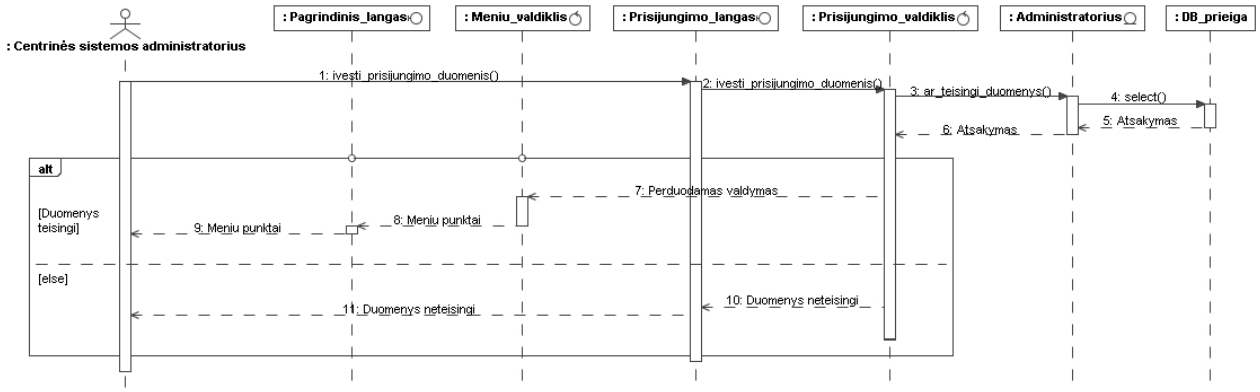
Šiame poskyryje pateikiamos centrinės bei klientinės sistemų elgsenos diagramos.

5.7.1. Centrinės sistemos elgsenos modelis

Toliau pateikiamos pagrindinių centrinės sistemos posistemių panaudojimo atvejų sekų diagramos. Jose parodoma, kaip centrinės sistemos administratorius ir/arba klientinė sistema sąveikauja su centrine sistema, kokia tvarka kokie pranešimai siunčiami, kokios klasių objektų operacijos kviečiamos.

5.7.1.1. Centrinės sistemos prisijungimo posistemio sekų diagramos

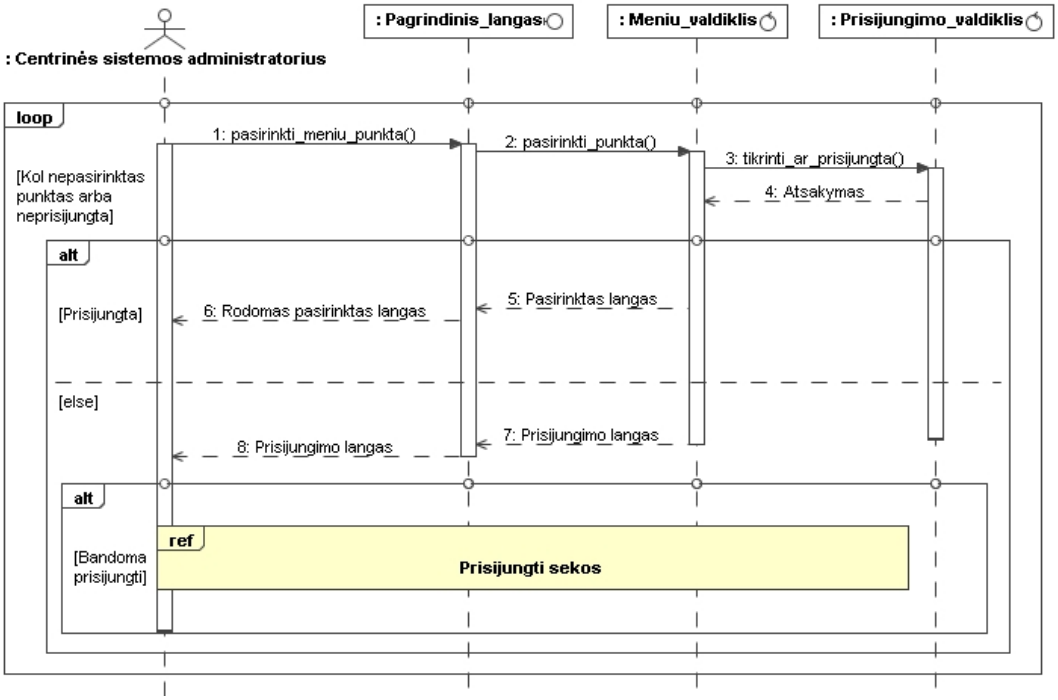
61 pav. pateiktas detalus panaudojimo atvejo „Prisijungti“ sekų modelis.



61 pav. PA „Prisijungti“ sekų modelis projektavimo etape

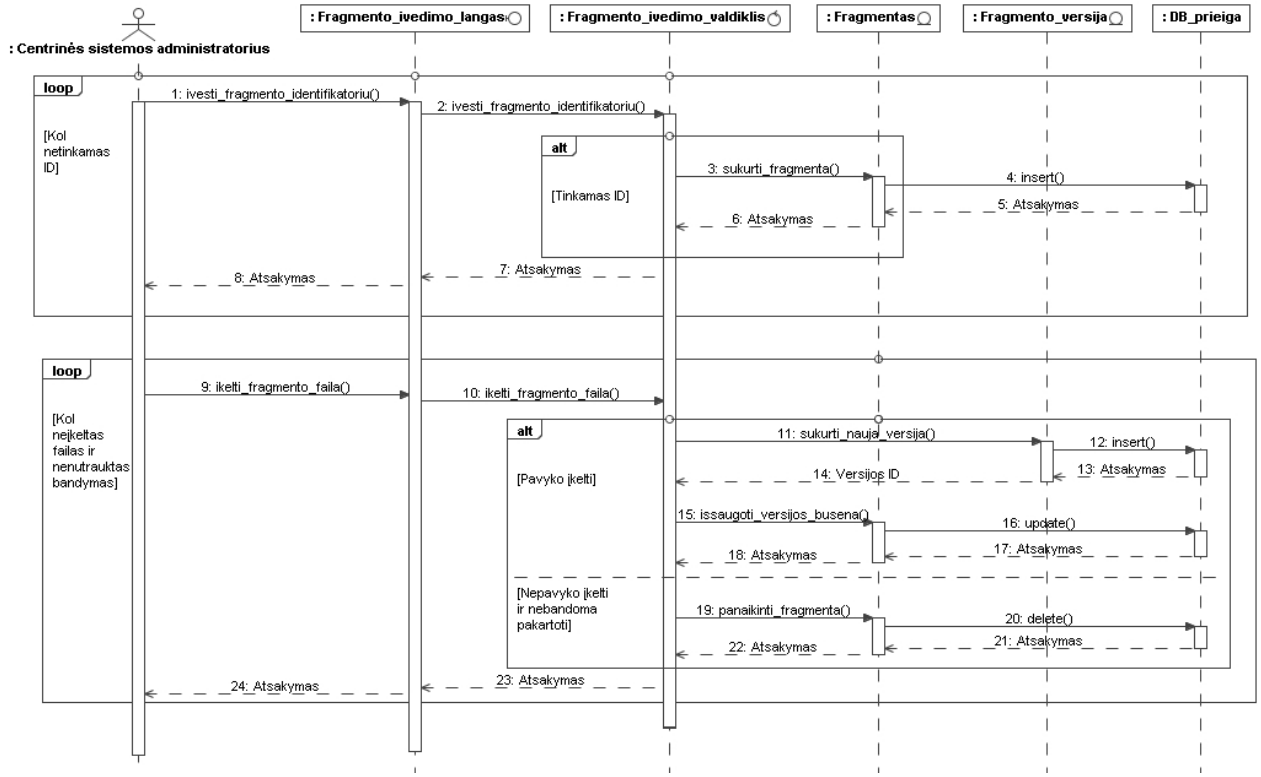
5.7.1.2. Centrinės sistemos administravimo posistemio sekų diagramos

62 pav. pateiktas detalus panaudojimo atvejo „Prižiūrėti sistemą“ sekų modelis.



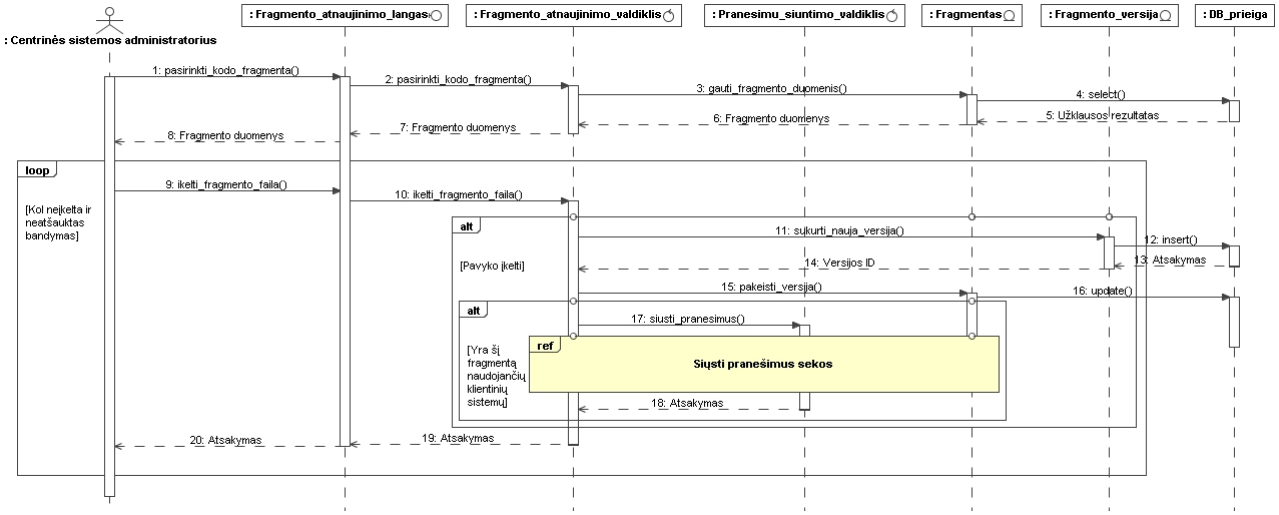
62 pav. PA „Prižiūrėti sistemą“ sekų modelis projektavimo etape

63 pav. pateiktas detalus panaudojimo atvejo „Įtraukti naują kodo fragmentą“ sekų modelis.



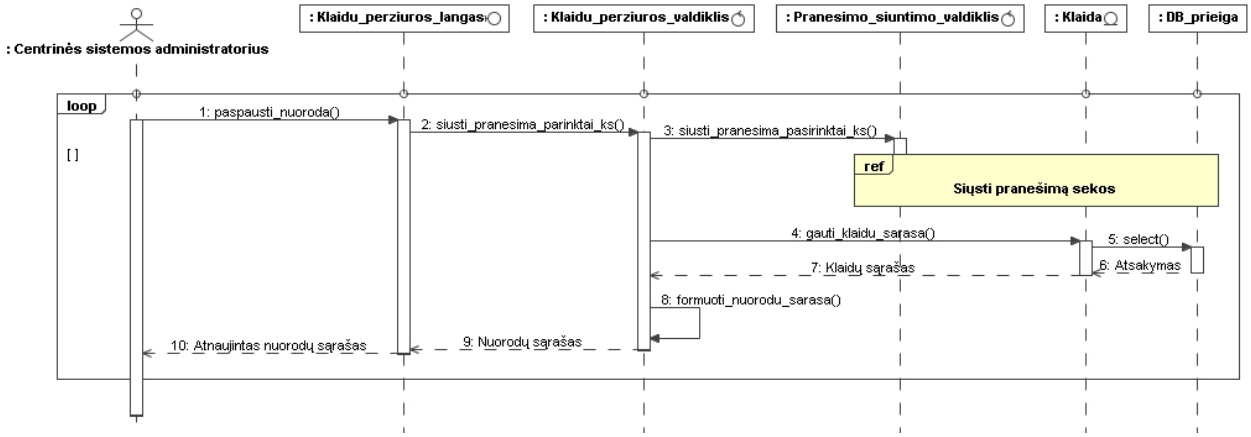
63 pav. PA „Įtraukti naują kodo fragmentą“ sekų modelis projektavimo etape

64 pav. pateiktas detalus panaudojimo atvejo „Atnaujinti esamą kodo fragmentą“ sekų modelis.



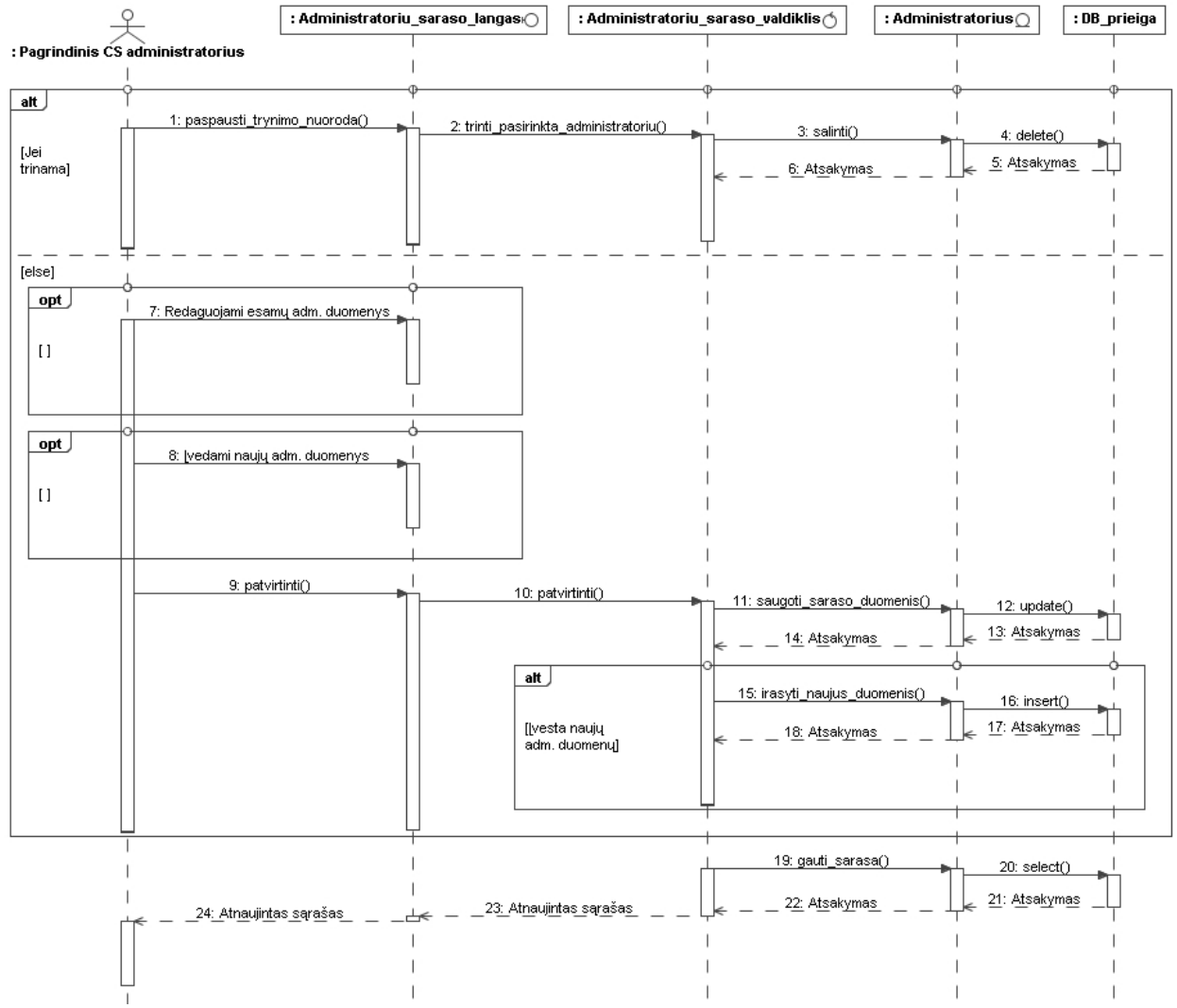
64 pav. PA „Atnaujinti esamą kodo fragmentą“ sekų modelis projektavimo etape

65 pav. pateiktas detalus panaudojimo atvejo „Peržiūrėti klaidų pranešimus“ sekų modelis.



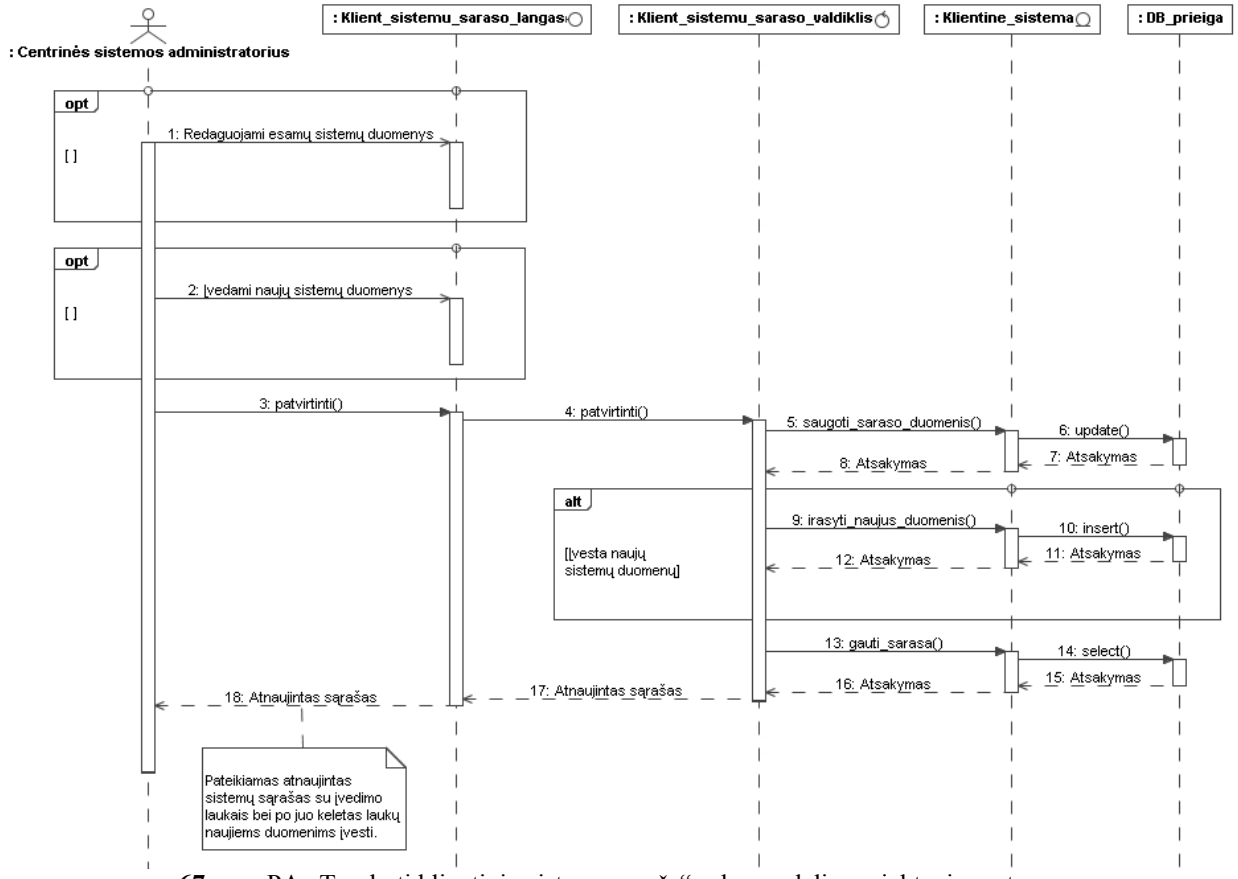
65 pav. PA „Peržiūrėti klaidų pranešimus“ sekų modelis projektavimo etape

66 pav. pateiktas detalus panaudojimo atvejo „Tvarkyti administratorių sąrašą“ sekų modelis.



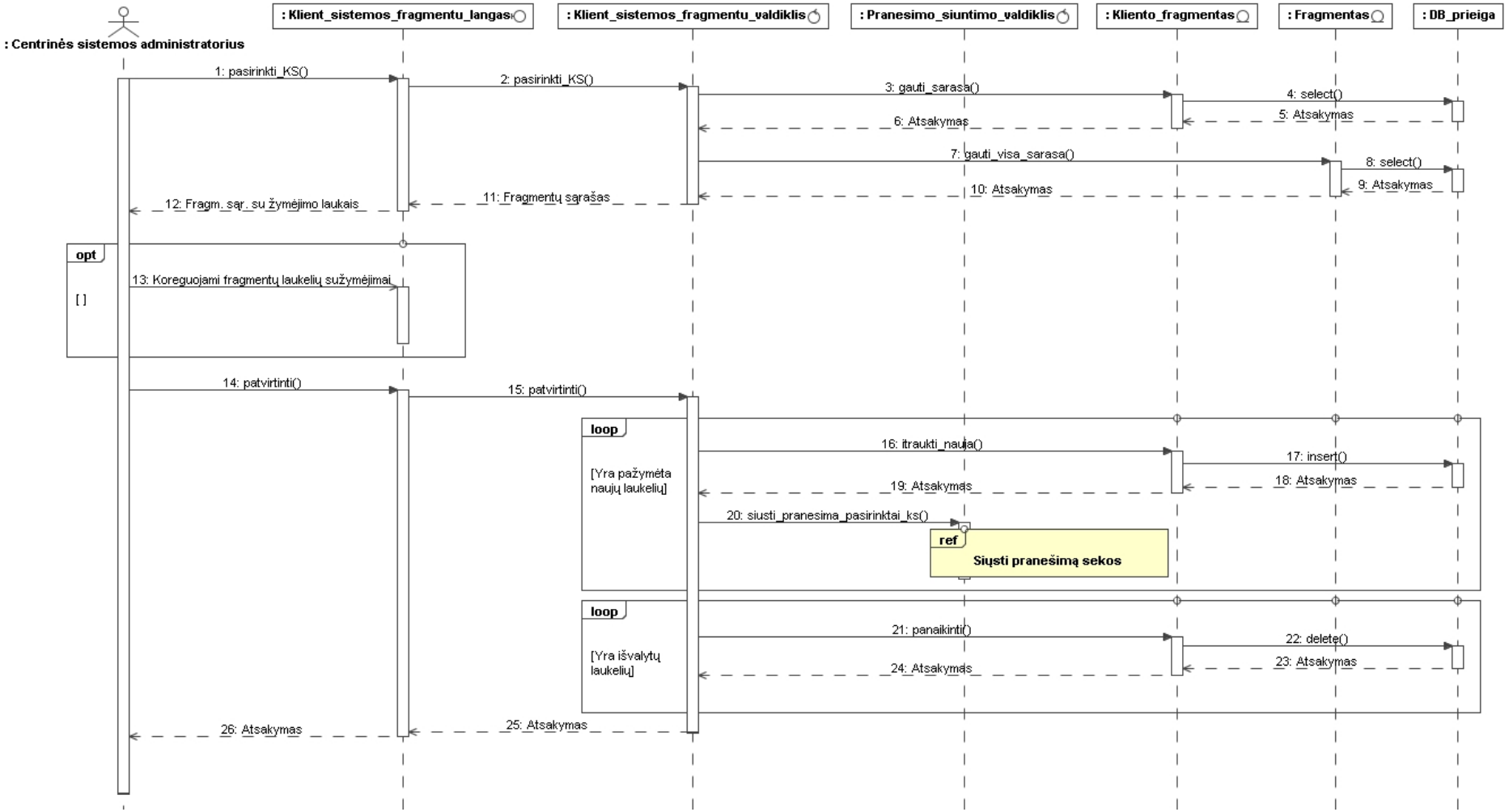
66 pav. PA „Tvarkyti administratorių sąrašą“ sekų modelis projektavimo etape

67 pav. pateiktas detalus panaudojimo atvejo „Tvarkyti klientinių sistemų sąrašą“ sekų modelis.



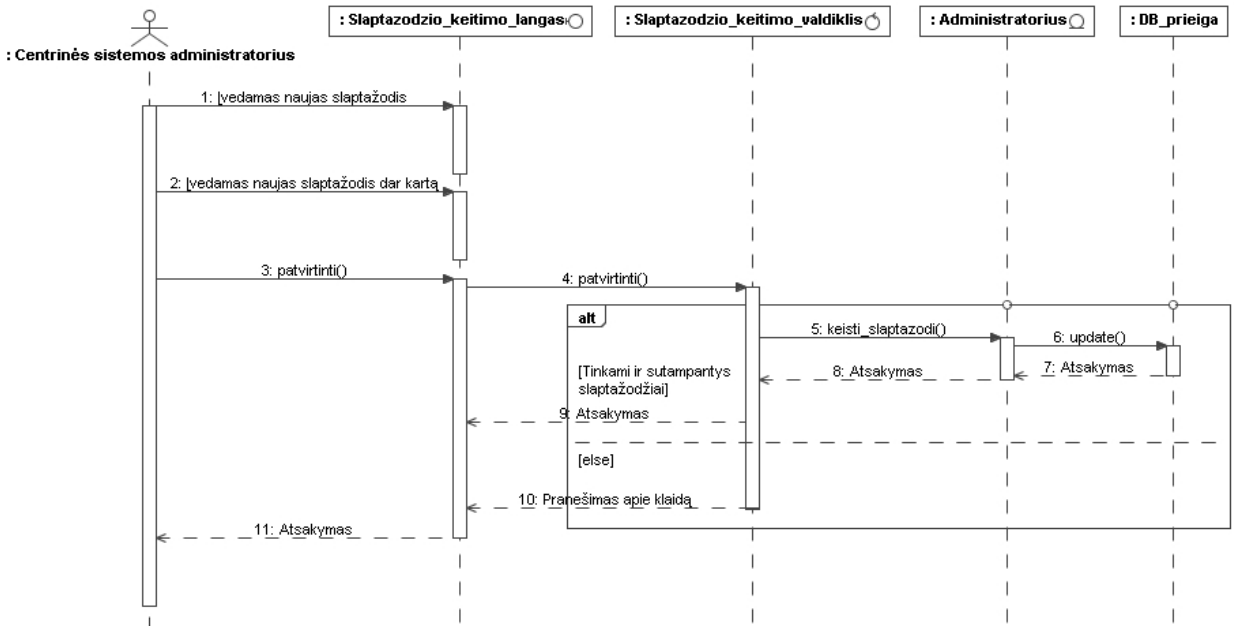
67 pav. PA „Tvarkyti klientinių sistemų sąrašą“ sekų modelis projektavimo etape

68 pav. – detalus panaudojimo atvejo „Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą“ sekų modelis.



68 pav. PA „Tvarkyti klientinės sistemos naudojamų fragmentų sąrašą“ sekų modelis projektavimo etape

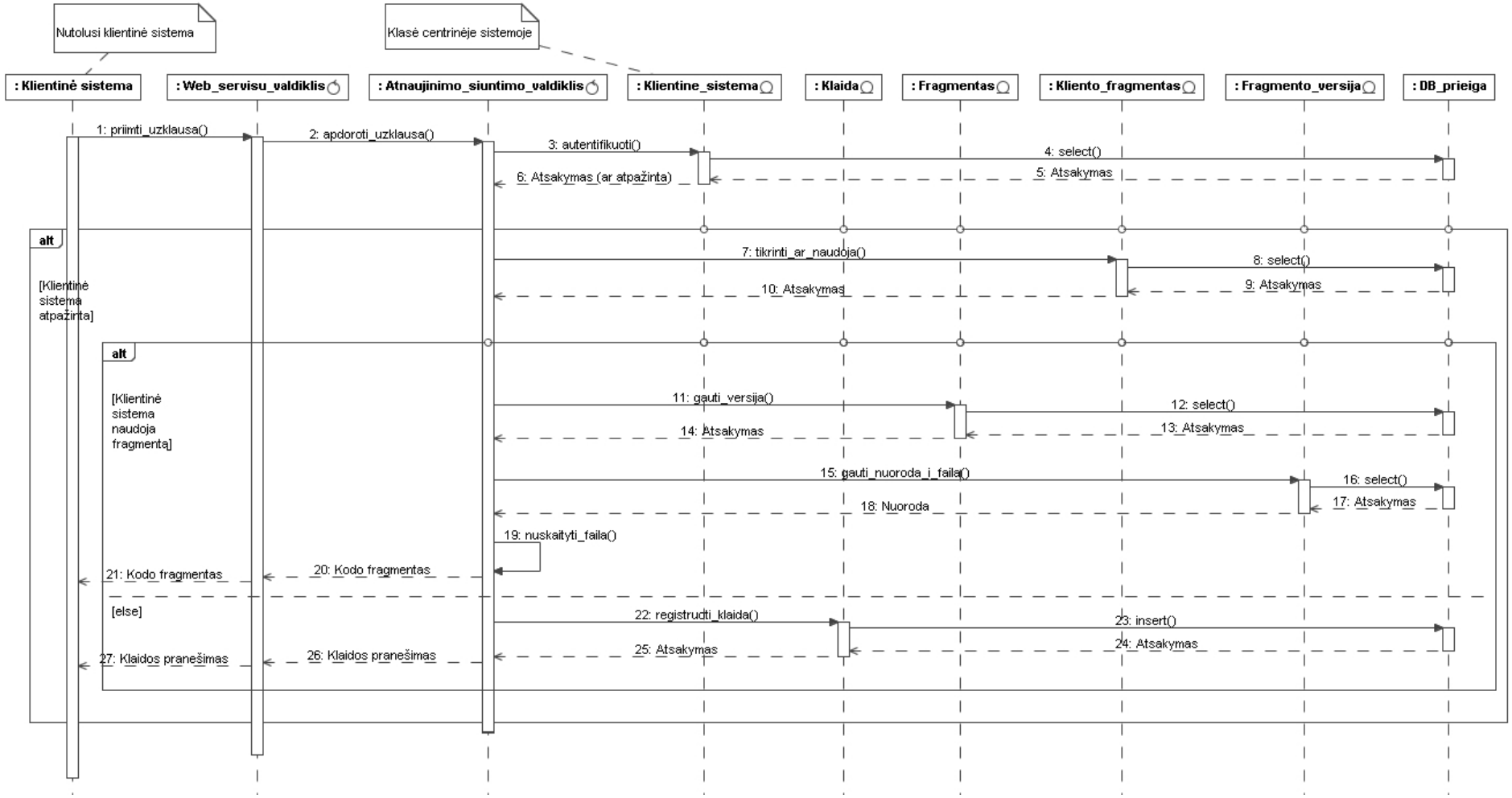
69 pav. pateiktas detalus panaudojimo atvejo „Keisti prisijungimo slaptažodį“ sekų modelis.



69 pav. PA „Keisti prisijungimo slaptažodį“ sekų modelis projektavimo etape

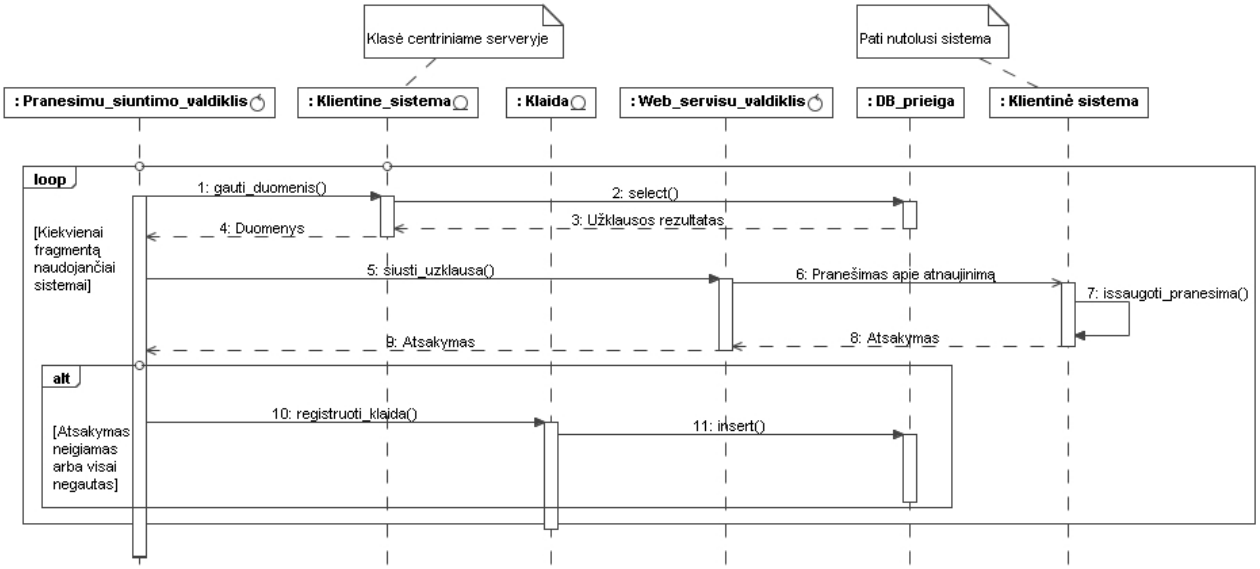
5.7.1.3. Bendravimo su klientine sistema posistemio sekų diagramos

70 pav. pateiktas detalus panaudojimo atvejo „Siųsti atnaujinimą klientinei sistemai“ sekų modelis.

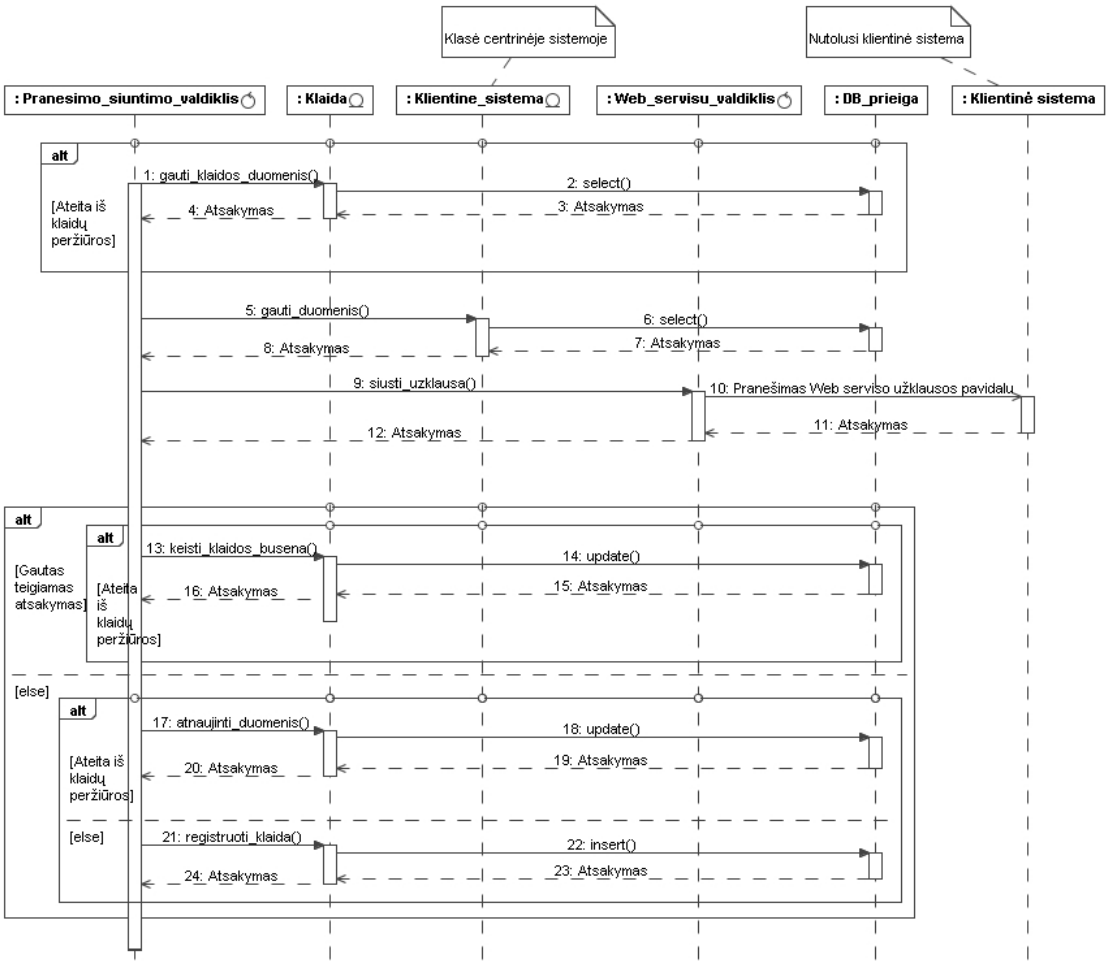


70 pav. PA „Siųsti atnaujinimą klientinei sistemai“ sekų modelis projektavimo etape

71 pav. pateiktas detalus panaudojimo atvejo „Siųsti pranešimus klientinėms sistemoms“, o 72 pav. – „Siųsti pranešimą klientinei sistemai“ sekų modeliai.



71 pav. PA „Siųsti pranešimus klientinėms sistemoms“ sekų modelis projektavimo etape



72 pav. PA „Siųsti pranešimą klientinei sistemai“ sekų modelis projektavimo etape

5.7.2. Klientinės sistemos elgsenos modelis

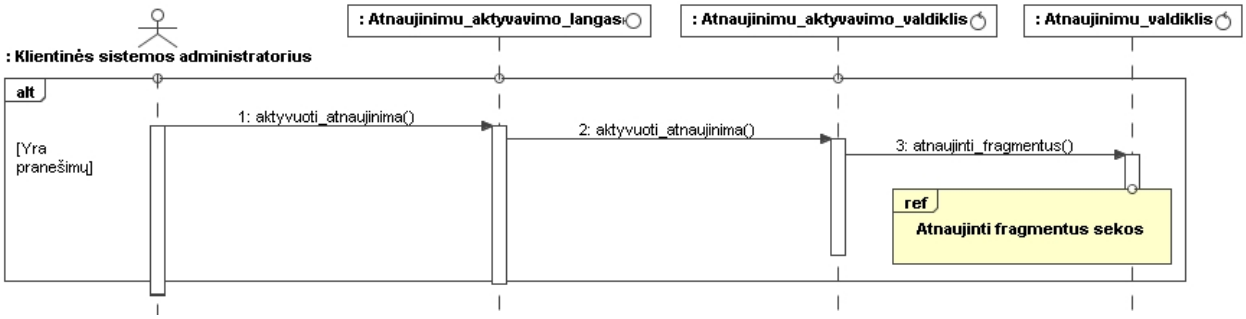
Toliau analogiškai pateikiamos pagrindinių klientinės sistemos posistemių panaudojimo atvejų sekų diagramos. Jose parodoma, kaip klientinės sistemos administratorius ir/arba centrinė sistema sąveikauja su klientine sistema, kokia tvarka kokie pranešimai siunčiami, kokios klasių objektų operacijos kviečiamos.

5.7.2.1. Klientinės sistemos prisijungimo posistemio sekų diagramos

Klientinės sistemos panaudojimo atvejo „Prisijungti“ sekų diagrama yra identiška analogiško centrinės sistemos panaudojimo atvejo sekų diagramai (61 pav.).

5.7.2.2. Klientinės sistemos administravimo posistemio sekų diagramos

73 pav. pateiktas detalus panaudojimo atvejo „Peržiūrėti atnaujinimų pranešimus“ sekų modelis.



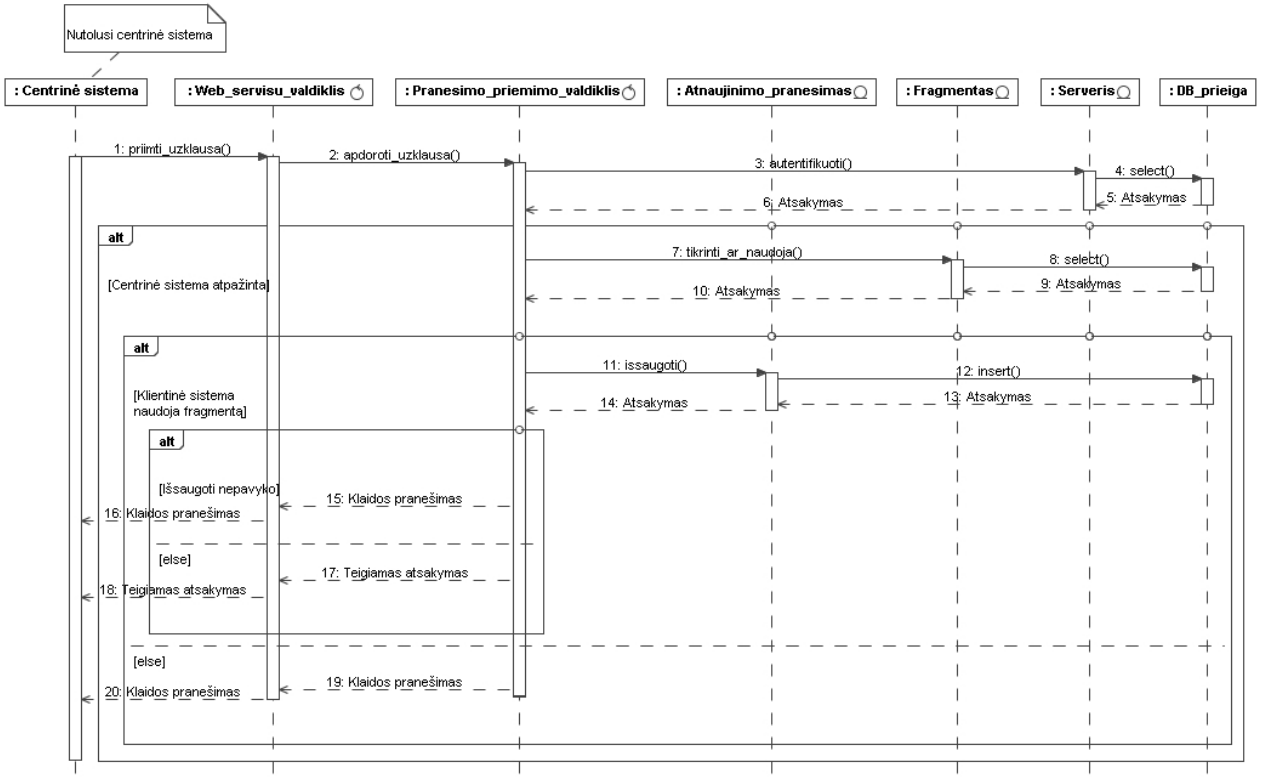
73 pav. PA „Peržiūrėti atnaujinimų pranešimus“ sekų modelis projektavimo etape

Klientinės sistemos panaudojimo atvejo „Prižiūrėti klientinę sistemą“ sekų diagrama yra identiška analogiško centrinės sistemos panaudojimo atvejo sekų diagramai (62 pav.). Taip pat sutampa ir panaudojimo atvejų „Tvarkyti administratorių sąrašą“ (66 pav.) bei „Keisti prisijungimo slaptažodį“ (69 pav.) sekų modeliai.

Panaudojimo atvejyje „Peržiūrėti klaidų sąrašą“ yra tiktai išvedama informacija, todėl jokių sąveikų su vartotoju nėra. O „Tvarkyti centrinių sistemų sąrašą“ atitinka centrinės sistemos panaudojimo atvejį „Tvarkyti klientinių sistemų sąrašą“ (67 pav.), išskyrus tai, kad čia nėra papildomų nuorodų į sistemų naudojamų fragmentų sąrašus.

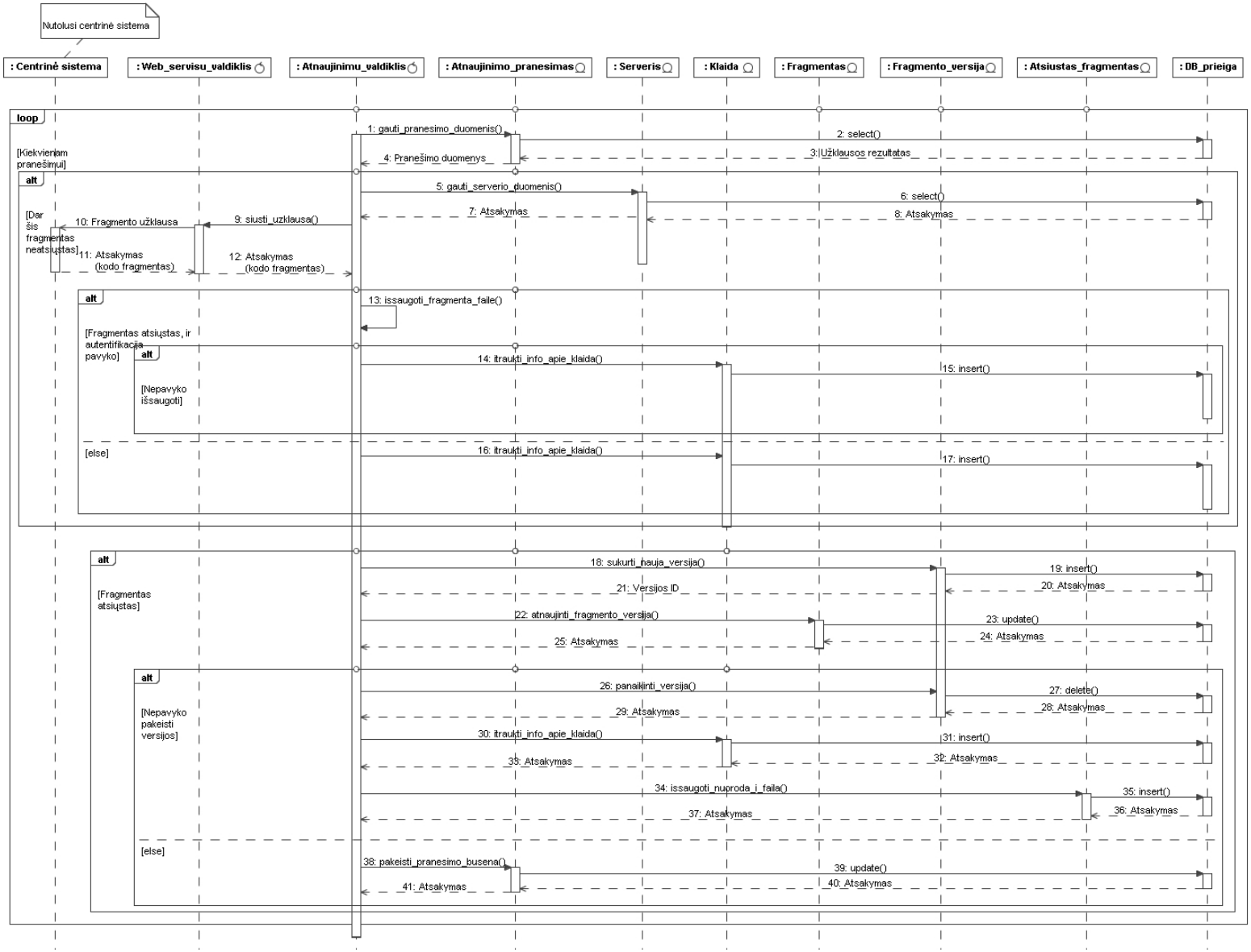
5.7.2.3. Bendravimo su centrine sistema posistemio sekų diagramos

74 pav. pateiktas detalus panaudojimo atvejo „Išsaugoti pranešimą apie atnaujinimą“ sekų modelis.



74 pav. PA „Išsaugoti pranešimą apie atnaujinimą“ sekų modelis projektavimo etape

75 pav. pateiktas detalus panaudojimo atvejo „Atnaujinti kodo fragmentus“ sekų modelis.

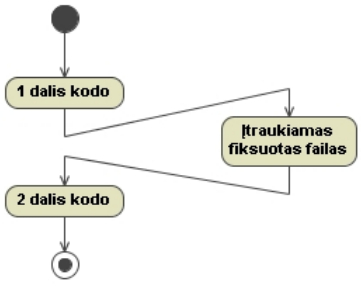


75 pav. PA „Atnaujinti kodo fragmentus“ sekų modelis projektavimo etape

5.8. Metodų, algoritmų, technologijų, atspindinčių darbo idėją bei naujumą, aprašas

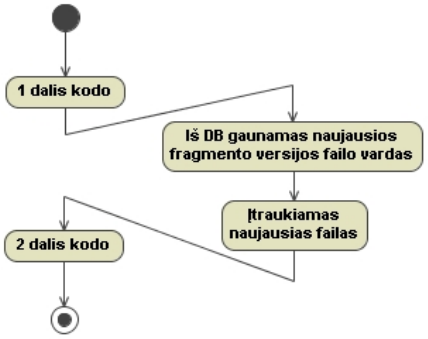
Darbe modeliuojamos dvi – centrinė ir klientinė – sistemos bei sąveika tarp jų. Projekto modelio diagramose atskleidžiamos visos šių sistemų sąveikos subtilybės, tad šiame skyriuje parodoma, kuo šiuo modeliu bei centralizuota architektūra paremtas informacijos sistemų kūrimas yra pranašesnis už įprastus metodus.

76 pav. parodyta, kaip PHP ar kita kalba formuojamas HTML kodas įprastose užsakovų sistemose. Vykdant pagrindinį kodo formavimo scenarijų tenka įtraukti kitus failus, kurių pavyzdžiai pateikti 10 skyriuje (1 priedas). Tačiau tie failai yra fiksuoti, t. y. niekada neatnaujinami arba atnaujinami rankiniu būdu kiekvienoje sistemoje atskirai.



76 pav. Įprasto klientinės sistemos HTML teksto formavimo schemos fragmentas

77 pav. parodyta patobulinta schema, kuri atspindi šiame darbe kuriamą modelį. Scenarijus iš pirmo žvilgsnio lieka tas pats – PHP ar kita kalba formuojant HTML tekstą tenka įtraukti papildomus failus. Tačiau šioje vietoje scenarijus keičiasi. Vietoje to, kad būtų įtraukiamas fiksuotas failas, yra kreipiamasi į duomenų bazę, kad būtų gautas konkretaus fragmento naujausios versijos failo vardas. Tuomet įtraukiamas būtent naujausias tam fragmentui skirtas failas. Kad nereikėtų daug kartų kreiptis į duomenų bazę, prieš pradedant formuoti kodą galima iš jos išgauti visų naudojamų fragmentų naujausių failų vardus.



77 pav. Atnaujinamos klientinės sistemos HTML teksto formavimo schemos fragmentas

Taigi centrinės bei klientinės sistemų modeliai užtikrina, kad kodo fragmentai visada bus atnaujinti. O užsakovo informacijos sistemą realizuojant pagal 77 pav. pateiktą schemą bus užtikrintas naujausių fragmentų versijų panaudojimas.

6. SISTEMOS PROTOTIPO REALIZACIJA

Realizacijos etape, remiantis projektavimo etapo modeliais bei specifikacijomis, sukuriamias veikiantis centrinės bei klientinės sistemų prototipas bei realizuojama keletas eksperimentinių pavyzdžių, besiskiriančių naudojamų fragmentų dydžiu bei kiekiu. Šiame skyriuje aprašomas sistemos realizacijos modelis bei jos diegimo instrukcija, taip pat testavimo planas bei rezultatai.

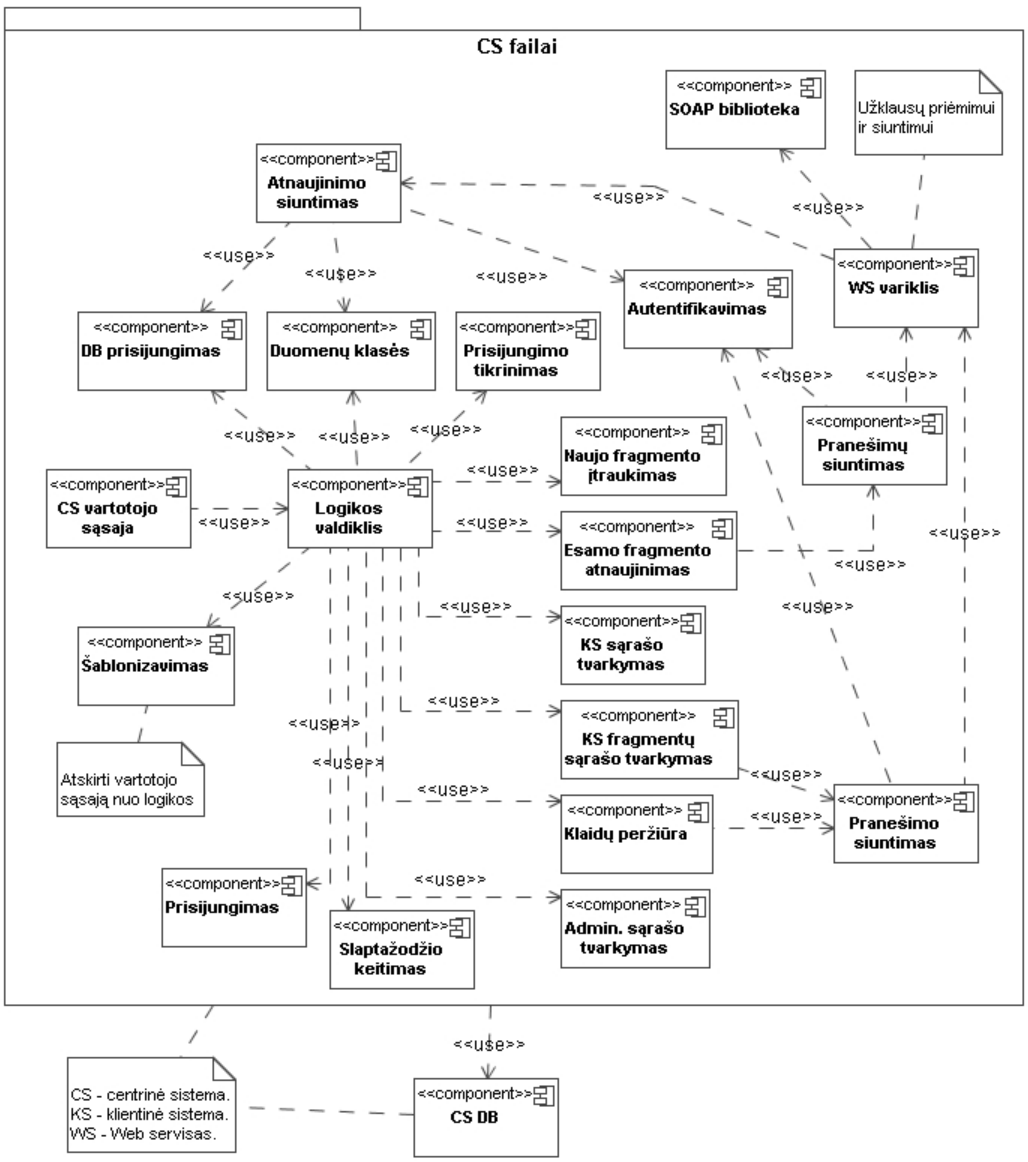
6.1. Centrinės sistemos realizacijos modelis

Šiame poskyryje parodoma, kokie loginiai komponentai sudaro centrinę sistemą, kokiais fiziniais artefaktais jie realizuojami. Taip pat parodoma, kokiuose įrenginiuose artefaktai egzistuoja.

78 pav. pateiktas centrinės sistemos komponentų modelis. Vartotojo sąsajai skirtas vienas komponentas *CS vartotojo sąsaja*, o jo naudojamas *Logikos valdiklis*, remdamasis gaunamais GET ir/arba POST parametrais, nusprendžia, kokį langą suformuoti, įtraukdamas vieną iš šių komponentų: *Prisijungimas*, *Naujo fragmento įtraukimas*, *Esamo fragmento atnaujinimas*, *KS sąrašo tvarkymas*, *KS fragmentų sąrašo tvarkymas*, *Klaidų peržiūra*, *Admin. sąrašo tvarkymas*, *Slaptažodžio keitimas*. Visiems išvardintiems komponentams priklauso ir juos atitinkantys vartotojo sąsajos komponentai, tačiau dėl diagramos kompaktiškumo jie nevaizduojami. Vartotojo sąsajos elementai galiausiai yra užpildomi sugeneruota informacija, panaudojant komponentą *Šablonizavimas*.

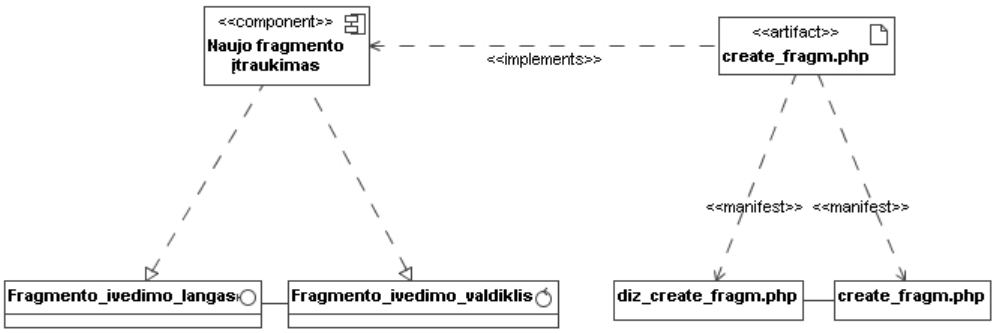
Jei reikia, kai kurie iš šių komponentų panaudoja pranešimų siuntimui klientinėms sistemoms skirtus komponentus (*Pranešimo siuntimas* arba *Pranešimų siuntimas*), kurie naudoja tinklo paslaugų variklį (*WS variklis*) bei autentifikavimo komponentą (*Autentifikavimas*) skaitmeninio parašo sudarymui. Jei ateina užklausa iš klientinės sistemos, pirmiausiai ji priimama to paties tinklo paslaugų variklio, kuris naudoja komponentą *Atnaujinimo siuntimas*. Šis savo ruožtu dar turi atpažinti klientinę sistemą, taip pat naudodamas komponentą *Autentifikavimas*.

Dauguma komponentų kreipiasi į centrinės sistemos duomenų bazę, tad paprastumo dėlei modelyje pavaizduota, kad į DB kreipiamasi iš viso centrinės sistemos failų paketo.



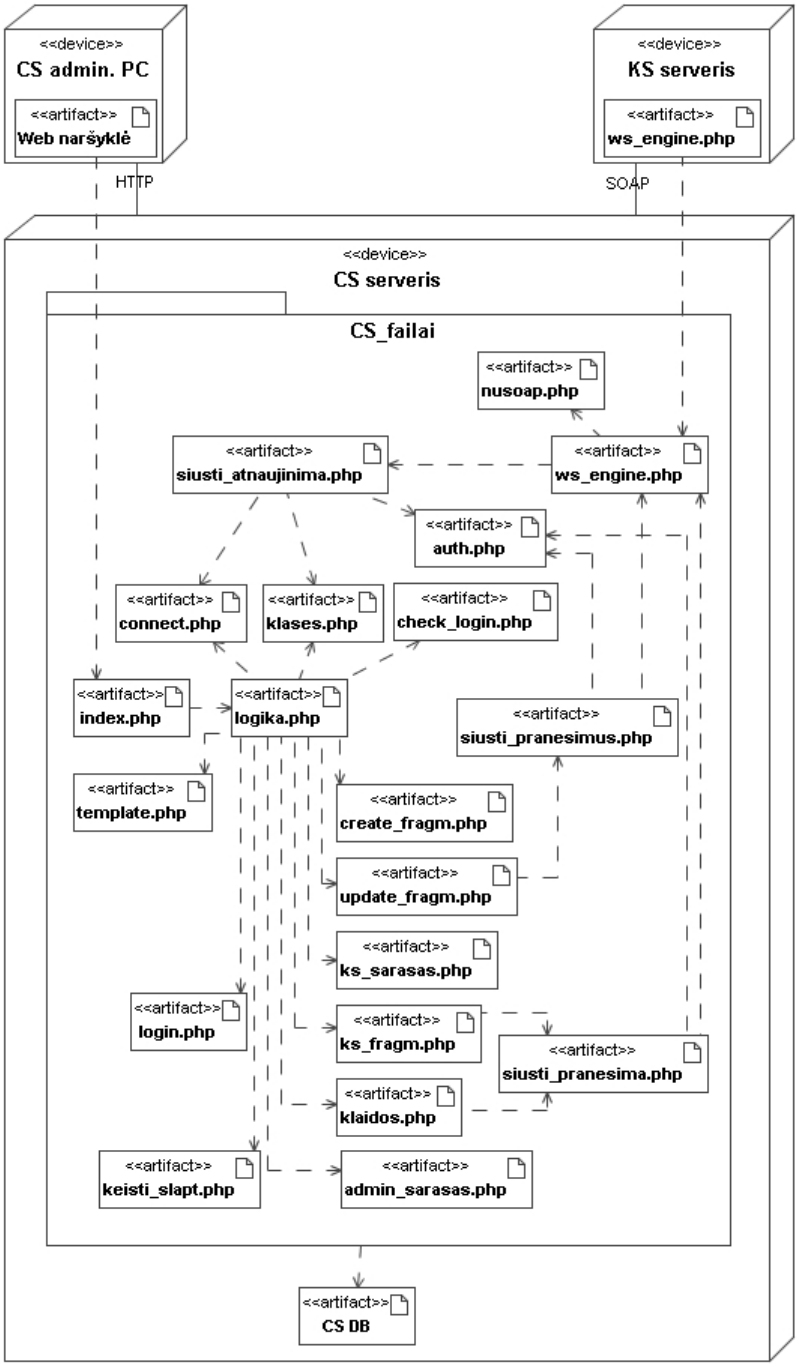
78 pav. Centrinės sistemos komponentų modelis

Kadangi kiekvieną komponentą atitinka jį realizuojantis artefaktas (fizinis failas serveryje), čia parodoma tik vieno komponento realizavimas artefaktu (79 pav.).



79 pav. Komponentų realizavimas artefaktais

80 pav. pateiktas centrinės sistemos įdiegimo modelis. Jame jau vaizduojami ne komponentai, o juos atitinkantys artefaktai. Parodyta, kad visi centrinės sistemos failai bei jos duomenų bazė yra viename Web serveryje. Su administratoriaus kompiuteriu serveris bendrauja HTTP, o su klientine sistema – SOAP [1, 2] protokolais.



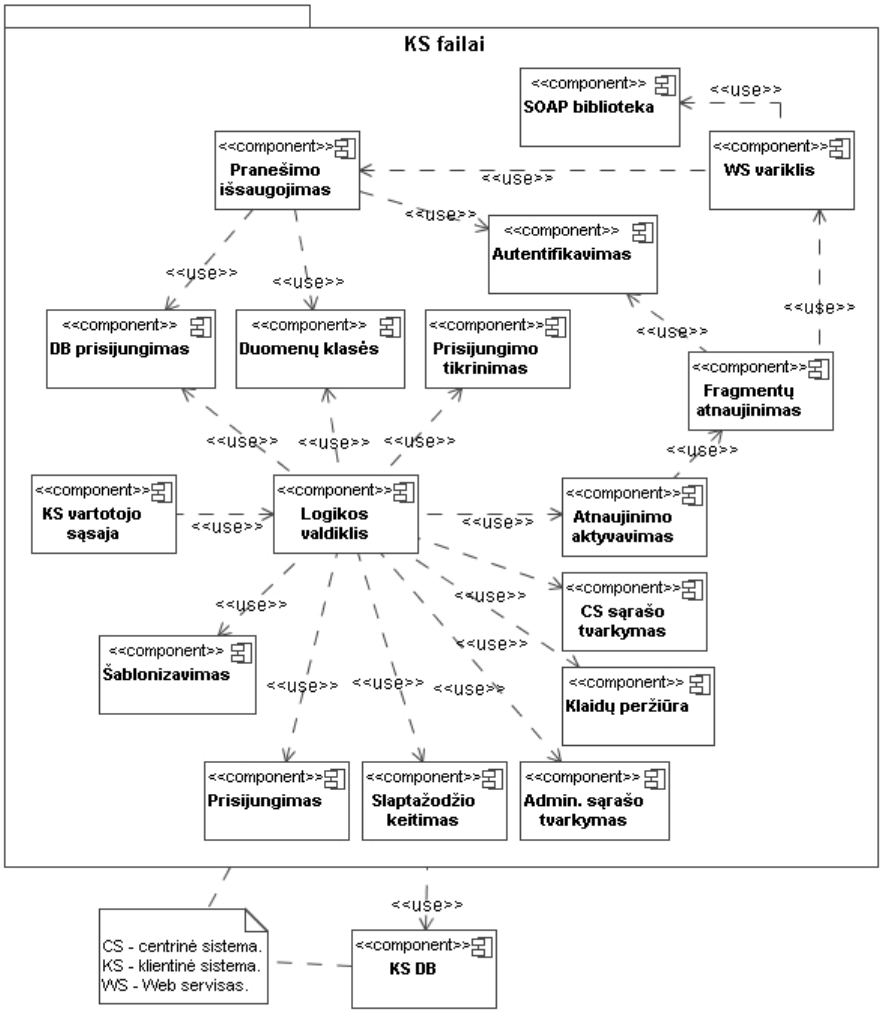
80 pav. Centrinės sistemos įdiegimo modelis

6.2. Klientinės sistemos realizacijos modelis

Šiame poskyryje parodoma, kokie loginiai komponentai sudaro klientinę sistemą, kokiais fiziniais artefaktais jie realizuojami. Taip pat parodoma, kokiuose įrenginiuose artefaktai egzistuoja.

81 pav. pateiktas klientinės sistemos komponentų modelis. Jame, kaip ir centrinės sistemos komponentų modelyje, visi komponentai patalpinti viename pakete. Kadangi duomenų bazę naudoja dauguma komponentų, kreipinys į ją paprastumo dėlei rodomas ne atskirai iš kiekvieno komponento, bet iš viso paketo.

Vartotojo sąsajai skirtas komponentas *KS vartotojo sąsaja*, kurio naudojamas *Logikos valdiklis* nusprendžia, kurį iš šių komponentų įtraukti: *Prisijungimas*, *Atnaujinimo aktyvavimas*, *CS sąrašo tvarkymas*, *Klaidų peržiūra*, *Admin. sąrašo tvarkymas* ar *Slaptažodžio keitimas*. Kaip ir centrinės sistemos realizacijos modelyje, šie komponentai turi juos atitinkančius, tačiau diagramoje nevaizduojamus vartotojo sąsajai skirtus komponentus.

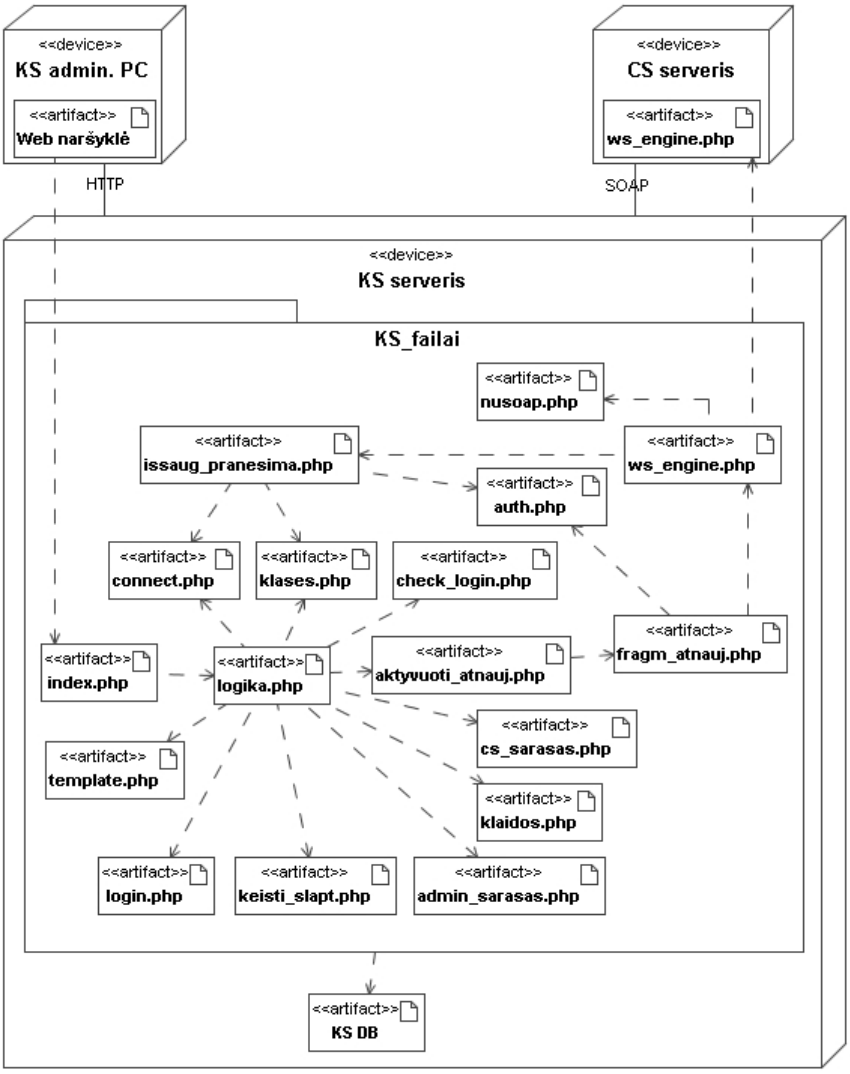


81 pav. Klientinės sistemos komponentų modelis

Jeigu aktyvuojamas atnaujinimas, naudojamas komponentas *Fragmentų atnaujinimas*, kuris naudoja komponentą *Autentifikavimas* parašo generavimui ir kreipiasi į tinklo paslaugų variklį. O jei ateina pranešimas apie atnaujinimą iš centrinės sistemos tinklo paslaugos užklausos pavidalu, tuomet tas pats tinklo paslaugų variklis priima užklausą bei naudoja pranešimo išsaugojimo

komponentą. Šis privalo autentifikuoti siuntėją, t. y. centrinę sistemą, kad būtų išvengta nesankcionuotų bandymų pakenkti sistemai.

82 pav. pateiktas klientinės sistemos įdiegimo modelis. Jame, kaip ir analogiškame centrinės sistemos modelyje, jau vaizduojami ne komponentai, o juos atitinkantys artefaktai. Parodyta, kad visi klientinės sistemos failai bei klientinės sistemos duomenų bazė yra viename Web serveryje. Su administratoriaus kompiuteriu serveris bendrauja HTTP, o su centrine sistema – SOAP [1, 2] protokolais.



82 pav. Klientinės sistemos įdiegimo modelis

6.3. Sistemų autentifikavimas

Saugiam duomenų perdavimui tarp centrinės ir klientinės sistemų būtinas užklausos siuntėjo autentifikavimas, t. y. būtina nustatyti, ar užklausą siunčianti sistema yra atpažįstama užklausą gavusios sistemos. Tokiu būdu yra išvengiama nesankcionuotų bandymų atsisiųsti kodo fragmentą iš centrinės sistemos ar nusiųsti netikrą pranešimą klientinei sistemai.

Autentifikavimą pasirinkta vykdyti pagal panašius principus į IB Pay (atsiskaitymo e-bankuose) sistemas [8], kad saugumas būtų kuo didesnis (maksimaliam saugumui užtikrinti paprastai duomenų transportavimui reikėtų naudoti HTTPS protokolą, tačiau šiuo atveju tai visiškai nėra būtina – 6.4 posk.). Kiekviena sistema (tiek centrinė, tiek klientinė) turi sugeneruotą raktų porą – privatų bei viešą. Privatus raktas turi būti jokiais būdais neprieinamas iš sistemos išorės. Viešas raktas gali būti laisvai platinamas. Šiame darbe sukurtame modelyje vieši raktai įrašomi tvarkant centrinių ar klientinių sistemų sąrašus. Klientinės sistemos turi turėti centrinių sistemų, kuriose saugomi ir atnaujinami jų naudojami kodo fragmentai, viešus raktus, o centrinės – klientinių sistemų, kurių naudojamus fragmentus saugo.

Siunčiamoje užklausoje vienas iš parametrų yra skaitmeninis parašas. Jis gaunamas savo privačiu raktu pasirašant teksto eilutę, sudarytą iš likusių parametrų pagal griežtą algoritmą (žr. žemiau). Pagal tą patį algoritmą iš gautų parametrų sudaroma teksto eilutė užklausą gavusioje sistemoje. Galiausiai remiantis šia eilute, gautu parašu bei užklausos siuntėjo viešu raktu nustatoma, ar siuntėjas yra atpažintas.

Pasirašoma tekstinė eilutė Y sudaroma sujungiant teksto blokus X_i, gautus iš kiekvieno į eilutę įtraukiamo parametro. Teksto blokas X_i gaunamas pagal tokią formulę:

$$X_i = p(x_i)||x_i, \qquad \text{(1 formulė)}$$

čia x_i – i-ojo parametro tekstinė išraiška, $p(x_i)$ – to parametro ilgio triženklė išraiška, pridedant priekyje nulius, jei ilgio skaitinė vertė mažesnė už 100. Ženklas „||“ reiškia teksto bloko prijungimą. Taigi visa pasirašoma eilutė, kai parametrų kiekis lygus n, sudaroma taip:

$$Y = X_1||X_2||\ldots||X_n = p(x_1)||x_1||p(x_2)||x_2\ ||\ldots||p(x_n)||x_n. \qquad \text{(2 formulė)}$$

Tiek centrinė, tiek klientinė sistema pasirašomas eilutes formuoja iš trijų parametrų: fragmento identifikatoriaus (*fid*), siunčiančios sistemos adreso (*adresas*) bei išsiuntimo laiko (*laikas*). Tarkime, fid = 123, adresas = „http://www.xyz.lt/ws_engine.php“, laikas = 1198368000 (sekundžių kiekis nuo 1970-ųjų metų). Tuomet p(fid) = „003”, p(adresas) = „031”, p(laikas) = „010”, o visa eilutė Y = „003123031http://www.xyz.lt/ws_engine.php0101198368000“.

Iš suformuotos eilutės saugiu maišos algoritmu SHA-1 gaunama eilutės santrauka. Ji pasirašoma privačiu raktu naudojant RSA algoritmą.

Toks autentifikavimas vykdomas ne tik užklausose. Lygiai tokiu pačiu principu yra autentifikuojama centrinė sistema, atsakanti į klientinės sistemos užklausą. Tai daroma tam, kad būtų išvengta nesankcionuotų grąžinamų kodo fragmentų pakeitimų tarpiniuose mazguose tarp centrinės ir klientinės sistemų. Taip užtikrinamas visapusiškas perduodamų duomenų saugumas.

6.4. Perduodamų duomenų šifravimas

Kad perduodant duomenis iš vienos sistemos į kitą jie nebūtų nesankcionuotai nuskaitomi tarpiniuose mazguose, paprastai transportavimui naudojamas saugus protokolas HTTPS. Tačiau šiame darbe tai nebūtina, kadangi svarbiausi duomenys, t. y. programinio kodo fragmentai prieš siunčiant yra užšifruojami. Naudojamos tos pačios privačių ir viešų raktų poros, tiktai priešinga tvarka – užšifruojama viešu raktu, o dešifruojama privačiu.

RSA algoritmu negalima užšifruoti ilgesnių teksto eilučių nei pats privatus raktas (šiame darbe naudojami 1024 bitų raktai), o kodo fragmentai gali būti gerokai didesnės apimties. Todėl daromas tarpinis veiksmas – sugeneruojamas vienkartinis slaptas raktas, o kodo fragmentas užšifruojamas šiuo sugeneruotu raktu, naudojant paprastesnį šifravimo algoritmą (RC4). Tuomet sugeneruotasis raktas užšifruojamas gavėjo viešuoju raktu.

Gavėjui perduodamas ir užšifruotas kodo fragmentas, ir vienkartinis slaptas raktas. Pirmiausiai privačiu raktu dešifruojamas vienkartinis slaptas raktas. Tuomet juo dešifruojamas gautas kodo fragmentas.

Tekstas, užšifravus jį gavėjo viešuoju raktu, tampa niekam nesuprantamas ir bereikšmis, išskyrus gavėją. Toks mechanizmas užtikrina, kad niekas kitas, išskyrus konkrečią klientinę sistemą, negalės tinkamai interpretuoti siunčiamo kodo fragmento.

6.5. Transakcijų valdymas

Kodo fragmento atnaujinimas klientinėje sistemoje vykdomas transakcijos principu. Jei bent vienas sudedamasis atnaujinimo proceso veiksmas neatliekamas arba atliekamas nesėkmingai, fragmentas neatnaujinamas.

Atnaujinimo procesą sudaro tokie sudedamieji veiksmai:

1) gaunamas centrinės sistemos atsakymas į atnaujinimo užklausą;
2) tikrinama, ar teisinga gauta duomenų struktūra;
3) autentifikuojamas atsakymo siuntėjas, t. y. centrinė sistema;
4) tikrinama, ar atsakyme nėra klaidos pranešimo;
5) dešifruojamas atsiųstas kodo fragmentas;
6) fragmentas išsaugomas tekstiniame faile;
7) sukuriamas naujos versijos įrašas duomenų bazėje su nuoroda į sukurtą failą;
8) pakeičiama fragmento įrašo duomenų bazėje nuoroda į naujausios versijos įrašą.

Paskutinis veiksmas negali būti vykdomas, jei neatlikti visi kiti, arba jei bent vieno iš jų atsakymas yra neigiamas (pavyzdžiui, neteisinga gauta duomenų struktūra). Tokiu būdu užtikrinama, kad esant tinklo sutrikimams, nesankcionuotiems bandymams pakeisti siunčiamus

duomenis tarpiniuose mazguose arba sutrikimams pačiuose serveriuose, klientinės sistemos darbas nebus sutrikdytas.

6.6. Sistemų testavimo planas ir rezultatai

Realizuoto centrinės bei klientinės sistemų prototipo testavimo metu siekiama išsiaiškinti, ar yra tenkinami iškelti funkciniai reikalavimai, kad būtų galima patvirtinti, jog užtikrintas teisingas tiek centrinės, tiek klientinės sistemų veikimas. Kadangi šios sistemos veikia ne tik atskirai, bet ir sąveikauja tarpusavyje, būtina testavimo metu atskleisti esmines klaidas bei jas pašalinti, kad bendravimas tarp sistemų vyktų nepriekaištingai bei nesutrikdytų klientinės sistemos darbo. Be to, būtina ištestuoti ir kodo formavimą užsakovo interneto svetainėje pagal šiame darbe sukurtą modelį. Testavimą galima suskirstyti į tokius etapus:

- centrinės sistemos panaudojimo atvejų realizacijos bei komponentų testavimas;
- klientinės sistemos panaudojimo atvejų realizacijos bei komponentų testavimas;
- centrinės bei klientinės sistemų sąveikos, atnaujinimų vykdymo testavimas;
- centrinės bei klientinės sistemų autentifikavimo testavimas;
- sąveikos tarp daugiau nei dviejų sistemų testavimas;
- kodo formavimo užsakovo interneto svetainėje testavimas.

Testavimas buvo vykdomas visos realizacijos metu, kadangi pašalinti esmines klaidas ankstesnėje realizacijos stadijoje yra gerokai paprasčiau nei tai daryti tuomet, kai visa sistema jau realizuota. Kadangi pirmiau buvo realizuojama centrinė sistema, ji ir buvo testuojama pirmoji. Po kiekvieno panaudojimo atvejo įgyvendinimo buvo tikrinama, ar sistemoje įvykdoma būtent tai, ko reikalaujama, ar tinkamai veikia kiekvienas naujas komponentas. Taip pat pridėjus kiekvieno naujo panaudojimo atvejo realizaciją buvo tikrinama, ar jis nepadaro neigiamos įtakos jau anksčiau realizuotiems panaudojimo atvejams. Aptikus klaidų, jos buvo šalinamos, o po to iš naujo testuojama probleminė sistemos dalis. Toks testavimas buvo vykdomas tol, kol buvo realizuoti visi centrinės sistemos panaudojimo atvejai, išskyrus tuos, kurie skirti sąveikai su klientinėmis sistemomis, kadangi dar nebuvo realizuota pati klientinė sistema.

Klientinės sistemos testavimas buvo vykdomas tokiu pat principu, kaip ir centrinės – po kiekvieno panaudojimo atvejo realizavimo buvo tikrinama, ar panaudojimo atvejis įvykdomas teisingai ir kokią įtaką jis daro bendram sistemos veikimui. Taip buvo testuojama iki tol, kol buvo realizuota visa klientinė sistema, išskyrus tuos panaudojimo atvejus, kuriuos vykdant bendraujama su centrine sistema.

Svarbiausias testavimo etapas – patikrinti, ar teisingai sąveikaujama tarp centrinės bei klientinės sistemų, ar teisingai įvykdomas kodo fragmento atnaujinimas klientinėje sistemoje. Turi

būti užtikrinta, kad atnaujinimas vyktų transakcijos principu (6.5 posk.). Tik tokiu atveju galima teigti, kad sistemų sąveikavimas veikia tinkamai. Kad būtų nustatyta, ar programinio kodo fragmentai tinkamai išsaugomi klientinėje sistemoje, testavimo metu buvo skaičiuojamos fragmentų santrumpų (angl. hash) eilutės prieš išsiunčiant fragmentą iš centrinės sistemos ir po jo išsaugojimo klientinėje sistemoje. Jas sulyginus buvo galima įvertinti, ar išsaugojimas įvykdomas tinkamai. Taip pat buvo bandoma dirbtinai sukelti situacijų, kurių metu transakcija neįvyktų (uždraustas failų įkėlimas serveryje, pakeisti duomenų bazės lentelės laukų pavadinimai). Kaip ir numatyta, klientinė sistema tokiu atveju nesutriko, o klaidos buvo užfiksuotos.

Saugumo dėlei buvo svarbu įsitikinti, ar teisingai veikia sistemų autentifikacija (6.3 posk.). Dėl to testavimo metu buvo bandoma išsiųsti pranešimus klientinei sistemai bei užklausas centrinei sistemai su neteisingai suformuotu parašu. Buvo įsitikinta, kad tokiu atveju užfiksuojama klaida. Tai reiškia, kad autentifikavimas veikia teisingai.

Galiausiai buvo svarbu įsitikinti, ar lygiai taip pat tinkamai vyksta sąveika ne tarp dviejų, o tarp didesnio kiekio sistemų. Tam buvo įdiegtos dvi centrinės bei penkios klientinės sistemos skirtinguose serveriuose, naudojančios skirtingą kiekį programinio kodo fragmentų. Viena klientinė sistema buvo įdiegta tame pačiame serveryje, kuriame veikė viena iš centrinių sistemų, kad būtų galima įsitikinti, jog dėl to nesutrinka tinklo paslaugų veikimas. Testavimo metu naudoti tokių charakteristikų serveriai:

- Quad-Core Xeon 2,33 GHz procesorius, 10 GB RAM, 72GB HDD (dvi centrinės bei viena klientinė sistema);
- Dual Xeon 3,4 GHz procesorius, 1 GB RAM, 400 GB HDD (dvi klientinės sistemos);
- Intel Xeon 3,4 GHz procesorius, 10 GB RAM, 120 GB HDD (dvi klientinės sistemos).

Buvo įsitikinta, kad visais atvejais, kai dirbtinai nebandoma iššaukti klaidos (nustatant neteisingą serverio adresą ar viešąjį raktą, uždraudžiant įkelti failus serveryje), pranešimų siuntimai klientinėms sistemoms bei fragmentų atnaujinimai jose įvykdomi teisingai. Priešingu atveju užfiksuojama klaida centrinėje arba klientinėje sistemoje. Toks ir buvo numatytas sistemų veikimas.

Be pačios sistemos reikėjo papildomai ištestuoti, ar teisingai įvykdomas kodo formavimas pavyzdiniame užsakovo interneto svetainės puslapyje. Tam vienu atveju buvo generuojamas puslapis įprastu būdu, o kitu atveju pagal šiame darbe sukurtą modelį. Sulyginus HTML puslapių santrumpas buvo prieita išvada, kad kodas formuojamas tinkamai. Kodo formavimo trukmės pokyčių tyrimas aprašytas 7.3 poskyryje.

6.7. Sistemos realizacijos priemonės bei technologijos

Tiek centrinės, tiek klientinės sistemos realizavimui pasirinktas PHP programavimo kalbos bei MySQL duomenų bazės derinys, kadangi modelis skirtas būtent PHP ir panašiomis technologijomis realizuojamoms sistemoms. Norint išvengti nesuderinamumų tarp senesnių bei naujesnių PHP ir MySQL versijų, prototipo realizacijai naudotos palyginti senos šių technologijų versijos: *PHP 4.2.0* bei *MySQL 3.23*.

Tinklo paslaugų užklausų siuntimo bei priėmimo realizavimui panaudota *NuSOAP* klasių biblioteka [6, 11].

Kad būtų galima autentifikuoti užklausų siuntėjus, PHP turi būti įdiegtas su OpenSSL [15] palaikymu. Šio paketo funkcionalumas naudojamas pasirašant pranešimus bei tikrinant parašus, taip pat užšifruojant ir dešifruojant kodo fragmentus.

Modeliavimui naudotas CASE įrankis *Magic Draw UML 11.5*. Programuota naudojant *Macromedia Dreamweaver MX 6.1* programinį paketą.

6.8. Sistemos diegimas

Norint įdiegti sistemos prototipą, reikia turėti bent du Web serverius, palaikančius PHP kalbą (bent 4.2.0 versiją) bei turinčius MySQL duomenų bazes (bent 3.23 versiją). PHP turi būti įdiegtas su OpenSSL palaikymu, kadangi tam tikros šio išplėtimo funkcijos yra būtinos sistemų autentifikavimui. Toliau aprašomas centrinės sistemos diegimas darant prielaidą, kad klientinės sistemos diegimas yra analogiškas.

Centrinės sistemos įdiegimo kataloge yra failas *connect.php*, skirtas prisijungimui prie duomenų bazės. Šio failo duomenis reikia pakeisti į konkretaus serverio duomenis. Atlikus šią modifikaciją, visą įdiegimo katalogo turinį reikia įkelti į Web serverį. Tuomet naudojant interneto naršyklę reikia surinkti tokį adresą: http://XYZ/install/create_tables.php, kur XYZ – serverio adresas (įtraukiant ir kelią iki įkelto įdiegimo katalogo turinio, jei jis įkeltas ne į šakninį serverio katalogą). Jei teisingai įvesti MySQL prisijungimo duomenys, įvedus šį adresą bus sukurtos sistemai reikalingos duomenų bazės lentelės bei įvesti pradiniai duomenys (sukurti sistemos administratoriaus prisijungimo duomenys). Be to, būtina katalogui *fragmentai* suteikti tokias priėjimo teises, kad jame būtų leidžiama įrašyti failus. Taip pat serveryje turi būti galimybė keisti didžiausią leistiną kodo vykdymo laiką (konkrečios trukmės išmatuojamos atliekant eksperimentą – 7 sk.). Centrinė sistema tokiu atveju tampa paruošta naudojimui. Analogiškai įdiegiama klientinė sistema. Įdiegto centrinės bei klientinės sistemų prototipo vartotojo sąsajos pavyzdžiai pateikti 10 skyriuje (2 priedas).

7. EKSPERIMENTINIS PASLAUGŲ SISTEMOS TYRIMAS

Šiame skyriuje aprašoma pagal sukurtą prototipą realizuotų pavyzdinių sistemų eksperimentinio tyrimo eiga bei rezultatai. Tyrime išskirtos trys atskiros dalys: pranešimų siuntimo trukmės, atnaujinimų vykdymo trukmės bei programinio kodo formavimo trukmės pokyčių tyrimai.

7.1. Pranešimų siuntimo klientinėms sistemoms trukmės tyrimas

Šio etapo tikslas – nustatyti atnaujinimo pranešimų siuntimo iš centrinės į klientinę sistemą trukmės priklausomybę nuo pranešimų kiekio. Ši priklausomybė reikalinga tam, kad, numatant didžiausią galimą klientinių sistemų, naudojančių vieną fragmentą, skaičių (t. y. didžiausią pranešimų apie vieną atnaujinimą gavėjų skaičių), būtų galima apskaičiuoti, koks pranešimų siuntimo kodo vykdymo laikas turi būti nustatytas centrinėje sistemoje. Pranešimo siuntimo trukme čia vadinamas laiko tarpas nuo pranešimo formavimo pradžios iki atsakymo gavimo iš klientinės sistemos.

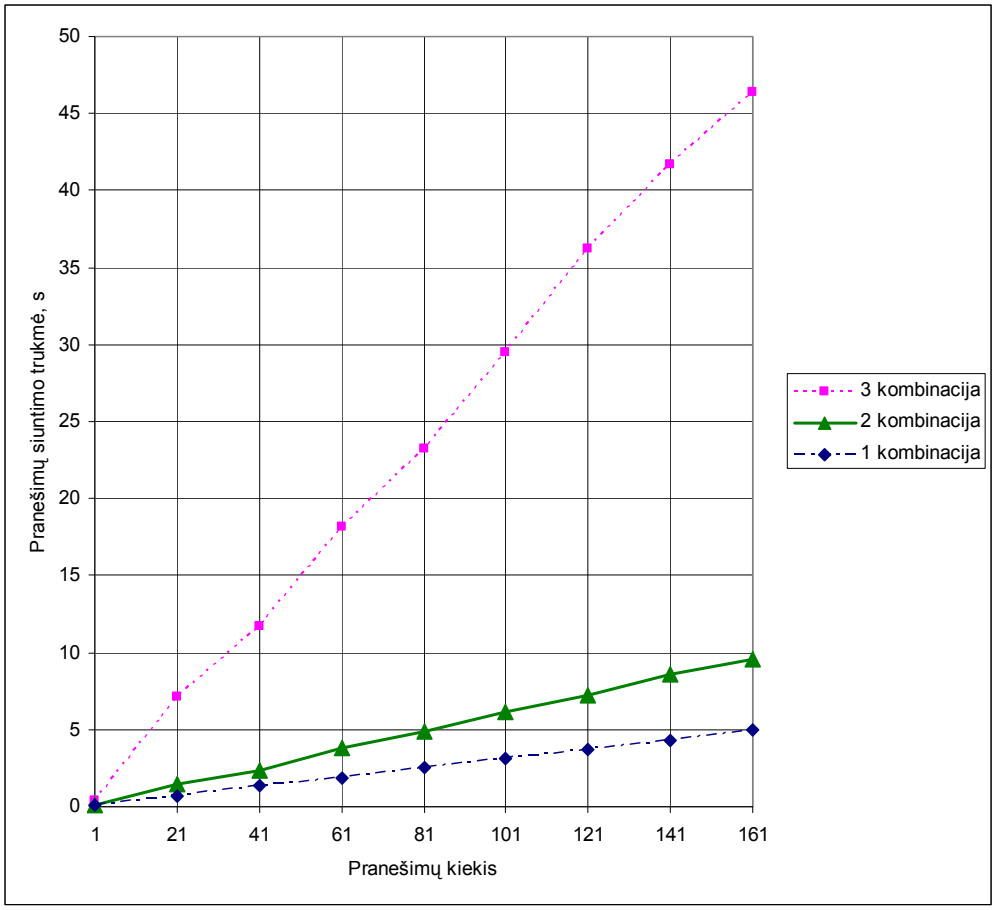
Priklausomybei nuo sistemų kiekio nustatyti reikia įtraukti labai daug klientinių sistemų, todėl toks tyrimas būtų nepatogus. Vietoje to pasirinkta keletas skirtingų centrinės bei klientinės sistemų serverių kombinacijų, o pranešimai vieno eksperimento metu buvo siunčiami vienai sistemai skirtingą kiekį kartų. Toks sprendimas ne tik neiškreipia rezultatų, bet ir padeda nustatyti, kokie gali būti pranešimų siuntimo trukmės kitimo intervalai.

Kiekvienas trukmės įvertis buvo nustatomas atliekant matavimus nuo 3 iki 30 kartų bei apskaičiuojant vidurkį arba medianą tam, kad rezultatai būtų mažiau paveikti atsitiktinių tinklo apkrovimų bei procesų, vykdomų tiek centrinės, tiek klientinės sistemų serveriuose. 83 pav. pateikta pranešimų siuntimo trukmės priklausomybė, esant skirtingoms centrinės bei klientinės sistemų serverių kombinacijoms:

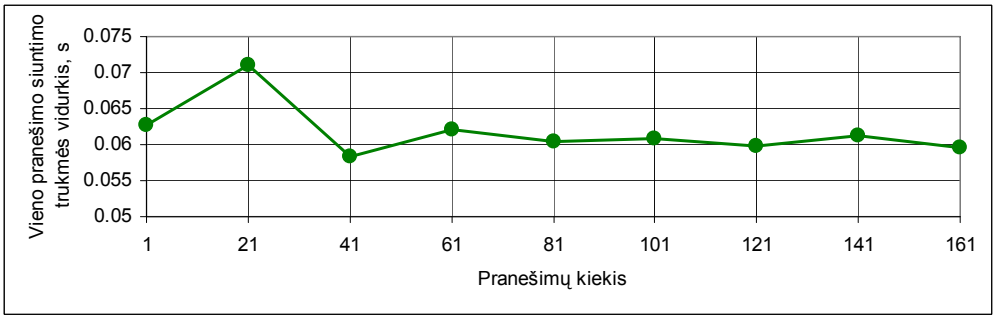
- 1 kombinacija – centrinės sistemos serveris papildomai neapkrautas, o klientinės sistemos serveris sparčiausias iš visų, kuriuose eksperimentuota;
- 2 kombinacija – centrinės sistemos serveris papildomai neapkrautas, o klientinės sistemos serveris turi beveik dvigubai mažesnį pralaidumą nei pirmu atveju;
- 3 kombinacija – centrinės sistemos serveris papildomai apkrautas (jame paleista 11 papildomų fiktyvių procesų), o klientinės sistemos serveris tas pats, kaip antru atveju.

Grafike matoma, kad papildomai apkrovus centrinės sistemos serverį pranešimų siuntimo trukmė ženkliai išauga. Mažiausia trukmė, kaip ir buvo galima nuspėti, išmatuota siunčiant pranešimus į didžiausią pralaidumą turintį serverį. Tačiau realiausias atvejis yra antroji kombinacija, kai klientinės sistemos serveris nėra toks spartus kaip pirmu atveju, bet centrinės sistemos serveris nėra papildomai apkrautas. Trečioji kombinacija gali būti reali tik tokiu atveju,

jeigu centrinė sistema patalpinta labai ribotos spartos virtualiame serveryje. Tačiau darant prielaidą, kad centrinei sistemai skiriamas spartus virtualus arba dedikuotas serveris, toliau pateikiami tik su antrąja kombinacija susiję matavimai. Taip pat verta paminėti, kad papildomu darbu apkrovus klientinės sistemos serverį pranešimų siuntimo trukmė praktiškai beveik nepakito. Dėl šios priežasties grafike tokios serverių kombinacijos nepateikiamos.



83 pav. Pranešimų siuntimo, esant skirtingoms serverių kombinacijoms, trukmės priklausomybės nuo pranešimų kiekio palyginimas



84 pav. Vieno pranešimo siuntimo trukmės priklausomybė nuo pranešimų kiekio

Pranešimų siuntimo trukmės grafike (83 pav.) matoma, kad priklausomybė yra labai artima tiesinei. Kad tai būtų tiesa, reikia, jog vieno pranešimo siuntimo trukmė būtų pastovi, t. y. nepriklausoma nuo pranešimų kiekio. 84 pav. pateiktas vieno pranešimo siuntimo trukmės

priklausomybės nuo pranešimų kiekio grafikas. Esant mažam pranešimų kiekiui bendra trukmė yra labiau paveikiama pašalinių procesų, todėl esama svyravimų. Tačiau matoma, kad pranešimo siuntimo trukmė nusistovi arti 0,06 s. Todėl galima teigti, kad pranešimų siuntimo klientinėms sistemoms trukmė tiesiškai priklauso nuo pranešimų kiekio. Šią priklausomybę galima išreikšti tokia formule:

$$t \approx t_0 \cdot N, \qquad \text{(3 formulė)}$$

čia t – bendra trukmė, t_0 – vieno pranešimo siuntimo trukmė (šiuo atveju t_0 apytiksliai lygus 0,06 s, o jeigu taikoma trečioji kombinacija, $t_0 \approx 0{,}3$ s), N – siunčiamų pranešimų kiekis.

Kad būtų galima numatyti, kiek gali kisti vieno pranešimo siuntimo trukmė, papildomai surastas šios trukmės patikimumo intervalas, kai N lygus didžiausiam tyrimo metu naudotam pranešimų kiekiui, t. y. kai N = 161. Priimta, kad pakankamas patikimumo lygis yra 95 %. Kadangi tikrasis trukmės vidurkis nėra žinomas (kad jį būtų galima surasti, atliktų bandymų kiekis turėtų artėti link begalybės), patikimumo intervalo rėžiai apskaičiuoti pagal *Student t* pasiskirstymo dėsnį:

$$\bar{x} \pm t \frac{s}{\sqrt{n}}, \qquad \text{(4 formulė)}$$

čia $\bar{x}$ - išmatuotų trukmės įverčių vidurkis, t – įvertis iš *Student* pasiskirstymo lentelės, priklausomas nuo pasirinkto patikimumo lygio bei matavimų kiekio, s – standartinis trukmės nuokrypis, n – matavimų kiekis. Gautos tokios šių parametrų reikšmės:

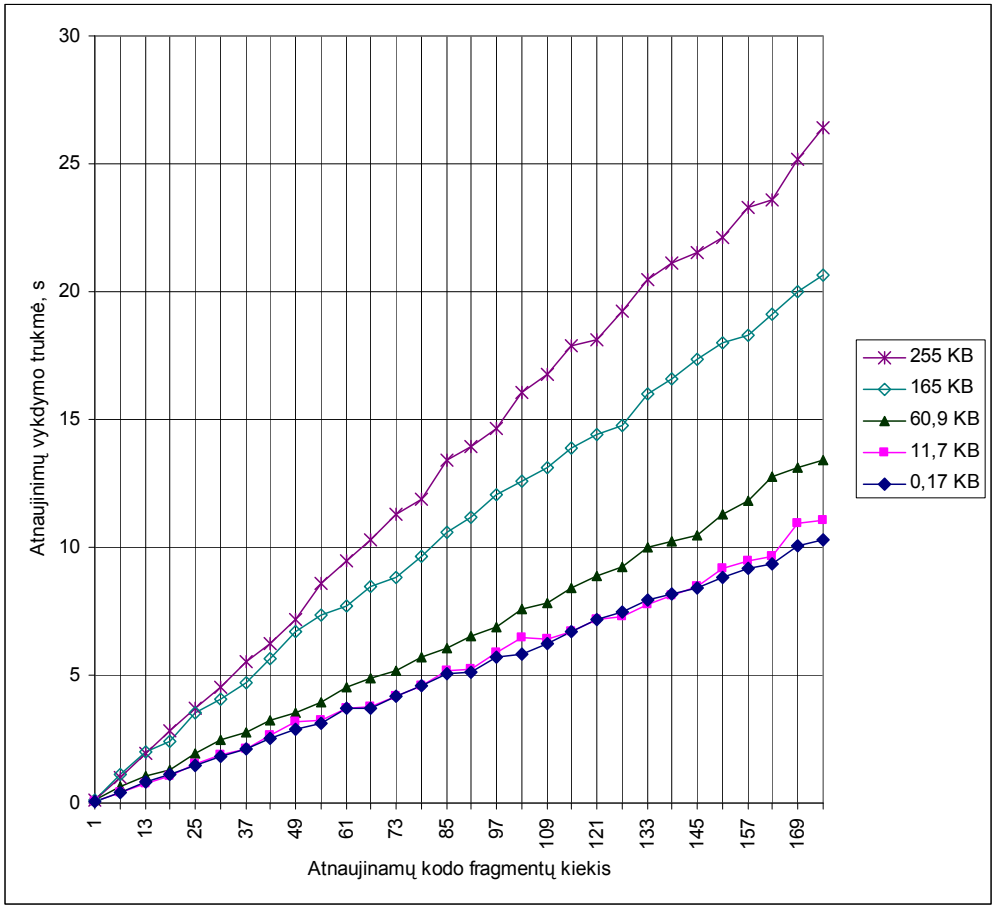
- n = 161 (sutampa su N);
- $\bar{x} \approx 0{,}059452$ s;
- t = 1,645 (95 % patikimumo lygiui bei 161 matavimui);
- s ≈ 0,005586 s.

Patikimumo intervalas šiuo atveju toks: [0,058728; 0,060176]. Tai reiškia, kad su 95 % tikimybe vieno pranešimo siuntimas užtruks nuo 0,058728 s iki 0,060176 s.

Taigi numatant didžiausią galimą klientinių sistemų, naudojančių vieną fragmentą, kiekį N bei pridedant papildomą laiką t_{papild}, reikalingą įkelto kodo fragmento išsaugojimui centrinėje sistemoje, galima apskaičiuoti, koks turi būti nustatytas leistinas pranešimų siuntimo kodo vykdymo laikas t. Išmatavus t_{papild} paaiškėjo, kad šis įvertis lygus vos kelioms šimtosioms sekundės dalims net ir išsaugant 256 KB apimties kodo fragmentą. Taigi praktiškai kodo vykdymo laikui apskaičiuoti galima taikyti 3 formulę.

7.2. Atnaujinimų vykdymo klientinėse sistemose trukmės tyrimas

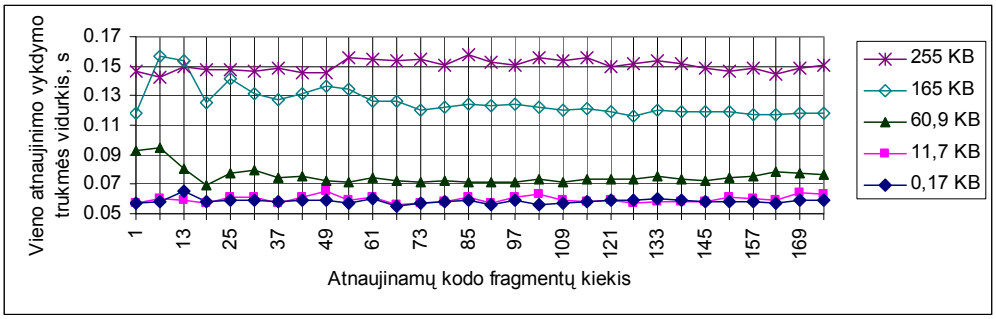
Šiame eksperimentinio tyrimo etape siekta nustatyti atnaujinimų vykdymo klientinėje sistemoje trukmės priklausomybes nuo atnaujinamų kodo fragmentų kiekio bei dydžio. Šios priklausomybės reikalingos tam, kad, numatant didžiausią galimą vienoje klientinėje sistemoje naudojamų kodo fragmentų skaičių bei didžiausią galimą fragmento dydį, būtų galima apskaičiuoti, koks fragmentų atnaujinimų kodo vykdymo laikas turi būti nustatytas klientinėje sistemoje. Į vieno fragmento atnaujinimo trukmę įeina laiko tarpas, skirtas užklausos centrinei sistemai formavimui, siuntimui, atsakymo gavimui, taip pat atsiųsto kodo fragmento išsaugojimui klientinėje sistemoje bei fragmento versijos pakeitimui.



85 pav. Atnaujinimų vykdymo, esant skirtingiems fragmentų dydžiams, trukmės priklausomybės nuo atnaujinamų fragmentų kiekio palyginimas

Kiekvienas trukmės įvertis buvo nustatomas atliekant matavimus nuo 3 iki 8 kartų bei apskaičiuojant vidurkį arba medianą tam, kad rezultatai būtų mažiau paveikti atsitiktinių tinklo apkrovimų bei procesų, vykdomų tiek centrinės, tiek klientinės sistemų serveriuose. 85 pav. pateiktame grafike parodyta atnaujinimų vykdymo trukmės priklausomybė nuo atnaujinamų kodo fragmentų kiekio, atnaujinant skirtingo dydžio fragmentus. Grafike matoma, kad priklausomybė yra artima tiesinei, kaip ir pranešimų siuntimo atveju. Kad tuo būtų galima įsitikinti, 86 pav. pateikta

vieno atnaujinimo vykdymo trukmės vidurkio priklausomybė nuo atnaujinamų fragmentų kiekio. Matoma, kad didėjant fragmentų kiekiui vidutinė trukmė apytiksliai nusistovi arba svyruoja apie vieną ar kitą reikšmę. Tai reiškia, kad kodo fragmentų kiekis vieno atnaujinimo trukmei įtakos nedaro, t. y. atnaujinimų vykdymo trukmė iš tiesų tiesiškai priklauso nuo atnaujinamų fragmentų kiekio.

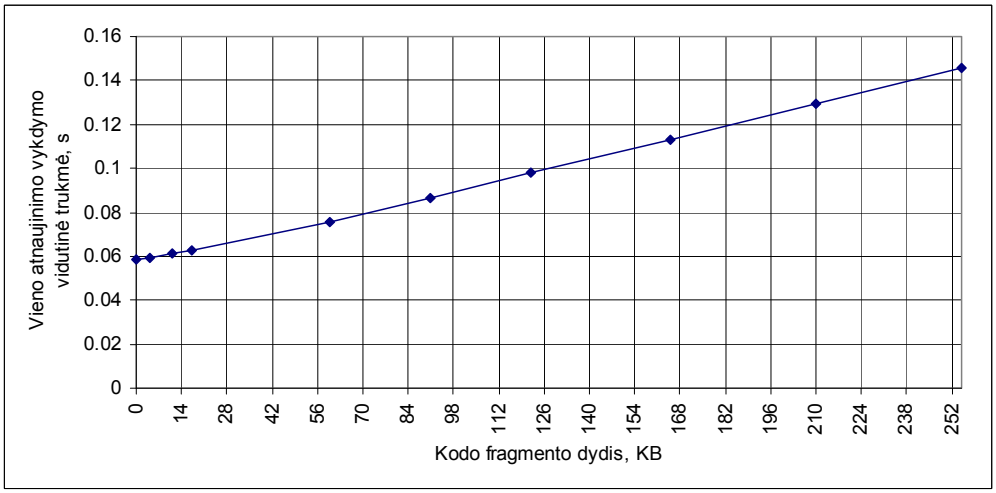


86 pav. Vieno atnaujinimo vykdymo trukmės priklausomybės nuo atnaujinamų fragmentų kiekio palyginimas

Įvertinus šį tiesiškumą, atnaujinimo trukmės priklausomybės nuo atnaujinamo fragmento dydžio nustatymui galima naudoti vidutinius nusistovėjusius trukmės įverčius. Kad būtų galima numatyti, kaip kinta nusistovėjusios reikšmės, buvo rastas 11,7 KB dydžio fragmento siuntimo trukmės įverčio patikimumo intervalas (tokio dydžio fragmentas pasirinktas tik kaip pavyzdys). *Student t* pasiskirstymo dėsnio (4 formulė) parametrų reikšmės tokios:

- n = 175;
- $\bar{x} \approx 0{,}061947$ s;
- t = 1,645 (95 % patikimumo lygiui bei 175 matavimams);
- s ≈ 0,006573 s.

Patikimumo intervalas šiuo atveju toks: [0,061130; 0,062765]. Tai reiškia, kad su 95 % tikimybe vieno atnaujinimo siuntimas užtruks nuo 0,061130 s iki 0,062765 s.

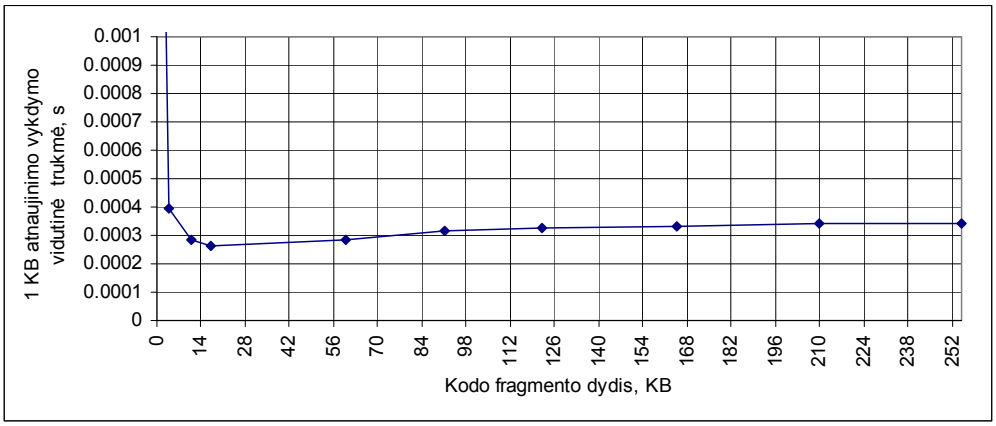


87 pav. Vieno atnaujinimo vykdymo trukmės priklausomybė nuo atnaujinamo fragmento dydžio

Vieno atnaujinimo vykdymo trukmės priklausomybė nuo kodo fragmento dydžio parodyta 87 pav. Matoma, kad vidutinė atnaujinimo trukmė nuo fragmento dydžio priklauso tiesiškai, tačiau net ir atnaujinant labai mažos apimties fragmentus, trukmė neartėja prie 0. Taip yra dėl to, kad atnaujinimo metu tarp centrinės ir klientinės sistemų yra apsikeičiama tinklo paslaugų užklausų antraštėmis bei kitais papildomais duomenimis. Susidaro papildomas pastovus dydis – trukmė t_{papild}, kuri šiuo atveju apytiksliai lygi 0,059 s. Taigi vieno atnaujinimo vykdymo trukmę t_0 galima apskaičiuoti pagal tokią formulę:

$$t_0 \approx t_1 \cdot L + t_{papild}, \qquad \text{(5 formulė)}$$

čia L – atnaujinamo fragmento dydis kilobaitais, t_1 – vieno kilobaito apimties kodo fragmento atnaujinimo trukmė, nesant papildomos trukmės t_{papild}. Dydį t_1 galima rasti iš 88 pav. pateikto grafiko. Šiuo atveju jis nusistovi apytiksliai ties 0,00035 s riba.



88 pav. 1 KB atnaujinimo trukmės priklausomybė nuo atnaujinamo fragmento dydžio, neįvertinant trukmės t_{papild}

Remiantis 3 bei 5 formulėmis, galima išvesti bendros atnaujinimų vykdymo trukmės t apskaičiavimui skirtą formulę:

$$t \approx (t_1 \cdot L + t_{papild}) \cdot N, \qquad \text{(6 formulė)}$$

čia L – atnaujinamo fragmento dydis kilobaitais, t_1 – vieno kilobaito apimties kodo fragmento atnaujinimo trukmė, nesant papildomos trukmės t_{papild}, N – atnaujinamų fragmentų kiekis.

Taigi taikant 6 formulę galima apskaičiuoti didžiausią galimą atnaujinimo kodo vykdymo klientinėje sistemoje trukmę. Tam vietoje N reikia įstatyti didžiausią galimą klientinėje sistemoje naudojamų fragmentų kiekį N_{max}, o vietoje L – didžiausią galimą fragmento dydį L_{max}. Jei L_{max} lygus 256 KB, o N_{max} – 175, tuomet eksperimentiniame serveryje $t \approx 26$ s.

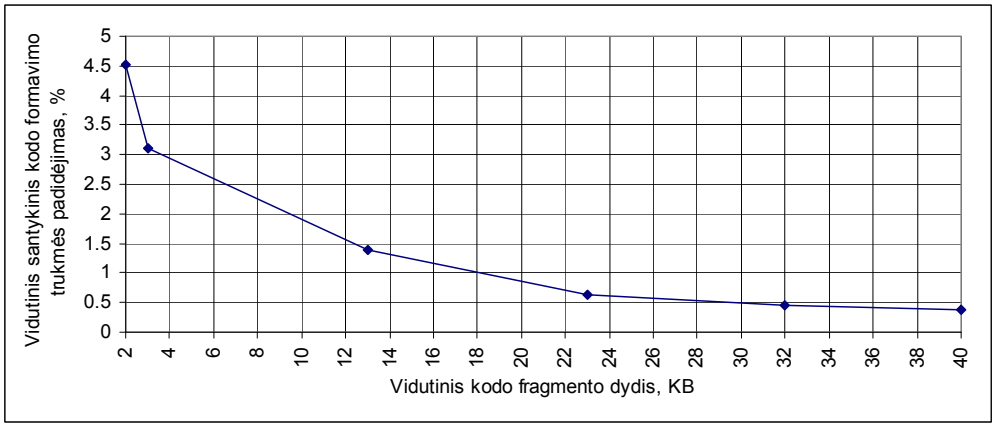
7.3. Programinio kodo formavimo klientinėse sistemose trukmės pokyčių tyrimas

Šiame tyrimo etape siekta nustatyti, ar klientinių sistemų kodą formuojant pagal šiame darbe sukurto modelio principus tenkinamas 3.2 poskyryje iškeltas nefunkcinis greito HTML puslapių formavimo reikalavimas. Jo tinkamumo kriterijus – HTML puslapių formavimo trukmė neturi padidėti daugiau kaip 5 %, lyginant su įprastu kodo formavimu. Kadangi pačiame HTML puslapių formavime skirtumas nuo įprasto formavimo iš esmės yra tik toks, kad vykdomas papildomas kreipinys į duomenų bazę, buvo tikimasi, kad tinkamumo kriterijus bus tenkinamas. Kad būtų galima nustatyti trukmės padidėjimą, realizuotas vienas pavyzdinis puslapio formavimas įprastu būdu, t. y. kodo fragmentus įtraukiant statiškai (76 pav.), o kitas – tokio pat puslapio formavimas remiantis šiame darbe sukurtu modeliu, t. y. prieš įtraukiant fragmentus gaunant iš duomenų bazės naujausių versijų failų vardus (77 pav.).

Siekta nustatyti pokyčių priklausomybes nuo fragmentų dydžio bei kiekio, kad būtų galima numatyti ribas, iki kurių tinkamumo kriterijus tenkinamas. Iškeltos dvi hipotezės:

1) kuo didesni kodo fragmentai, tuo santykinis kodo formavimo trukmės padidėjimas mažesnis;
2) kuo daugiau kodo fragmentų, tuo santykinis kodo formavimo trukmės padidėjimas mažesnis.

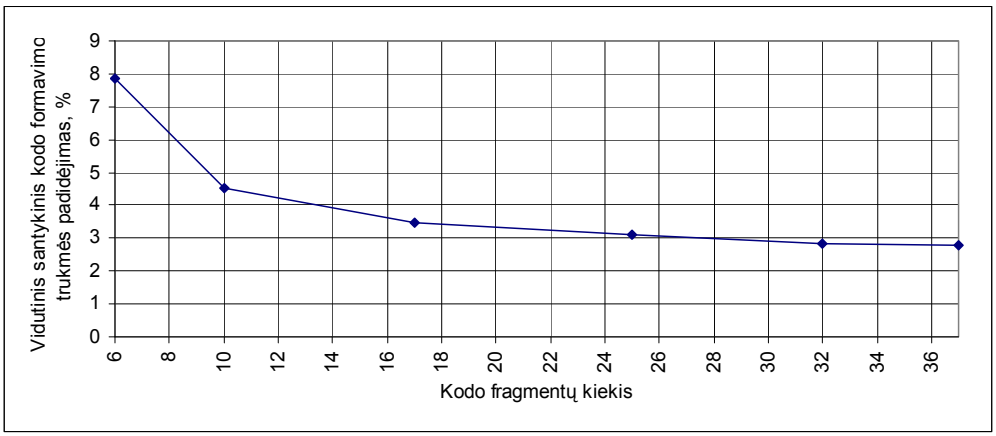
Pirmoji hipotezė logiškai kyla iš to, kad didesnių fragmentų įtraukimo trukmė yra didesnė nei mažesnių, todėl ji užima santykinai didesnę dalį nei duomenų bazės užklausos, kurios ir lemia matuojamą trukmės padidėjimą. Antrosios hipotezės atsiradimo priežastis yra tame, kad norint gauti naujausių kodo fragmentų versijų failų vardus stengiamasi į duomenų bazę kreiptis tik vieną kartą, todėl didesnis fragmentų kiekis sumažina kiekvieno iš jų failo vardo išgavimui skirtą trukmę.



89 pav. Santykinio kodo formavimo trukmės padidėjimo priklausomybė nuo fragmentų dydžio, kai fragmentų kiekis lygus 25

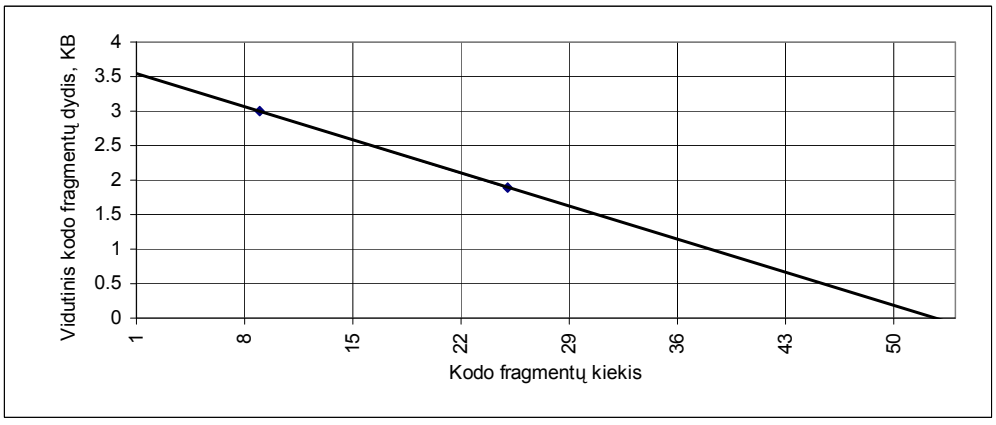
Kad hipotezės būtų priimtos, jas būtina patvirtinti eksperimentais. Pirmajai hipotezei patikrinti atlikta keletas kodo formavimo iš vienodo kiekio, bet skirtingo dydžio fragmentų bandymų. 89 pav. pateiktas šio eksperimento rezultatas, kai fragmentų kiekis lygus 25. Akivaizdu, kad pirmoji hipotezė teisinga, kadangi santykinis trukmės padidėjimas mažėja, didėjant kodo fragmentams.

90 pav. pateiktame grafike parodytas eksperimento, skirto patikrinti antrąją hipotezę, rezultatas. Šiame grafike taip pat akivaizdu, kad antroji hipotezė teisinga, kadangi santykinis trukmės padidėjimas mažėja, didėjant kodo fragmentų kiekiui.



90 pav. Santykinio kodo formavimo trukmės padidėjimo priklausomybė nuo fragmentų kiekio, kai vidutinis fragmentų dydis lygus 3 KB

Galiausiai reikia nustatyti, kokios yra aukščiau minėto tinkamumo kriterijaus tenkinimo ribos. Tam 91 pav. parodyta tiesė, kurios apatinėje dalyje esančios kodo fragmentų kiekio bei vidutinio dydžio kombinacijos tinkamumo kriterijaus netenkins, t. y. santykinis kodo formavimo trukmės padidėjimas bus didesnis nei 5 %. Pavyzdžiui, klientinėje sistemoje naudojant 36 fragmentus, kurių dydžio vidurkis 1 KB, tinkamumo kriterijus nebus tenkinamas, o naudojant tą patį kiekį 1,5 KB dydžio fragmentų kriterijus bus tenkinamas.



91 pav. Greito HTML puslapių formavimo reikalavimo tinkamumo kriterijaus tenkinimo ribos

Papildomai buvo apskaičiuotas kodo formavimo abiem būdais trukmės bei santykinio tos trukmės padidėjimo patikimumo intervalai, kai įtraukiami 25 fragmentai, kurių dydis 2 KB (toks dydis pasirinktas tik kaip pavyzdys). Kodo formavimo pagal šiame darbe sukurtą modelį trukmės *Student t* pasiskirstymo dėsnio (4 formulė) parametrų reikšmės tokios:

- n = 300;
- $\bar{x} \approx 0{,}006356$ s;
- t = 1,645 (95 % patikimumo lygiui bei 300 matavimų);
- s ≈ 0,000806 s.

Patikimumo intervalas toks: [0,006280; 0,006433]. Tai reiškia, kad su 95 % tikimybe kodo fragmentų įtraukimas užtruks nuo 0,006280 s iki 0,006433 s. Kodo formavimo įprastu būdu *Student t* parametrų reikšmės tokios:

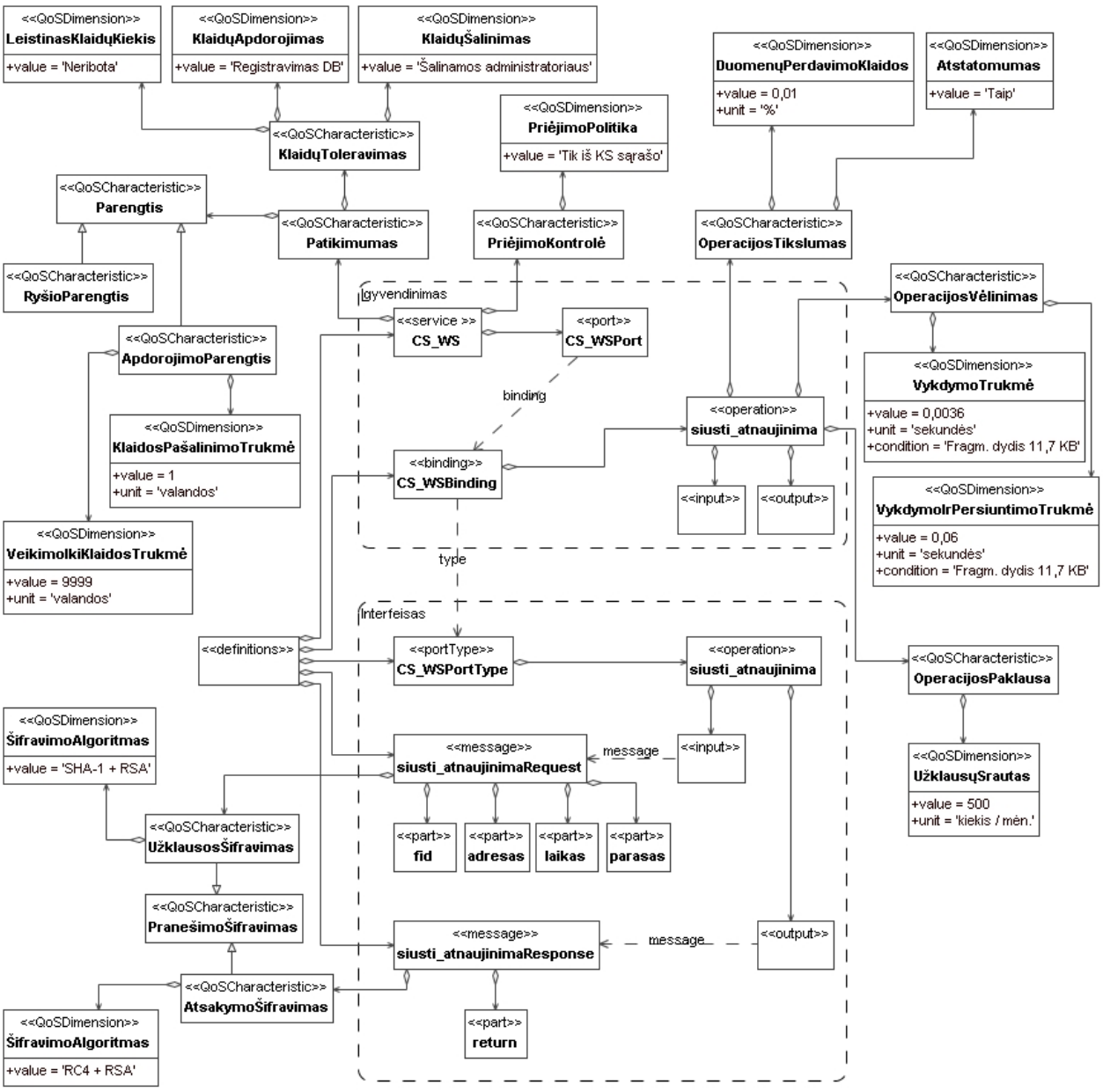
- n = 300;
- $\bar{x} \approx 0{,}006081$ s;
- t = 1,645 (95 % patikimumo lygiui bei 300 matavimų);
- s ≈ 0,000793 s.

Patikimumo intervalas toks: [0,006006; 0,006157]. Tai reiškia, kad su 95 % tikimybe kodo fragmentų įtraukimas užtruks nuo 0,006006 s iki 0,006157 s.

Remiantis šiais dviem intervalais apskaičiuotas trečiasis – kodo formavimo trukmės padidėjimo patikimumo intervalas. Mažesniajam intervalo rėžiui apskaičiuoti paimta pirmojo intervalo mažesnysis bei antrojo intervalo didesnysis rėžiai. Didesniajam atvirkščiai – pirmojo intervalo didesnysis ir antrojo mažesnysis. Gautas toks trukmės padidėjimo patikimumo intervalas: [1,994540; 7,110385]. Tai reiškia, kad su 95 % tikimybe kodo fragmentų įtraukimas pagal šiame darbe sukurtą modelį užtruks nuo 1,994540 % iki 7,110385 % ilgiau, lyginant su įprastu kodo formavimu. Didesnysis rėžis viršija greito kodo formavimo reikalavimo tinkamumo kriterijų, tačiau trukmės padidėjimo vidurkis – ne.

7.4. Nustatytų kokybinių charakteristikų įtraukimas į paslaugų apibrėžimus

Realizavus informacijos sistemų atnaujinimo sistemą bei atlikus eksperimentą galima į 4.3 skyriuje pateiktą išplėsto WSDL aprašą įtraukti realias tinklo paslaugų charakteristikų dimensijų reikšmes. 92 pav. pateikta išplėsto WSDL aprašo su nustatytomis kokybinėmis charakteristikomis schema. Joje kiekviena dimensija turi reikšmę (value). Kiekviena skaitinę reikšmę turinti dimensija turi ir matavimo vienetą (unit).



92 pav. Išplėsta centrinės sistemos tinklo paslaugos aprašo, įtraukiant nustatytas charakteristikas, struktūrinė schema

Operacijos vėlinimą apibrėžiančios dimensijos gautos eksperimento metu. Schemoje pateiktos reikšmės, kurios gautos atnaujinant vidutinės 11,7 KB apimties kodo fragmentus. Esant kitokiems fragmentų dydžiams keisis ir operacijos vykdymo bei vykdymo su persiuntimu trukmė.

Užklausų srautas apskaičiuotas darant prielaidą, kad vidutiniškai kiekvienas fragmentas yra atnaujinamas kas 5 mėnesius. Be to, priimta, kad yra 100 klientinių sistemų, kurių kiekviena naudoja apytiksliai 20 fragmentų. Tokiu būdu gaunama, kad per 5 mėn. gaunama 2000 užklausų arba 400 užklausų per 1 mėn. Taigi galima teigti, kad paslauga nebus nuolat apkrauta darbu, o tai padidina stabilios paslaugos spartos tikimybę.

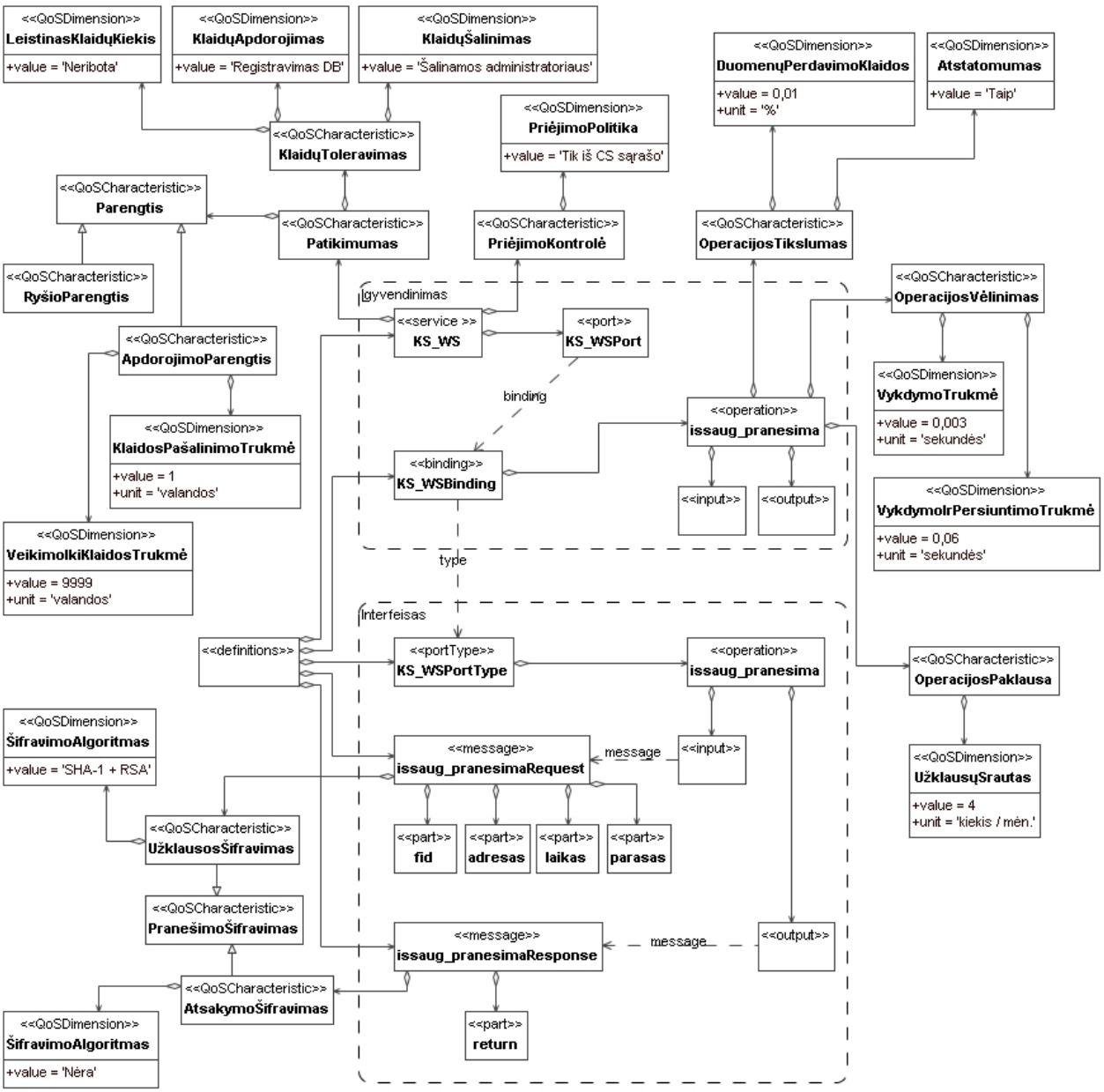
Atliekant keletą tūkstančių bandymų kiekvieną kartą buvo tikrinama, ar atsiųsto fragmento turinys sutampa su išsiųsto fragmento turiniu. Tam buvo lyginamos kodo santraukos. Visais atvejais

jos sutapdavo, bet vis tiek bent minimalią neteisingo duomenų perdavimo tikimybę reikia palikti. Todėl duomenų perdavimo klaidų tikimybės dimensijos reikšmė parinkta 0,01 %. Tačiau neteisingai perdavus duomenis jokie pakeitimai sistemoje neatliekami, taigi galioja savaiminis atstatomumas.

Klaidų toleravimas centrinėje sistemoje yra didelis, kadangi bet kokiai klaidai įvykus jokie pakeitimai nėra atliekami. Taigi sistemos darbas nesutrikdomas. Visos klaidos yra užregistruojamos, o vėliau sistemos administratorius gali jas pašalinti. Nuo klaidų priklauso ir paslaugos apdorojimo (vykdymo) parengtis. Kai sistema tinkamai sukonfigūruota (teisingai suformuotas klientinių sistemų sąrašas, jų adresai bei vieši raktai, joms priskirti naudojami kodo fragmentai), nėra kliūčių netinkamai įvykdyti užklausą. Dėl to veikimo iki klaidos trukmė parinkta 9999 val., o klaidos pašalinimo (konfigūracijos pakeitimo) trukmė – 1 val. (tokios reikšmės parinktos tam, kad paslaugos prieinamumas būtų lygus populiariai 99,99 % vertei). Reikia paminėti, kad bendra parengtis taip pat priklauso ir nuo ryšio parengties, todėl bendras prieinamumas gali būti mažesnis nei 99,99 %. Tačiau ryšio parengtis priklauso nuo konkrečių serverių veikimo stabilumo, todėl jos dimensijos nepateiktos.

Analogiška schema sudaryta ir išplėstam klientinės sistemos paslaugos aprašui (93 pav.). Joje užklausų srautas lygus 4 užklausoms per 1 mėn. Toks kiekis gautas remiantis tomis pačiomis prielaidomis, kaip ir centrinės sistemos užklausų srauto skaičiavime.

Taigi turint tokį kokybinėmis charakteristikomis išplėstą bei užpildytą modelį galima jį naudoti dviem tikslais. Standartinę dalį galima naudoti realizuojant tinklo paslaugas. Išplėstinė dalis naudinga derantis su užsakovu, nes galima iš anksto sužinoti ne tik tai, ką paslauga darys, bet ir tai, kokia jos atliekamo darbo kokybė.



93 pav. Išplėsta klientinės sistemos tinklo paslaugos aprašo, įtraukiant nustatytas charakteristikas, struktūrinė schema

8. DARBO APIBENDRINIMAS

Šiame skyriuje apibendrinami darbo rezultatai, įvertinamas sukurtos sistemos atitikimas paslaugų projektavimo principams bei pateikiamos darbo išvados.

8.1. Darbo rezultatai

Šiame darbe atskleista užsakovų informacijos sistemų atnaujinimo problema, išanalizuota organizacijos veikla ir standartinė paslaugų architektūra bei nuspręsta, kad atnaujinimo problemai spręsti reikalinga centralizuota architektūra. Užsibrėžta sukurti centralizuoto atnaujinimo modelį bei programinį prototipą, patogų galutiniam vartotojui bei paremtą kokybiškomis tinklo paslaugomis. Realizacija tinklo paslaugomis pasirinkta dėl jų standartizuotumo bei išplečiamumo.

Nustatyti funkciniai reikalavimai centrinei bei klientinei sistemoms panaudojimo atvejų pavidalu. Aukščiausiame lygmenyje numatyta, kaip kiekvienas panaudojimo atvejis turi būti vykdomas. Taip pat nustatyti ir nefunkciniai sistemų reikalavimai.

Išnagrinėti 8 pagrindiniai paslaugų projektavimo principai: *standartizuotas paslaugos kontraktas*, *laisvas paslaugos susiejimas*, *paslaugos abstrakcija*, *pakartotinis paslaugos panaudojamumas*, *paslaugos autonomija*, *paslaugos būsenų neturėjimas*, *paslaugos atrandamumas*, *paslaugos komponuojamumas*. 8.2 poskyryje aprašytas sukurtos sistemos atitikimas šiems principams. Taip pat sukurtas centrinės bei klientinės sistemų tinklo paslaugų modelis bei jo išplėtimas kokybinėmis charakteristikomis, leidžiantis ne tik aprašyti paslaugų sąsają, bet ir įvertinti jų kokybę.

Remiantis dauguma paslaugų projektavimo principų bei laikantis nustatytų reikalavimų suprojektuotas išsamus sąveikaujančių sistemų modelis. Aprašyta sistemų loginė architektūra, sukurti klasių, duomenų bazės, vartotojo sąsajos bei elgsenos modeliai. Taip pat aprašyta informacijos sistemų puslapius formuojančių probleminių kodo fragmentų realizavimo atskiruose failuose idėja.

Pagal centralizuotos sistemos modelį realizuotas programinis prototipas bei jį atitinkantys 2 centrinių bei 5 klientinių sistemų pavyzdžiai. Visos realizacijos metu bei po jos buvo testuojamas kiekvienas panaudojimo atvejis, kad klaidos būtų aptiktos kuo ankstesnėje stadijoje.

Atliktas eksperimentinis realizuotų pavyzdžių tyrimas, kurio metu išmatuoti 2 operacijų vėlinimo kokybinių charakteristikų įverčiai: pranešimų siuntimo klientinėms sistemoms trukmė bei atnaujinimų vykdymo klientinėse sistemose trukmė. Taip pat eksperimento metu nustatyta, kiek padidėja puslapių formavimo trukmė informacijos sistemose, priklausomai nuo naudojamų probleminių kodo fragmentų kiekio bei dydžio. Galiausiai nustatytos kokybinės charakteristikos įtrauktos į išplėstą paslaugų aprašo modelį.

Darbo rezultatas – remiantis paslaugų projektavimo principais bei išplečiant tinklo paslaugų aprašą kokybinėmis charakteristikomis sukurtas centralizuoto užsakovų informacijos sistemų atnaujinimo paslaugų modelis bei programinis prototipas, kuris yra patogus galutiniam vartotojui, leidžia sumažinti atnaujinimo laiko sąnaudas bei nepertraukia klientinių sistemų darbo.

8.2. Sukurtos sistemos atitikimo paslaugų kūrimo principams įvertinimas

Šiame poskyryje pateikiamas sukurtų tinklo paslaugų atitikimo 4.1 poskyryje išvardintiems paslaugų projektavimo principams įvertinimas. Trumpai įvertinamas kiekvieno principo laikymasis arba nesilaikymas, o galiausiai pateikiamas apibendrinimas.

Standartizuotas paslaugos kontraktas (Standardized Service Contract). Šiame darbe šio principo yra laikomasi, kadangi tiek centrinės, tiek klientinės sistemų tinklo paslaugos aprašomos WSDL kalba, kuri yra laikoma standartu. Taigi galima formuoti tikslias užklausas, žinant, kokių atsakymų reikia laukti.

Laisvas paslaugos susiejimas (Service Loose Coupling). Šiame darbe paslaugų aprašymai nėra priklausomi nei nuo realizacijos, nei nuo kitų komponentų. Be to, paslaugos naudotojas (centrinės sistemos paslaugos atveju – klientinė sistema, ir atvirkščiai) formuoja užklausas būtent pagal paslaugos aprašymą. Taigi galima teigti, kad ir šio principo šiame darbe yra laikomasi.

Paslaugos abstrakcija (Service Abstraction). Šiame darbe paslaugos aprašomos WSDL dokumentais. Juose pateikiama informacija tiktai apie tai, kokios operacijos su kokiais parametrais gali būti kviečiamos bei kaip pasiekti pačią tinklo paslaugą. Jokia papildoma nereikalinga informacija neatskleidžiama, taigi šio principo irgi laikomasi.

Pakartotinis paslaugos panaudojamumas (Service Reusability). Šiame darbe naudojamas vienas iš šio principo atskirų atvejų – *taktinis pakartotinis panaudojamumas (Tactical Reusability)*. Jo esmė tokia, kad paslauga kuriama vienam tikslui, tačiau yra numatyta galimybė ją išplėsti, kad ateityje atsiradus naujiems tikslams būtų galima pridėti papildomo funkcionalumo.

Paslaugos autonomija (Service Autonomy). Šiame darbe suprojektuotos tinklo paslaugos vykdymo metu nepriklauso nuo jokių išorinių komponentų, išskyrus duomenų bazę. Tačiau duomenų bazė tiek centrinėje, tiek klientinėje sistemose yra naudojama išskirtinai tik šiame darbe naudojamiems konceptams saugoti, todėl išoriniai veiksniai jos spartai papildomos įtakos nedaro. Taigi galima teigti, kad yra laikomasi *vykdymo autonomijos* principo. Taip pat laikomasi ir *projektavimo autonomijos* principo, kadangi yra galimybė pakeisti visą paslaugos realizaciją nepakeičiant jos kontrakto.

Paslaugos būsenų neturėjimas (Service Statelessness). Kadangi šiame darbe suprojektuotos paslaugos skirtos žinomam uždaram centrinių bei klientinių sistemų ratui, neišvengiamai būtina

saugoti bent minimalią informaciją vienoms apie kitas. Tokia informacija – autentifikavimui skirti viešieji raktai bei tinklo adresai. Centrinės sistemos užklausa palieka žymę klientinėsje sistemoje – joje išsaugomas pranešimas. Klientinės sistemos užklausa taip pat gali palikti pėdsaką centrinėje sistemoje – jei atnaujinimo siuntimo metu įvyksta klaida, ji išsaugoma centrinės sistemos duomenų bazėje. Taigi būsenos egzistuoja, tačiau jos saugomos duomenų bazėje, o ne pagrindinėje atmintinėje, todėl galima teigti, kad būsenų neturėjimo principo laikomasi dalinai.

Paslaugos atrandamumas (Service Discoverability). Šis principas yra skirtas plačiai naudojamoms paslaugoms. Šiame darbe tinklo paslaugos naudojamos uždarame sistemų rate. Centrinėje sistemoje saugomas tikslus kelias iki kiekvienos klientinės sistemos tinklo paslaugos, todėl nereikia papildomai jų ieškoti. Klientinėje sistemoje net nebūtina saugoti centrinių sistemų adresų, kadangi centrinės sistemos adresas ateina kartu su kiekviena jos užklausa. Natūralu, kad jokia vieša paslaugų saugykla čia nėra reikalinga, todėl šio principo nėra laikomasi.

Paslaugos komponuojamumas (Service Composability). Šiame darbe sukurtos paslaugos yra atominės, jų nereikia skaidyti į mažesnes. Taigi šio principo nesilaikoma.

Šiame darbe yra pilnai arba dalinai laikomasi daugumos pagrindinių paslaugų projektavimo principų. Nesilaikoma tik tų principų, kurių laikytis šiuo specifiniu paslaugų projektavimo atveju nėra prasmės.

8.3. Išvados

1. Informacijos sistemas kuriančios organizacijos veiklos analizė parodė, kad užsakovų informacijos sistemų atnaujinimas reikalauja didelių laiko sąnaudų ir yra nepatogus galutiniams vartotojams, nes sutrikdo jų darbą. Taigi nuspręsta sukurti tinklo paslaugomis grindžiamą centralizuoto atnaujinimo modelį, kuris teigiamai paveiks sistemų kūrimo bei atnaujinimo procesą.
2. Atlikus egzistuojančių atnaujinimo sistemų, tinklo paslaugų standartų bei architektūros analizę nuspręsta, jog standartinė paslaugų architektūra šiame darbe kuriamam modeliui nėra tinkama dėl perteklinio elemento – UDDI registro. Dėl šios priežasties jo atsisakyta. Toks sprendimas sumažina paslaugų užklausų kiekį, jų vykdymo trukmę, tuo pačiu sumažindamas ir klaidų tikimybę.
3. Išnagrinėjus paslaugų projektavimo principus, kuriant atnaujinimo modelį daugumos jų buvo laikomasi. Nesilaikyta tik tų principų, kurių šiuo konkrečiu atveju nėra prasmės laikytis (*atrandamumas* ir *komponuojamumas*).
4. Tinklo paslaugų WSDL aprašo modelio išplėtimas kokybinėmis paslaugų charakteristikomis leidžia aprašą naudoti ne tik užklausų generavimo tikslais, bet ir

įvertinant paslaugų kokybę. Šiame darbe sukurtos sistemos atveju tokios charakteristikos gali būti papildomas privalumas derybose su užsakovu. Bendru atveju kokybinės paslaugų charakteristikos leistų paslaugų naudotojams lengviau apsispręsti, kurią iš analogiškų paslaugų pasirinkti, nes paslaugų kokybė tampa lemiamu kriterijumi.

5. Modelis, sukurtas laikantis iškeltų reikalavimų bei paslaugų projektavimo principų, yra pakankamai pilnas, kad būtų galima realizuoti centrinės bei klientinės sistemų prototipą bet kuria pasirinkta atviro kodo programavimo kalba.
6. Keleto centrinių bei klientinių sistemų pavyzdžių, atitinkančių sukurtą prototipą, realizacija parodė, kad atnaujinimo sistemos įdiegimas problemų nesukelia, sistema veikia be sutrikimų. Naudotis tiek centrine, tiek klientine sistema administratoriams yra paprasta, intuityvu.
7. Atliktas atnaujinimo pranešimų siuntimo klientinėms sistemoms trukmės eksperimentinis tyrimas parodė, kad ši trukmė yra tiesiškai priklausoma nuo pranešimų kiekio. Atlikus atnaujinimų vykdymo klientinėje sistemoje trukmės tyrimą paaiškėjo, kad šios trukmės priklausomybė nuo atnaujinamų kodo fragmentų kiekio taip pat yra tiesinė. Be to, atnaujinimo vykdymo trukmę bei fragmentų dydį taip pat sieja tiesinė priklausomybė. Šie faktai leidžia lengvai apskaičiuoti didžiausią galimą pranešimų siuntimo ar atnaujinimų vykdymo trukmę.
8. Atlikus užsakovų informacijos sistemų kodo formavimo trukmės pokyčių tyrimą nustatytos greito HTML puslapių formavimo reikalavimo tinkamumo kriterijaus (trukmė neturi padidėti daugiau kaip 5 %, lyginant su įprastu kodo formavimu) tenkinimo ribos. Galima teigti, kad šis kriterijus praktiškai tenkinamas, jei klientinėje sistemoje naudojamų probleminių fragmentų kiekis yra didesnis nei 50, nepaisant jų dydžio, arba jų dydžio vidurkis yra didesnis nei 3,5 KB, nepaisant jų kiekio. Esant mažam fragmentų kiekiui bei dydžio vidurkiui, puslapių trukmės padidėjimas gali būti didesnis nei 5 %.
9. Šis tyrimas įgalins informacijos sistemas kuriančias organizacijas centralizuotai atnaujinti įdiegtas užsakovų sistemas. Tai sumažins jų administravimo laiką, atnaujinimas vyks nepertraukiant sistemų veikimo ir netrukdant galutiniams vartotojams.

9. LITERATŪRA

1. Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju. *Web Services: concepts, architectures and applications*. – Berlin [etc.]: Springer, 2004. – 354 p.
2. Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman ir kt. *SOAP: Simple Object Access Protocol*. [interaktyvus]. 2000, balandis [žiūrėta 2006-10-22]. Prieiga per internetą: http://static.userland.com/xmlRpcCom/soap/SOAPv11.htm.
3. Ethan Cerami. *WSDL Essentials*. [interaktyvus]. 2003, vasaris [žiūrėta 2006-10-22]. Prieiga per internetą: http://www.developer.com/services/article.php/1602051.
4. Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. *Web Services Description Language (WSDL) 1.1*. [interaktyvus]. 2001, kovas [žiūrėta 2006-10-22]. Prieiga per internetą: http://www.w3.org/TR/wsdl.
5. Andrea D'Ambrogio. *A Model-driven WSDL Extension for Describing the QoS of Web Services*// IEEE International Conference on Web Services (ICWS'06): tarptautinės konferencijos pranešimų medžiaga [Chicago, 2006 m. rugsėjo 18-22 d.]. – Chicago, 2006, p. 789-796.
6. Jean-Luc David. *Creating and Consuming Web Services With PHP*. [interaktyvus]. 2004, kovas [žiūrėta 2007-06-20]. Prieiga per internetą: http://webservices.xml.com/pub/a/ws/2004/03/24/phpws.html.
7. Thomas Erl – SOA Systems, Inc. *SOA: Principles of Service Design*. – Indiana: RR Donnelley, 2007, liepa. – 608 p.
8. Forbis. *IB Pay. Atsiskaitymai per FORPOST*Internet Banking. Informacija pardavėjams*. [interaktyvus]. - [žiūrėta 2007-10-12]. Prieiga per internetą: http://www.sampo.lt/files/el.prekyba.ibpay.LT.pdf.
9. Hao He. *What Is Service-Oriented Architecture*. [interaktyvus]. 2003, rugsėjis [žiūrėta 2006-10-16]. Prieiga per internetą: http://www.xml.com/pub/a/ws/2003/09/30/soa.html.
10. Raghu R. Kodali, JavaWorld.com. *What Is Service-Oriented Architecture*. [interaktyvus]. 2005, birželis [žiūrėta 2006-10-16]. Prieiga per internetą: http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html.
11. Scott Nichol. *Introduction to NuSOAP*. [interaktyvus]. 2004, lapkritis [žiūrėta 2007-06-20]. Prieiga per internetą: http://www.scottnichol.com/nusoapintro.htm.
12. Duane Nickull – Adobe Systems, Inc. *Service Oriented Architecture Whitepaper*. [interaktyvus]. 2005, vasaris [žiūrėta 2006-10-16]. Prieiga per internetą: http://www.adobe.com/enterprise/pdfs/Services_Oriented_Architecture_from_Adobe.pdf.

13. Object Management Group. *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms*. [interaktyvus]. 2006, gegužė [žiūrėta 2007-11-24]. Prieiga per internetą: http://www.omg.org/docs/formal/06-05-02.pdf.
14. Rational Software. *Rational Unified Process. Best Practices for Software Development Teams.* [interaktyvus]. 2001, lapkritis [žiūrėta 2007-12-15]. Prieiga per internetą: http://www.cse.ohio-state.edu/~neelam/courses/788/rup.pdf.
15. The OpenSSL Project. *OpenSSL dokumentacija*. [interaktyvus]. - [žiūrėta 2007-10-12]. Prieiga per internetą: http://www.openssl.org/docs/apps/openssl.html.
16. W3C Working Group. *Web Services Architecture*. [interaktyvus]. 2004, vasaris [žiūrėta 2006-10-22]. Prieiga per internetą: http://www.w3.org/TR/ws-arch/.

10. PRIEDAI

1 priedas. Programinio kodo fragmentų pavyzdžiai

Šiame priede pateikiama keletas kodo fragmentų pavyzdžių.

1 pavyzdys – įvedamų asmens duomenų tikrinimo PHP klasė:

```
<?php
class Kontrole_asmuo extends Kontrole_kita
{
  function tinka_vardas($txt)
  {
    if (strlen($txt) > 100) return "Vardo ilgis turi būti nuo 3 iki 100 simbolių.";
    if (strlen($txt) < 3) return "Vardo ilgis turi būti nuo 3 iki 100 simbolių.";
    if (!eregi("^([ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)( [ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)*$",$txt)) return "Vardas gali būti sudarytas tik iš raidžių
bei tarpo simbolio (atskirti du vardus).";

    return true;
  }

  function tinka_pavarde($txt)
  {
    if (strlen($txt) > 50) return "Pavardės ilgis turi būti nuo 3 iki 50 simbolių.";
    if (strlen($txt) < 3) return "Pavardės ilgis turi būti nuo 3 iki 50 simbolių.";
    if (!eregi("^([ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)( [ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)?$",$txt)) return "Pavardė gali būti sudaryta tik iš raidžių
bei tarpo simbolio (atskirti dvi pavardes).";

    return true;
  }

  function tinka_vardas_pavarde($txt)
  {
    if (strlen($txt) > 100) return "Vardo ilgis turi būti nuo 3 iki 100 simbolių.";
    if (strlen($txt) < 3) return "Vardo ilgis turi būti nuo 3 iki 100 simbolių.";
    if (!eregi("^([ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)( [ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)*$",$txt)) return "Vardas gali būti sudarytas tik iš raidžių
bei tarpo simbolio (atskirti du vardus).";

    return true;
  }

  function tinka_kontakt_asmens_vardas_pavarde($txt)
  {
    if (strlen($txt) > 100) return "Kontaktinio asmens vardo ilgis turi būti nuo 3 iki 100 simbolių.";
    if (strlen($txt) < 3) return "Kontaktinio asmens vardo ilgis turi būti nuo 3 iki 100 simbolių.";
    if (!eregi("^([ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)( [ĄČĘĖĮŠŲŪŽąčęėįšųūža-z]+)*$",$txt)) return "Kontaktinio asmens vardas gali būti
sudarytas tik iš raidžių bei tarpo simbolio (atskirti du vardus).";

    return true;
  }

  function tinka_gim_data($txt)
  {
    $d = explode ('-',$txt);
    if (count($d) != 3) return "Netinkamai pasirinkta gimimo data.";
    $dd = date("Y-m-d",mktime(0,0,0,$d[1]*1,$d[2]*1,$d[0]*1));
    if ($txt != $dd) return "Netinkamai pasirinkta gimimo data.";

    return true;
  }
}
?>
```

2 pavyzdys – šablonizavimo PHP klasė:

```
<?
  class artemplate
  {
    var $Kintamieji = array(); //kintamųjų masyvas

    function naujasKint($vardas,$reiksme)
    {
      if (is_array($reiksme)) $this->Kintamieji[$vardas] = array();
      $this->Kintamieji[$vardas] = $reiksme;
    }

    function apdoroti($tekstas,$kintamieji = '')
    {
      if ($kintamieji == '') $kintamieji = $this->Kintamieji;
      $naujas = "";
      $ilgis = strlen($tekstas);
      $i = 0;
      while ($i < $ilgis)
      {
        if ($tekstas[$i] != '{')
        {
          $naujas .= $tekstas[$i];
          $i++;
        }
        else
        {
          if ($tekstas[$i+1] != '{')
          {
            $naujas .= $tekstas[$i];
            $i++;
          }
          else
          {
            $i += 2;
            $kint = "";
            while (($tekstas[$i] != '}') || ($tekstas[$i+1] != '}'))
            {
              $kint .= $tekstas[$i];
              $i++;
            }
            $i += 2;
            if (substr($kint,0,7) == "CIKLAS+") /*CIKLO PRADŽIA!!!*/
            {
              $cikloPavad = substr($kint,7);
              $cikloTurinys = "";
              $cikloPabaiga = "{{CIKLAS-".$cikloPavad."}}";
              $cikloPabaigosIlgis = strlen($cikloPabaiga);
              while (substr($tekstas,$i,$cikloPabaigosIlgis) != $cikloPabaiga)
              {
                $cikloTurinys .= $tekstas[$i];
                $i++;
              }
              $i += $cikloPabaigosIlgis;
              while (list(,$value) = each($kintamieji[$cikloPavad]))
              {
                if (is_array($value))
                {
                  $naujas .= $this->apdoroti($cikloTurinys,$value);
                }
              }
            }
            else if (substr($kint,0,7) == "ALTERN+") /*ALTERNATYVOS PASIRINKIMO PRADŽIA!!!*/
            {
              $alternPavadIrReiksme = substr($kint,7);

              list($alternPavad,$alternReiksme) = explode("+",$alternPavadIrReiksme,2);
              if ($alternReiksme != $kintamieji[$alternPavad])
```

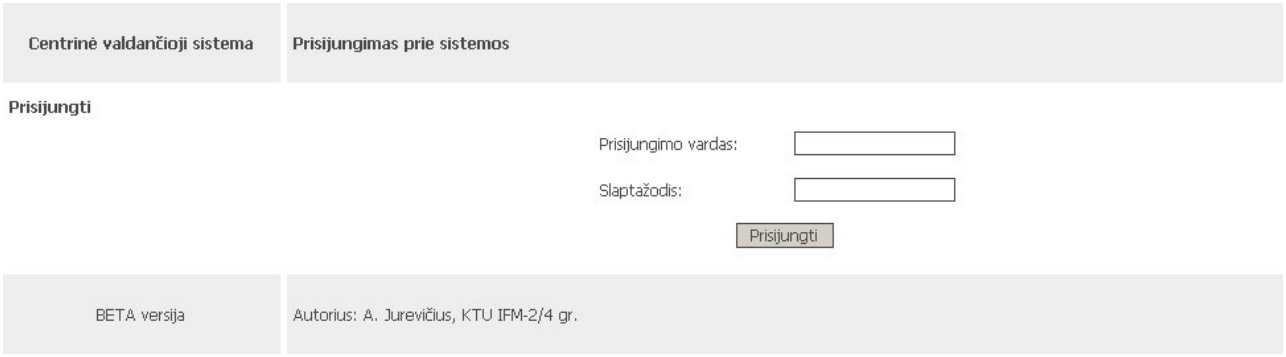
```
              {
                $alternPradzia = "{{ALTERN+".$alternPavad."+".$kintamieji[$alternPavad]."}}";
                $alternPradziosIlgis = strlen($alternPradzia);
                while (substr($tekstas,$i,$alternPradziosIlgis) != $alternPradzia)
                {
                  $i++;
                }
                $i += $alternPradziosIlgis;
              }
              $alternTurinys = "";
              $alternPabaiga = "{{ALTERN-".$alternPavad."}}";
              $alternKitaPradzia = "{{ALTERN+".$alternPavad."+";
              $alternPabaigosIlgis = strlen($alternPabaiga);
              $alternKitosPradziosIlgis = strlen($alternKitaPradzia);
              while ((substr($tekstas,$i,$alternPabaigosIlgis) != $alternPabaiga) && (substr($tekstas,$i,$alternKitosPradziosIlgis)
!= $alternKitaPradzia))
              {
                $alternTurinys .= $tekstas[$i];
                $i++;
              }
              if (substr($tekstas,$i,$alternPabaigosIlgis) == $alternPabaiga)
              {
                $i += $alternPabaigosIlgis;
              }
              else
              {
                while (substr($tekstas,$i,$alternPabaigosIlgis) != $alternPabaiga)
                {
                  $i++;
                }
                $i += $alternPabaigosIlgis;
              }
              $naujas .= $this->apdoroti($alternTurinys,$kintamieji);
            }
            else /*PAPRASTAS KINTAMASIS!!!*/
            {
              $naujas .= $kintamieji[$kint];
            }
          }
        }
      }
      return $naujas;
    }
  }
?>
```

3 pavyzdys – paprasčiausia pavardės tinkamumo tikrinimo JavaScript funkcija:

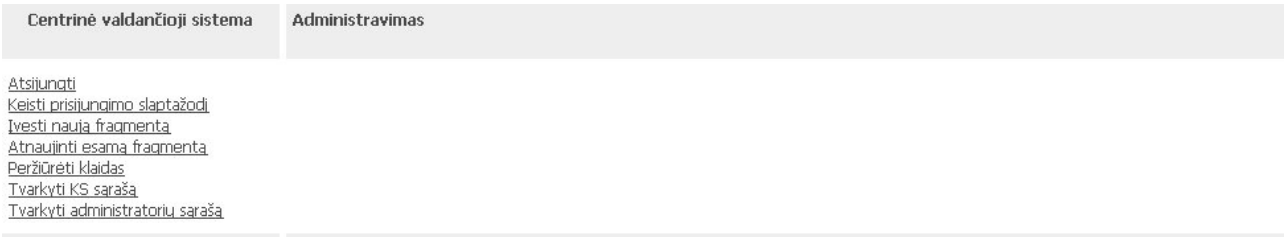
```
<script language="JavaScript" type="text/javascript">
    function ok_pavarde(txt)
    {
        var ok = true;
        if (txt.length == 0) ok = false;
        return ok;
    }
</script>
```

2 priedas. Įdiegto prototipo vartotojo sąsajos pavyzdžiai

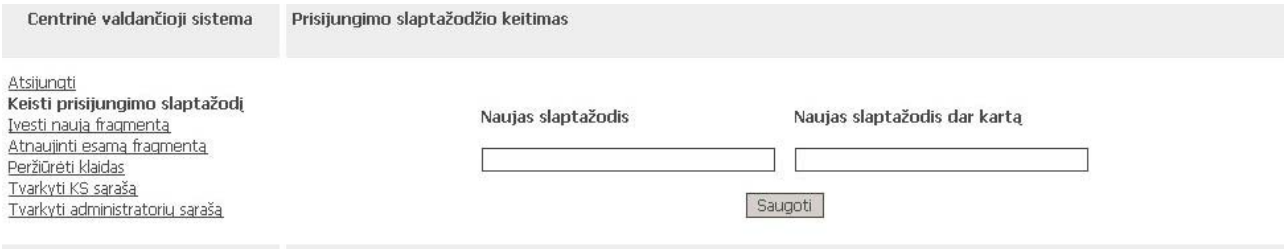
Šiame priede pateikiami centrinės bei klientinės sistemų vartotojo sąsajos pavyzdžiai.



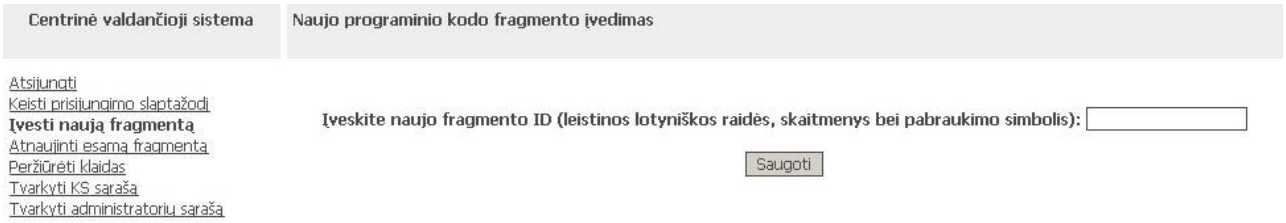
94 pav. Centrinės sistemos prisijungimo langas (analogiškas klientinės sistemos prisijungimo langui)



95 pav. Centrinės sistemos pagrindinis langas (prisijungus)

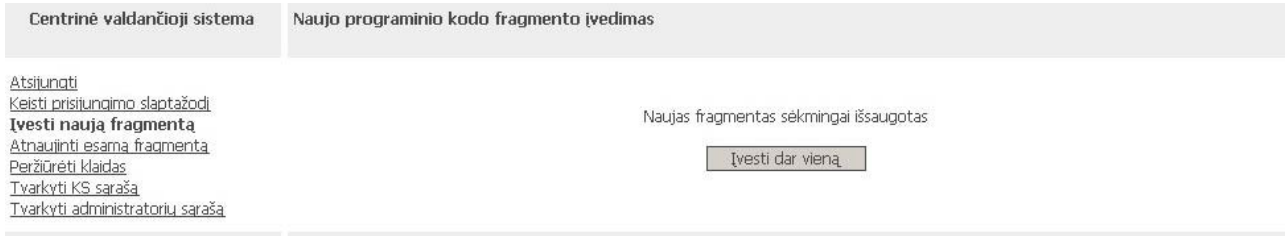


96 pav. Centrinės sistemos administratoriaus slaptažodžio keitimo langas (analogiškas klientinės sistemos administratoriaus slaptažodžio keitimo langui)

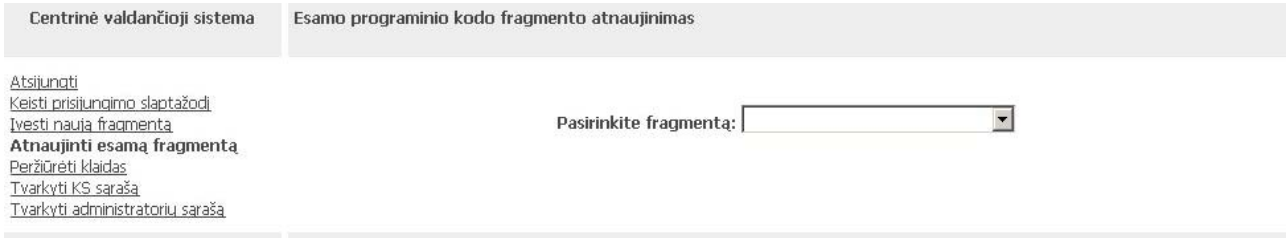


97 pav. Naujo kodo fragmento įvedimo centrinėje sistemoje langas

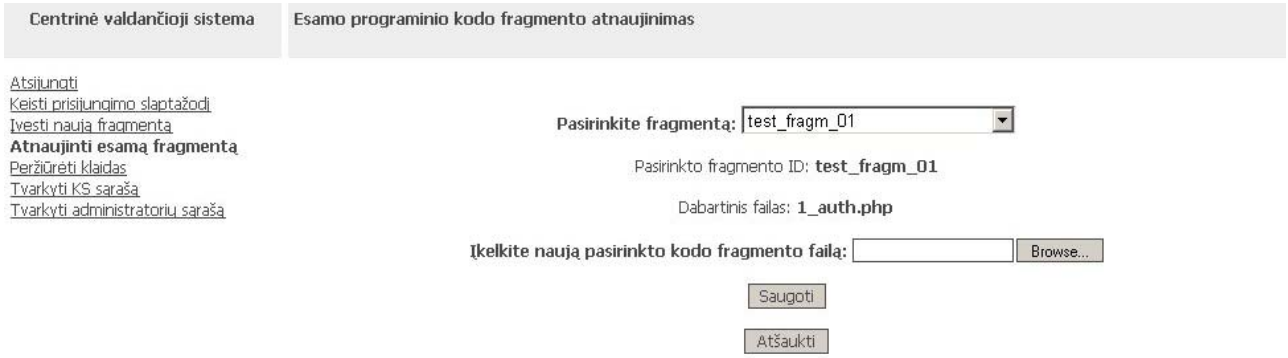
98 pav. Naujo kodo fragmento įvedimo centrinėje sistemoje langas (įvedus fragmento ID)



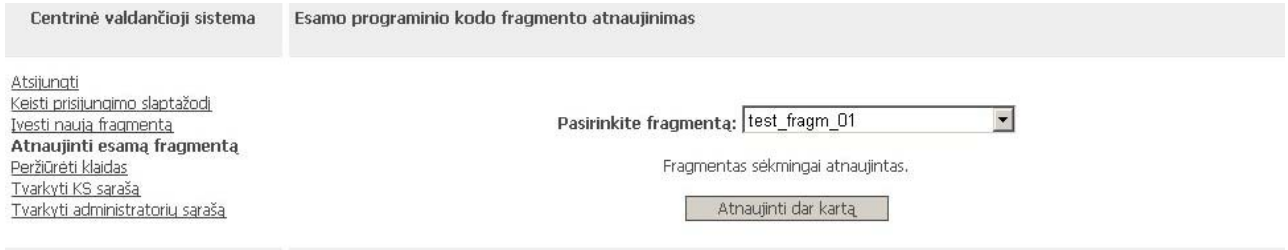
99 pav. Naujo kodo fragmento įvedimo centrinėje sistemoje langas (įkėlus fragmento failą)



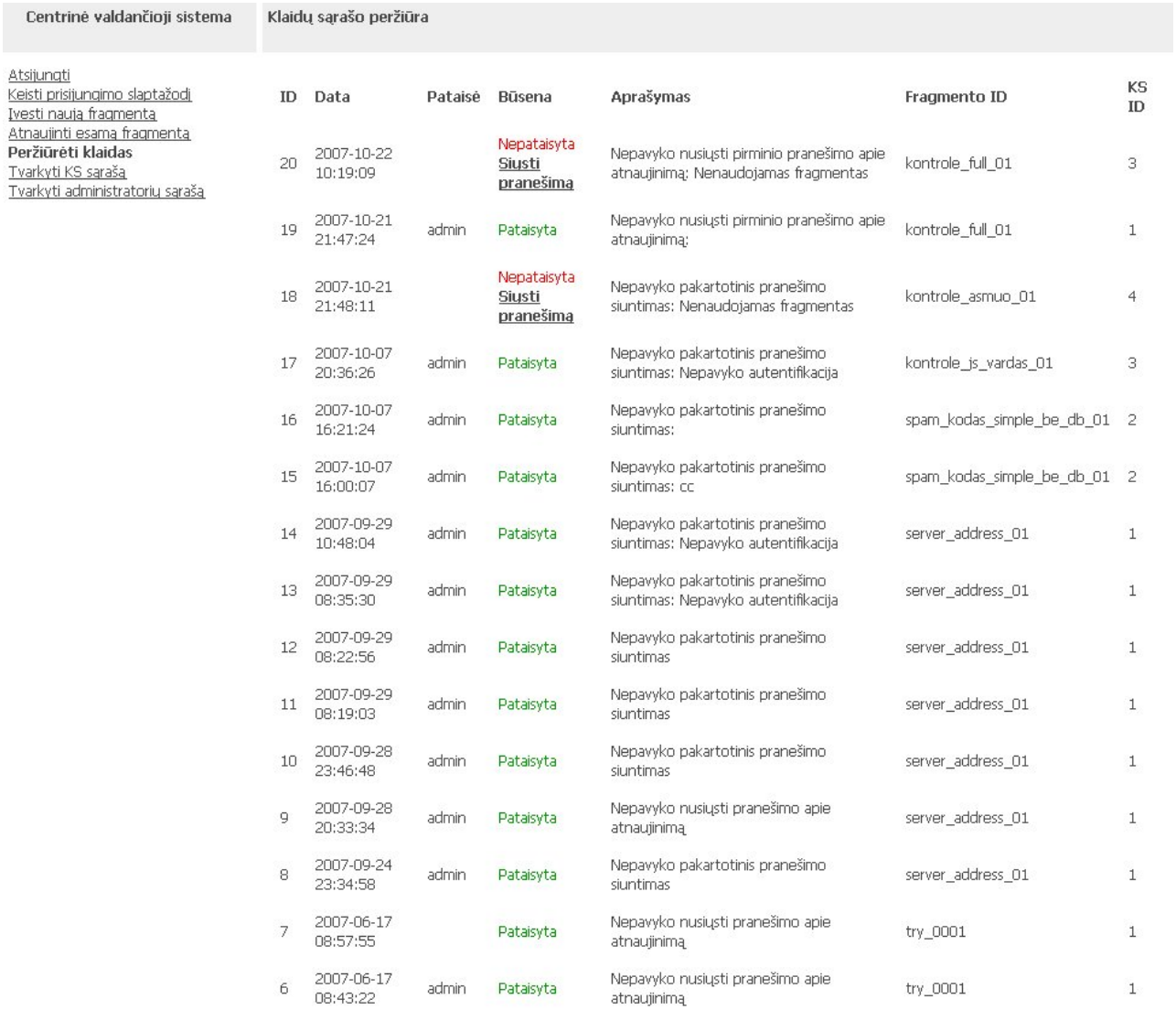
100 pav. Esamo kodo fragmento atnaujinimo centrinėje sistemoje langas



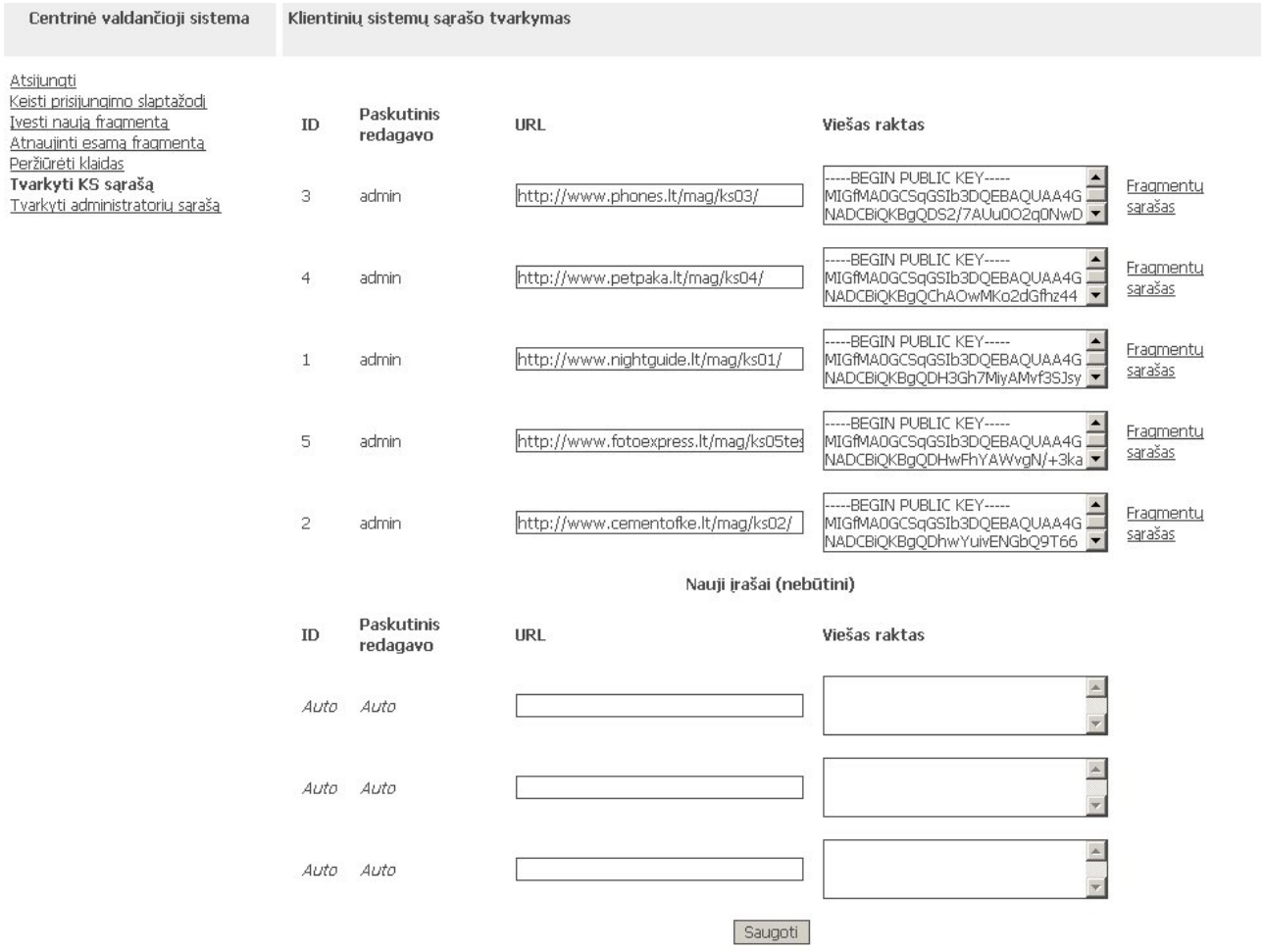
101 pav. Esamo kodo fragmento atnaujinimo centrinėje sistemoje langas (pasirinkus fragmentą)



102 pav. Esamo kodo fragmento atnaujinimo centrinėje sistemoje langas (įkėlus naują failą)

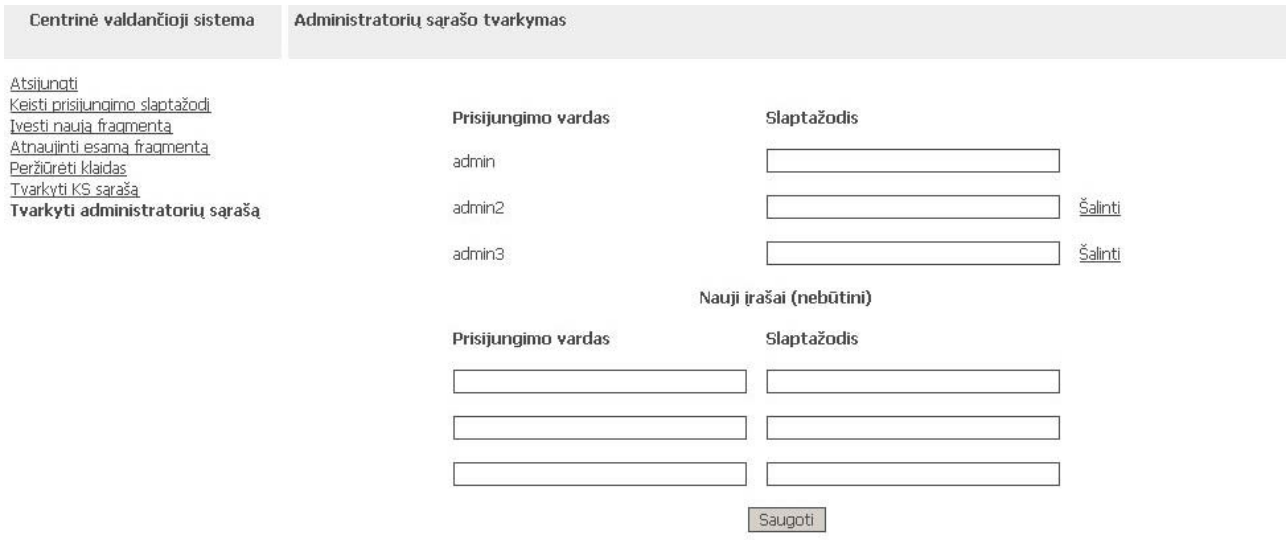


103 pav. Centrinės sistemos klaidų peržiūros langas

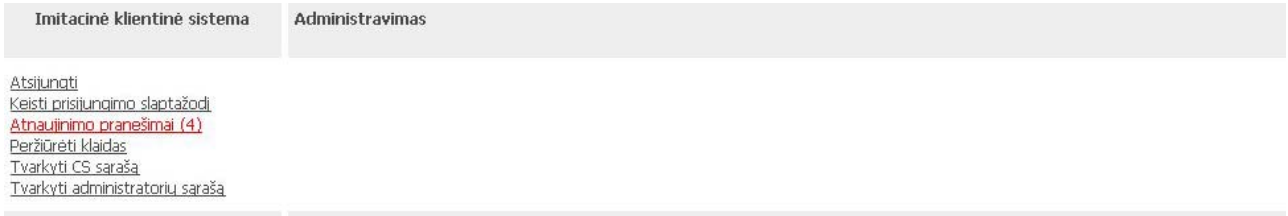


104 pav. Klientinių sistemų sąrašo tvarkymo centrinėje sistemoje langas

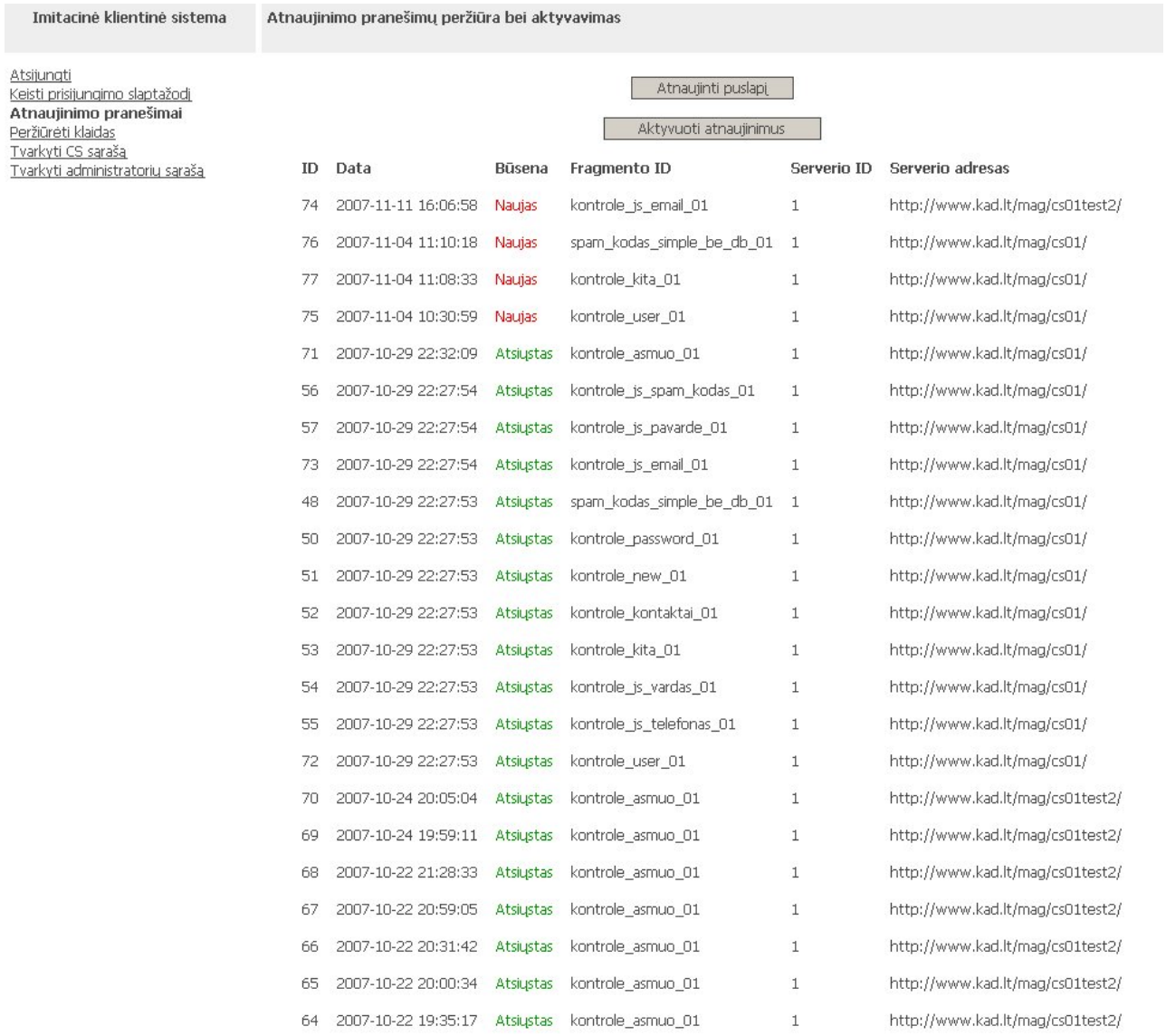
105 pav. Klientinės sistemos naudojamų fragmentų sąrašo tvarkymo centrinėje sistemoje langas



106 pav. Administratorių sąrašo tvarkymo centrinėje sistemoje langas (analogiškas klientinės sistemos administratorių sąrašo tvarkymo langui)

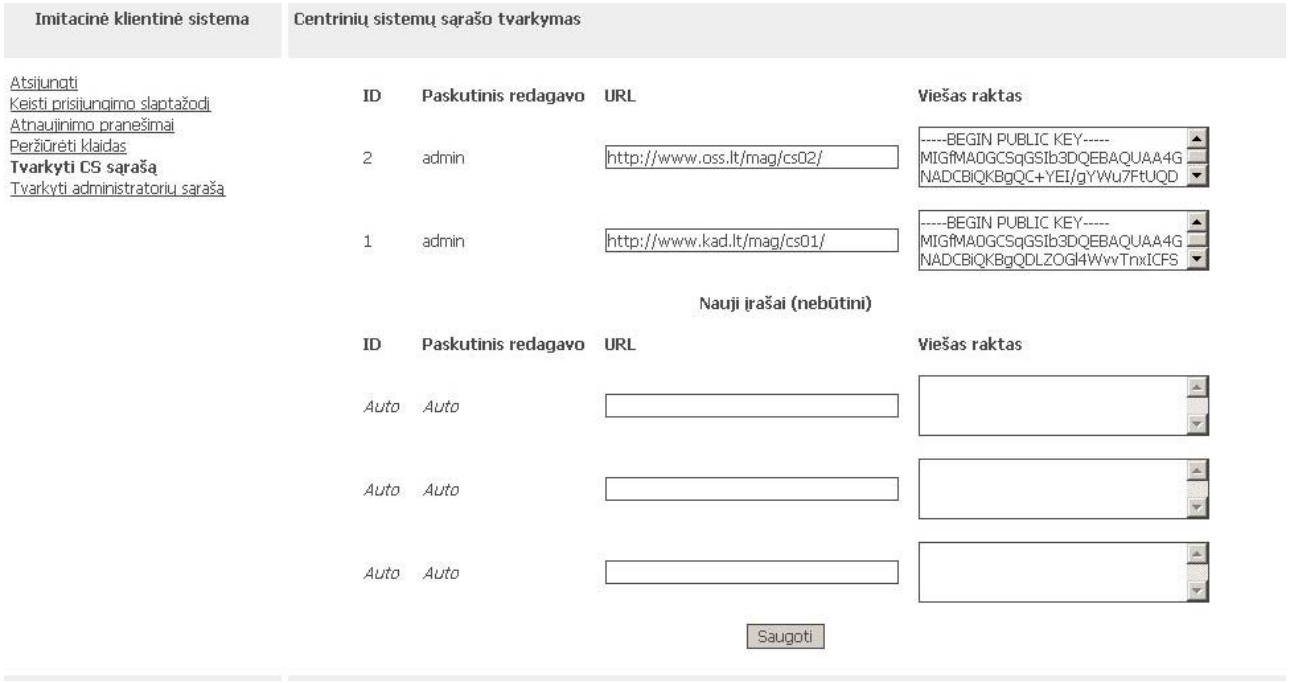


107 pav. Klientinės sistemos pagrindinis langas (prisijungus)

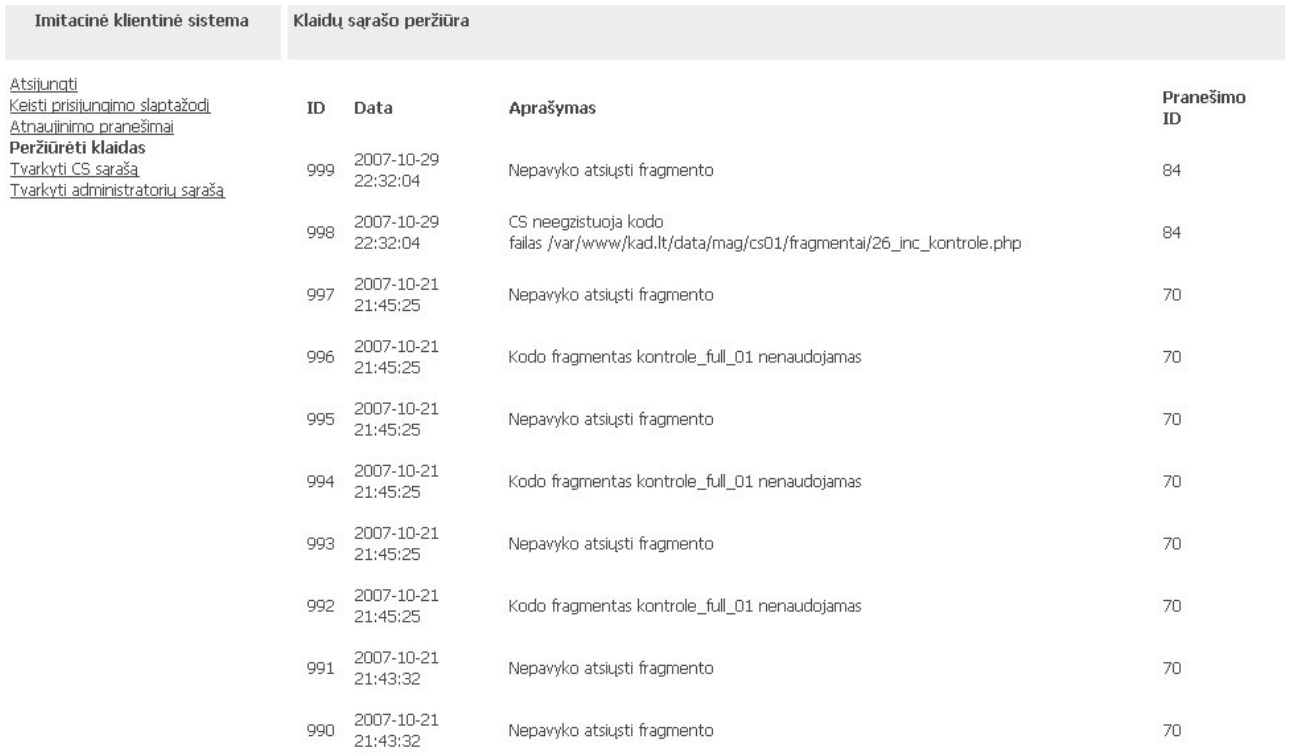


ID	Data	Būsena	Fragmento ID	Serverio ID	Serverio adresas
74	2007-11-11 16:06:58	Naujas	kontrole_js_email_01	1	http://www.kad.lt/mag/cs01test2/
76	2007-11-04 11:10:18	Naujas	spam_kodas_simple_be_db_01	1	http://www.kad.lt/mag/cs01/
77	2007-11-04 11:08:33	Naujas	kontrole_kita_01	1	http://www.kad.lt/mag/cs01/
75	2007-11-04 10:30:59	Naujas	kontrole_user_01	1	http://www.kad.lt/mag/cs01/
71	2007-10-29 22:32:09	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01/
56	2007-10-29 22:27:54	Atsiųstas	kontrole_js_spam_kodas_01	1	http://www.kad.lt/mag/cs01/
57	2007-10-29 22:27:54	Atsiųstas	kontrole_js_pavarde_01	1	http://www.kad.lt/mag/cs01/
73	2007-10-29 22:27:54	Atsiųstas	kontrole_js_email_01	1	http://www.kad.lt/mag/cs01/
48	2007-10-29 22:27:53	Atsiųstas	spam_kodas_simple_be_db_01	1	http://www.kad.lt/mag/cs01/
50	2007-10-29 22:27:53	Atsiųstas	kontrole_password_01	1	http://www.kad.lt/mag/cs01/
51	2007-10-29 22:27:53	Atsiųstas	kontrole_new_01	1	http://www.kad.lt/mag/cs01/
52	2007-10-29 22:27:53	Atsiųstas	kontrole_kontaktai_01	1	http://www.kad.lt/mag/cs01/
53	2007-10-29 22:27:53	Atsiųstas	kontrole_kita_01	1	http://www.kad.lt/mag/cs01/
54	2007-10-29 22:27:53	Atsiųstas	kontrole_js_vardas_01	1	http://www.kad.lt/mag/cs01/
55	2007-10-29 22:27:53	Atsiųstas	kontrole_js_telefonas_01	1	http://www.kad.lt/mag/cs01/
72	2007-10-29 22:27:53	Atsiųstas	kontrole_user_01	1	http://www.kad.lt/mag/cs01/
70	2007-10-24 20:05:04	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/
69	2007-10-24 19:59:11	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/
68	2007-10-22 21:28:33	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/
67	2007-10-22 20:59:05	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/
66	2007-10-22 20:31:42	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/
65	2007-10-22 20:00:34	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/
64	2007-10-22 19:35:17	Atsiųstas	kontrole_asmuo_01	1	http://www.kad.lt/mag/cs01test2/

108 pav. Atnaujinimo pranešimų peržiūros ir aktyvavimo klientinėje sistemoje langas



109 pav. Centrinių sistemų sąrašo tvarkymo klientinėje sistemoje langas



110 pav. Klaidų peržiūros klientinėje sistemoje langas

3 priedas. Straipsnis

CENTRALIZUOTO INFORMACIJOS SISTEMŲ ATNAUJINIMO PASLAUGŲ MODELIS IR PROTOTIPAS

Artūras Jurevičius, Lina Nemuraitė

Informacijos sistemų katedra, Kauno technologijos universitetas
Studentų 50, LT 51368 Kaunas

Santrauka. Organizacija, teikianti informacijos sistemų kūrimo paslaugas, dažnai turi užtikrinti ir įdiegtų sistemų palaikymą. Paprastai tokia įmonė naudoja tuos pačius arba panašius komponentus skirtinguose projektuose. Taigi jei viename iš komponentų yra klaidų, įmonė turi atlikti pataisymus atskirai kiekvienoje sistemoje. Šiame straipsnyje siūlomas tinklo paslaugomis grindžiamas centralizuotas informacijos sistemų atnaujinimo modelis atnaujinimo laiko sąnaudoms sumažinti. Idėja yra realizuoti probleminius komponentus atskiruose failuose, vadinamuose fragmentais. Vykstant atnaujinimui sukuriamas naujas failas bei pakeičiama atnaujinamo fragmento nuoroda į failą duomenų bazėje. Tokiu būdu užsakovų informacijos sistemų puslapiai visada generuojami naudojant naujausius komponentus.

1. Įvadas

Užsakovų informacijos sistemas kurianti bei diegianti organizacija dažnai turi užtikrinti ir įdiegtų sistemų palaikymą. Tačiau sistemų atnaujinimas reikalauja didelių laiko sąnaudų. Panašiems uždaviniams spręsti skirtingose sistemose naudojami tie patys ar panašūs komponentai. Taigi jeigu dėl kokių nors priežasčių būtina atlikti pakeitimus vienoje iš įmonės įdiegtų sistemų, didelė tikimybė, kad tokius pat pakeitimus reikės atlikti ir daugelyje kitų jos realizuotų sistemų. Tokiu atveju tiek įmonė patirs nuostolių dėl perteklinio darbo, tiek užsakovams gali būti padaryta žala dėl užsitęsusio netinkamo sistemų veikimo. Natūralu, kad kuriamų sistemų programinio kodo fragmentus (komponentus, funkcijas), kuriems yra tikimybė ateityje keistis, naudinga realizuoti taip, kad atlikus pakeitimus vienoje sistemoje, jie atsispindėtų ir kitose to reikalaujančiose sistemose. Kitaip sakant, kad būtų galima sumažinti sistemų atnaujinimo laiko sąnaudas, būtina centralizuota architektūra.

Informacijos sistemų puslapių formavimui šiame darbe siūloma tokia idėja: kiekvienas probleminis programinio kodo fragmentas, kuriam yra bent menkiausia tikimybė ateityje keistis, realizuojamas atskiru failu, o centralizuoto atnaujinimo metu sukuriamas naujas fragmento failas bei pakeičiama nuoroda į jį duomenų bazėje. Tokiu būdu užtikrinama, kad kiekvieną kartą formuojant informacijos sistemos puslapį bus naudojami atnaujinti programinio kodo failai.

Šiame darbe informacijos sistemų atnaujinimo problema sprendžiama kuriant centralizuoto atnaujinimo modelį tinklo paslaugų (angl. Web services) pagrindu. Tinklo paslaugų pirminė paskirtis yra spręsti įmonių taikomųjų uždavinių integravimo uždavinius pasauliniu mastu. Tačiau šiame darbe parodoma, kad atnaujinimo tikslais ir uždarame informacijos sistemų rate galima pritaikyti tinklo paslaugų teikiamas galimybes.

Problemą spręsti tinklo paslaugų pagrindu pasirinkta dėl standartizavimo ir plečiamumo 7. Tinklo paslaugomis paremtai sistemai neturi įtakos naudojamos technologijos, sąveikaujančios sistemos gali būti realizuotos skirtingose platformose.

Analizuojant paslaugų architektūrą remtasi [1], [9], [10], [12] šaltiniais. Išskirtinai tinklo paslaugoms nagrinėti naudotas [15] šaltinis. SOAP bei WSDL standartams analizuoti naudoti atitinkamai [2] ir [3], [4] šaltiniai. Paslaugų aprašymo išplėtimui kokybinėmis charakteristikomis remtasi [5] ir [13] šaltiniais. Konkrečiai tinklo paslaugų realizacijai PHP kalba remtasi [6] bei [11] šaltiniais. Tarp sistemų perduodamų duomenų saugumui užtikrinti remtasi IB Pay elektroninės bankininkystės paslaugų technine specifikacija [8] bei OpenSSL dokumentacija [14].

2 skyriuje aprašomi egzistuojantys atnaujinimo problemos sprendimai. 3 skyriuje aprašoma siūloma sprendimo idėja bei pateikiami esminiai atnaujinimo paslaugų modelio elementai. Be to, pateikiamas WSDL modelio išplėtimas kokybinėmis charakteristikomis ir aprašomas sprendimo saugumas, patikimumas. 4 skyriuje aprašomas eksperimentinis modelio realizacijos tyrimas. 5 skyriuje pateikiamos darbo išvados.

2. Esamos situacijos analizė

Analogiškų viešai paskelbtų paslaugų, kurios būtų skirtos informacijos sistemų atnaujinimui, nėra. Šiame poskyryje nagrinėjamas taikomųjų programų bei lietuviško produkto E-pasas atnaujinimas.

Atnaujinimo problema taikomosiose programose sprendžiama specialių programinių agentų ar posistemių pagalba. Startuojant taikomajai programai ar net jos naudojimo metu tikrinama, ar nėra atnaujinimų centriniame serveryje. Jei jų yra, tolesnis scenarijus gali būti įvairus. Pats paprasčiausias – atsisiunčiama nauja programos versija bei įdiegiama taip pat, kaip ir įdiegiant pirmą kartą. Bet vartotojui patogesnis toks atnaujinimo būdas, kai atnaujinimai

įdiegiami automatiškai arba gavus jo sutikimą. Šis būdas paplitęs daugelyje internetu atnaujinamų programinių produktų (pvz., daugelis Microsoft paketų, Mozilla Firefox naršyklė ir kt.).

Aukščiau aprašyti du scenarijai yra skirti taikomosioms programoms atnaujinti, tačiau informacijos sistemų atnaujinimui jie gali būti tik kaip pavyzdžiai. Šiek tiek panašumų į šiame darbe siūlomą atnaujinimo modelį informacijos sistemų terpėje turi lietuviškos E-paso prisijungimo sistemos (www.e-pasas.lt) atnaujinimas. E-pasas yra integruojamas produktas, skirtas prisijungimui su tais pačiais duomenimis keliuose tinklalapiuose. Integravus jį tinklalapyje, laikas nuo laiko gali būti vykdomas pagrindinių E-paso failų atnaujinimas. Tačiau jo scenarijus šiam darbui netinka, nes jis pritaikytas būtent tik E-pasui. Kad ir koks tinklalapių kiekis bebūtų, visuose juose E-paso failai bus vienodi. Šiame darbe priešingai – modelis skirtas skirtingoms informacijos sistemoms, kuriose dalis atnaujinamų programinio kodo fragmentų sutampa, atnaujinti. Be to, E-paso atnaujinimo scenarijaus trūkumas yra tai, kad atnaujinimo metu E-pasas yra atjungiamas.

1 lentelėje pateikiamas analizuotų scenarijų apibendrinimas.

1 lentelė. Analizuotų egzistuojančių atnaujinimo scenarijų apibendrinimas

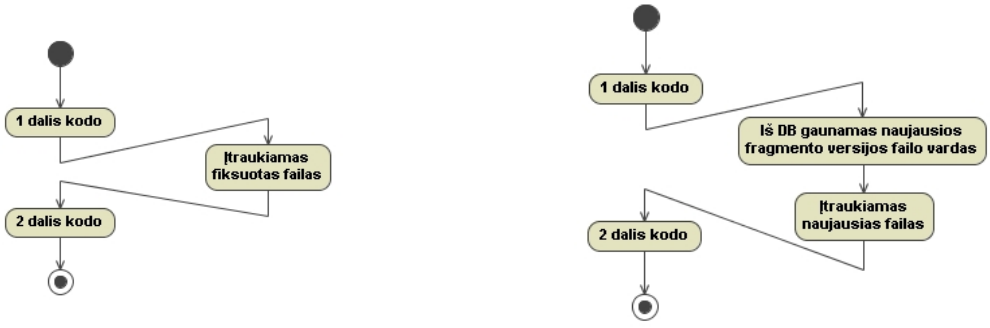
	Naujos versijos atsisiuntimas ir įdiegimas iš naujo	**Atnaujinimų įdiegimas minimaliai trikdant vartotojo darbą**	**E-paso failų atnaujinimo tinklalapiuose scenarijus**
Paskirtis	Taikomoji programinė įranga.	Taikomoji programinė įranga.	Interneto informacijos sistemos.
Patikimumas	Patikimiausias.	Gali būti nepatikimas.	Pakankamai patikimas.
Saugumas	Saugus, nebent vartotojas naują versiją atsisiunčia iš nepatikimo šaltinio.	Gali būti labai nesaugus, jeigu atnaujinimų serveris nėra autentifikuojamas.	Pakankamai saugus (šifravimas slaptuoju raktu).
Privalumai	Kūrėjams nereikia realizuoti atnaujinimų apdorojimo.	Vartotojo darbas pertraukiamas minimaliai. Vartotojui suteikiama galimybė atidėti atnaujinimą.	Nesėkmės atveju atstatomi seni failai.
Trūkumai	Nepatogu vartotojui.	Kūrėjams būtina realizuoti atnaujinimų apdorojimą.	Atnaujinimo metu E-pasas atjungiamas nuo tinklalapio.

3. Siūlomas informacijos sistemų atnaujinimo problemos sprendimas

Šiame skyriuje aprašoma problemos sprendimo idėja bei esminiai sukurto modelio elementai. Taip pat pateikiamas tinklo paslaugų modelio išplėtimas kokybinėmis charakteristikomis. Galiausiai aprašomas sprendimo saugumas bei patikimumas.

3.1. Sprendimo idėjos aprašymas

Šiame poskyryje parodoma, kuo šiame darbe siūlomu modeliu bei centralizuota architektūra paremtas informacijos sistemų kūrimas yra pranašesnis už įprastus metodus. 1 paveiksle a parodyta, kaip PHP ar kita kalba formuojamas HTML kodas įprastose užsakovų sistemose. Vykdant pagrindinį kodo formavimo scenarijų tenka įtraukti kitus failus. Tačiau tie failai yra fiksuoti, t. y. niekada neatnaujinami arba atnaujinami rankiniu būdu kiekvienoje sistemoje atskirai.



a) b).

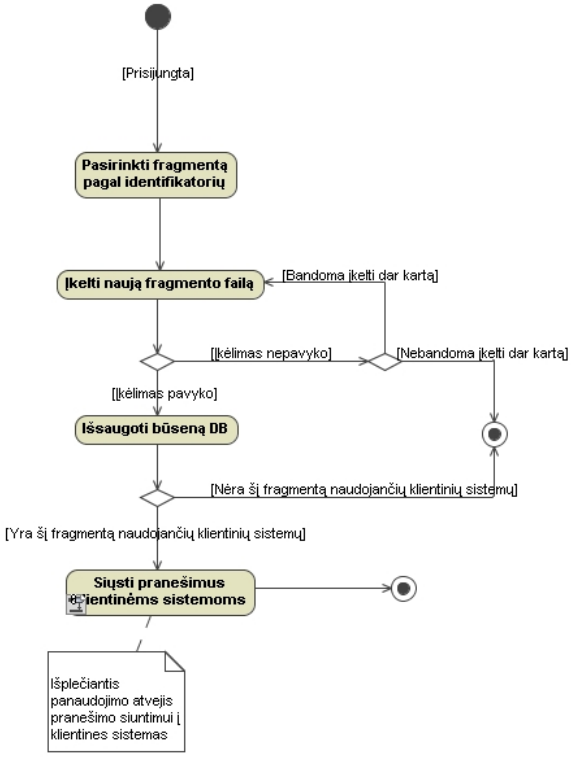
1 pav. Įprastos (a) ir atnaujinamos klientinės sistemos (b) HTML teksto formavimo schemos fragmentas

1 paveiksle b parodyta patobulinta schema, kuri atspindi šiame darbe kuriamą modelį. Scenarijus iš pirmo žvilgsnio lieka tas pats – formuojant HTML tekstą tenka įtraukti papildomus failus. Tačiau čia vietoje to, kad būtų įtraukiamas fiksuotas failas, yra kreipiamasi į duomenų bazę, kad būtų gautas konkretaus fragmento naujausios versijos failo vardas. Tuomet įtraukiamas būtent naujausias tam fragmentui skirtas failas. Kad nereikėtų daug kartų kreiptis į

duomenų bazę, prieš pradedant formuoti kodą galima iš jos išgauti visų naudojamų fragmentų naujausių failų vardus. Realizuojant užsakovo informacijos sistemą pagal 1 paveiksle b pateiktą schemą bus užtikrintas naujausių fragmentų versijų panaudojimas.

3.2. Esminių modelio elementų aprašymas

Šiame poskyryje aprašomi esminiai centrinės bei klientinės sistemų modeliai, skirti tam, kad kodo fragmentai visada būtų atnaujinti. 2 pav. pateikta esamo kodo fragmento atnaujinimo centrinėje sistemoje veiklos diagrama. Vykdymo metu pasirenkamas vienas iš įvestų fragmentų bei įkeliamas naujas failas. Atnaujinus fragmentą, jį naudojančioms klientinėms sistemoms siunčiami pranešimai.



2 pav. Esamo kodo fragmento atnaujinimo veiklos modelis

3 paveiksle pateikta atnaujinimo vykdymo klientinėje sistemoje veiklos diagrama. Atnaujinimai vykdomi tada, kai yra naujų pranešimų ir juos aktyvuoja administratorius.

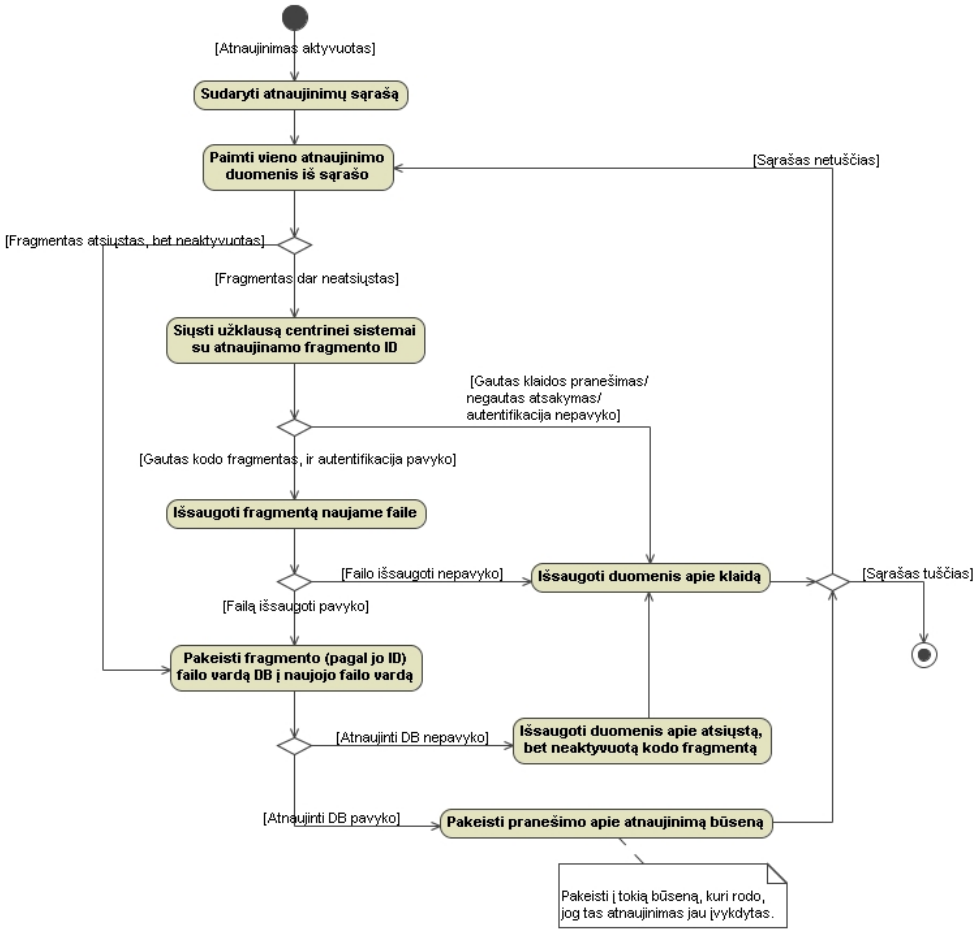
3.3. Tinklo paslaugos aprašymas bei išplėtimas kokybinėmis charakteristikomis

Šiame poskyryje aprašoma atnaujinimo paslauga. Centrinės sistemos tinklo paslauga skirta nusiųsti programinio kodo fragmentą užklaususiai klientinei sistemai. Siuntimas vykdomas kreipiantis į operaciją „siusti_atnaujinima“. Jos įeinantys parametrai yra tokie:

- fragmento identifikatorius (*fid*);
- užklaususios sistemos adresas (*adresas*);
- užklausos išsiuntimo laikas (*laikas*);
- skaitmeninis parašas (*parasas*) užklausėjo autentifikavimui.

Laiko parametras naudojamas tik tam, kad kiekvieną kartą būtų sugeneruotas skirtingas parašas. Operacijos išeinantis parametras yra *return*, kuriuo grąžinamas serializuotas (paverstas į tekstinę eilutę) masyvas. Jo turinys – vienas iš šių variantų:

- kodo fragmentas, užšifruotas sugeneruotu vienkartiniu slaptu raktu, bei pats slaptas raktas, užšifruotas viešu gavėjo raktu (3.5 posk.);
- įvykus klaidai – klaidos pranešimas.

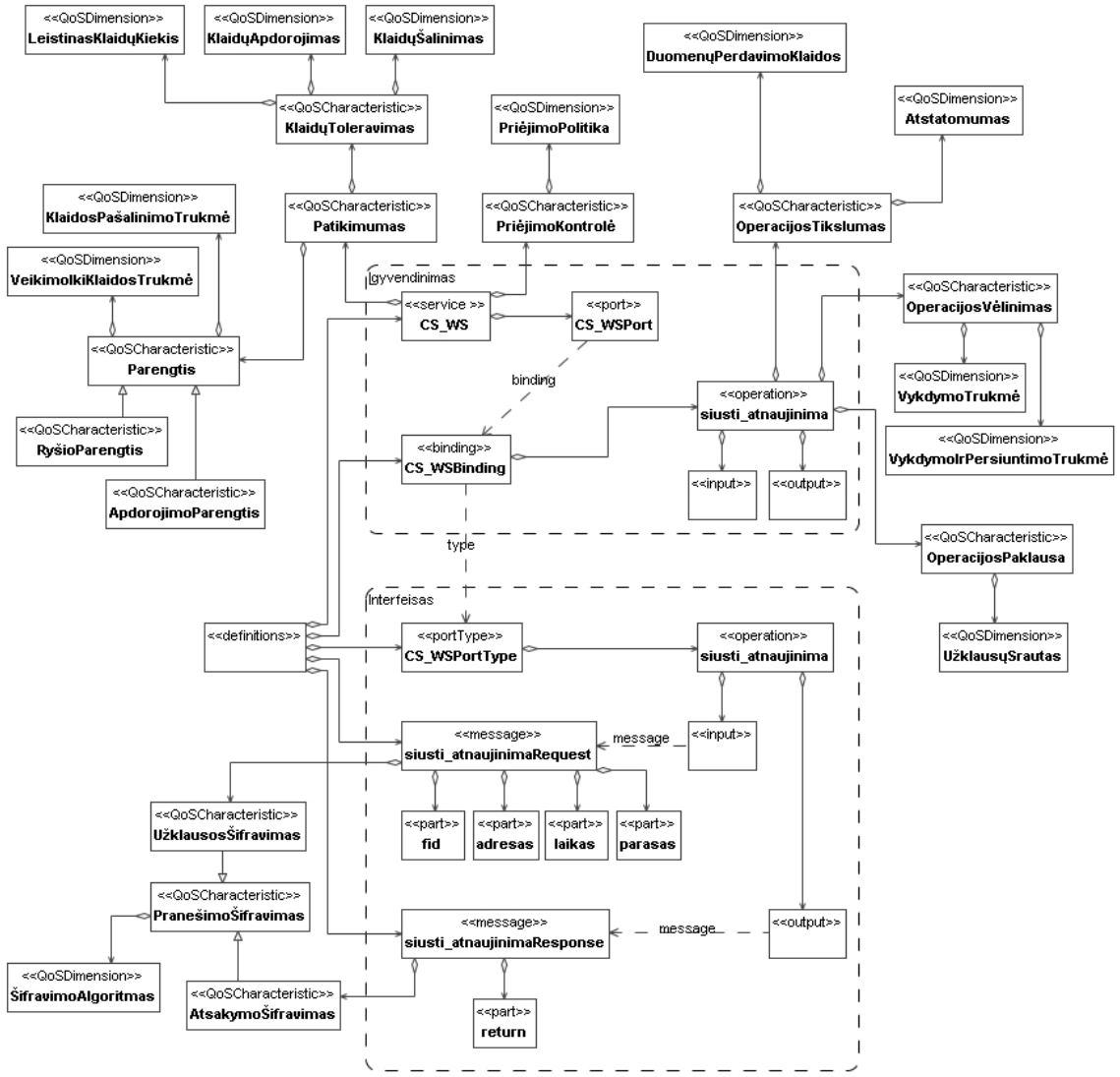


3 pav. Atnaujinimo vykdymo klientinėje sistemoje veiklos modelis

Įprastas WSDL aprašas skirtas specifikuoti, kokias operacijas galima kviesti bei kaip jas pasiekti, tačiau jame visiškai neatspindima, kokia paslaugų kokybė. Renkantis vieną iš keleto analogiškų paslaugų, kokybė gali būti pagrindinis pasirinkimo kriterijus. [5] šaltinyje buvo pasiūlyta išplėsti WSDL aprašą kokybinėmis tinklo paslaugų charakteristikomis. Remiantis šiame šaltinyje pateiktu WSDL aprašo išplėtimo metamodeliu bei paslaugų kokybės charakteristikų (angl. QoS characteristics) katalogu [13], centrinės sistemos tinklo paslaugos WSDL aprašo struktūrinė schema papildyta būtent tinklo paslaugoms svarbiomis kokybinėmis charakteristikomis (4 pav.**28 pav**).

Kiekviena kokybinė charakteristika turi stereotipą „QoSCharacteristic“ bei yra susijusi su vienu iš WSDL struktūrinės schemos elementų. Kiekvieną atominę charakteristiką apibrėžia viena ar daugiau dimensijų („QoSDimension“ stereotipas). Sudėtinės charakteristikos dekomponuojamos iki atominių. Toliau aprašoma kiekviena kokybinė tinklo paslaugų charakteristika bei ją apibrėžiančios dimensijos.

- *Patikimumas* (reliability) – sudėtinė charakteristika. Ji skaidoma į *parengties* bei *klaidų toleravimo* charakteristikas.
- *Parengtis* (availability) – tai paslaugos prieinamumas, t. y. būsena, kada paslauga yra prieinama ir gali būti naudojama. Šią charakteristiką specializuoja *ryšio parengtis*, nusakanti ryšio su serveriu stabilumą, bei *apdorojimo parengtis*, nusakanti paslaugos vykdymo serveryje stabilumą. Parengtį apibrėžia tokios dimensijos: *veikimo iki klaidos trukmė* (time between failure arba TBF) bei *klaidos pašalinimo trukmė* (time to repair arba TTR). Remiantis jomis galima rasti paslaugos prieinamumo santykinę vertę. Ji lygi TBF / (TBF + TTR) · 100 %.
- *Klaidų toleravimas* (fault tolerance) parodo, ar atsiradus klaidoms paslauga gali toliau būti prieinama ir vykdoma. Jį apibrėžia tokios dimensijos: *leistinas klaidų kiekis* (parodo, kiek klaidų gali įvykti, kol paslauga nebegalės būti vykdoma), *klaidų apdorojimas* (kaip sistemoje elgiamasi įvykus klaidoms) bei *klaidų šalinimas* (kokių priemonių imamasi klaidoms panaikinti).
- *Priėjimo kontrolė* (access control) nusako, kokiu būdu atrenkama, kas gali kviesti paslaugos operacijas. Atrinkimą apibrėžia šios charakteristikos dimensija *priėjimo politika*.



4 pav. Centrinės sistemos tinklo paslaugos aprašo su kokybinėmis charakteristikomis struktūrinė schema

- *Operacijos tikslumas* (operation accuracy) parodo, kiek gali skirtis gautas atsakymas nuo to, kurį tikimasi gauti. Tikslumą apibrėžia tokios dimensijos: *duomenų perdavimo klaidos* (kokia tikimybė, kad gali įvykti klaida duomenų perdavimo metu), *atstatomumas* (ar yra galimybė atstatyti sistemą į paskutinę stabilią būseną įvykus klaidai).
- *Operacijos vėlinimas* (operacion latency) parodo, kokia yra operacijos vykdymo trukmė. Šią charakteristiką apibrėžia tokios dimensijos: *vykdymo trukmė* (kiek laiko operacija vykdoma serveryje) bei *vykdymo ir persiuntimo trukmė* (turn-around time – koks laiko skirtumas tarp užklausos išsiuntimo ir atsakymo gavimo momentų).
- *Operacijos paklausa* (operation demand) nusako, ar dažnai operacija yra kviečiama. Ši charakteristika apibrėžiama dimensija *užklausų srautas* (užklausų kiekis per tam tikrą laiko tarpą).
- *Pranešimo šifravimas* (message encryption) skirtas siunčiamų pranešimų saugumui nusakyti. Jį specializuoja *užklausos šifravimas* bei *atsakymo šifravimas*. Šifravimą apibrėžia *šifravimo algoritmo* dimensija.

Kokybinių charakteristikų pateikimas gali tapti papildomu privalumu pasirašant sutartis su užsakovais. Taigi tinklo paslaugų aprašo išplėtimas kokybinėmis charakteristikomis svarbus ne tik įprastoje tinklo paslaugų architektūroje, bet ir šiame darbe sukurtam informacijos sistemų atnaujinimo modeliui. *Operacijos vėlinimo* dimensijų matavimas aprašytas 4 skyriuje.

Klientinės sistemos tinklo paslauga skirta centrinės sistemos pranešimo apie kodo fragmento atnaujinimą išsaugojimui. Tai vykdoma kreipiantis į operaciją „issaugoti_pranesima“. Operacijos įeinantys ir išeinantys parametrai yra identiški centrinės sistemos operacijai „siusti_atnaujinima“. Tinklo paslaugos struktūra analogiška centrinės sistemos tinklo paslaugos struktūrai.

3.4. Sistemų autentifikavimas

Saugiam duomenų perdavimui tarp centrinės ir klientinės sistemų būtinas užklausos siuntėjo autentifikavimas. Tokiu būdu yra išvengiama nesankcionuotų bandymų atsisiųsti kodo fragmentą iš centrinės sistemos ar nusiųsti netikrą pranešimą klientinei sistemai.

Siunčiamoje užklausoje vienas iš parametrų yra skaitmeninis parašas. Jis gaunamas savo privačiu raktu pasirašant teksto eilutę, sudarytą iš likusių parametrų pagal griežtą algoritmą. Pagal tą patį algoritmą iš gautų parametrų sudaroma teksto eilutė užklausą gavusioje sistemoje. Galiausiai remiantis šia eilute, gautu parašu bei užklausos siuntėjo viešu raktu nustatoma, ar siuntėjas yra atpažintas.

Pasirašoma tekstinė eilutė Y sudaroma sujungiant teksto blokus X_i, gautus iš kiekvieno į eilutę įtraukiamo parametro. Teksto blokas X_i gaunamas pagal tokią formulę:

$$X_i = \mathrm{p}(x_i)||x_i, \tag{1}$$

čia x_i – i-ojo parametro tekstinė išraiška, $\mathrm{p}(x_i)$ – to parametro ilgio triženklė išraiška, pridedant priekyje nulius, jei ilgio skaitinė vertė mažesnė už 100. Ženklas „||“ reiškia teksto bloko prijungimą. Taigi visa pasirašoma eilutė, kai parametrų kiekis lygus n, sudaroma taip:

$$Y = X_1||X_2||\ldots||X_n = \mathrm{p}(x_1)||x_1||\mathrm{p}(x_2)||x_2\,||\ldots||\mathrm{p}(x_n)||x_n. \tag{2}$$

Tiek centrinė, tiek klientinė sistema pasirašomas eilutes formuoja iš trijų parametrų: fragmento identifikatoriaus (*fid*), siunčiančios sistemos adreso (*adresas*) bei išsiuntimo laiko (*laikas*). Tarkime, *fid* = 123, *adresas* = „http://www.xyz.lt/ws_engine.php“, *laikas* = 1198368000 (sekundžių kiekis nuo 1970-ųjų metų). Tuomet p(*fid*) = „003”, p(*adresas*) = „031”, p(*laikas*) = „010”, o visa eilutė Y lygi „003123031http://www.xyz.lt/ws_engine.php0101198368000“.

Iš suformuotos eilutės saugiu maišos algoritmu SHA-1 gaunama eilutės santrauka. Ji pasirašoma privačiu raktu naudojant RSA algoritmą. Toks autentifikavimas vykdomas ne tik užklausose. Tokiu principu yra autentifikuojama ir centrinė sistema, atsakanti į klientinės sistemos užklausą. Tai daroma tam, kad būtų išvengta nesankcionuotų grąžinamų kodo fragmentų pakeitimų tarpiniuose mazguose tarp centrinės ir klientinės sistemų. Taip užtikrinamas visapusiškas perduodamų duomenų saugumas.

3.5. Perduodamų duomenų šifravimas

Kad perduodant duomenis iš vienos sistemos į kitą jie nebūtų nesankcionuotai nuskaitomi tarpiniuose mazguose, paprastai transportavimui naudojamas saugus protokolas HTTPS. Tačiau šiame darbe tai nebūtina, kadangi svarbiausi duomenys, t. y. programinio kodo fragmentai prieš siunčiant yra užšifruojami. Naudojamos tos pačios privačių ir viešų raktų poros, tiktai priešinga tvarka – užšifruojama viešu raktu, o dešifruojama privačiu.

RSA algoritmu negalima užšifruoti ilgesnių teksto eilučių nei pats privatus raktas (šiame darbe naudojami 1024 bitų raktai), o kodo fragmentai gali būti gerokai didesnės apimties. Todėl daromas tarpinis veiksmas – sugeneruojamas vienkartinis slaptas raktas, o kodo fragmentas užšifruojamas šiuo sugeneruotu raktu, naudojant paprastesnį šifravimo algoritmą (RC4). Tuomet sugeneruotasis raktas užšifruojamas gavėjo viešuoju raktu.

Gavėjui perduodamas ir užšifruotas kodo fragmentas, ir vienkartinis slaptas raktas. Pirmiausiai privačiu raktu dešifruojamas vienkartinis slaptas raktas. Tuomet juo dešifruojamas gautas kodo fragmentas. Tekstas, užšifravus jį gavėjo viešuoju raktu, tampa niekam nesuprantamas ir bereikšmis, išskyrus gavėją. Toks mechanizmas užtikrina, kad niekas kitas, išskyrus konkrečią klientinę sistemą, negalės tinkamai interpretuoti siunčiamo kodo fragmento.

3.6. Transakcijų valdymas

Kodo fragmento atnaujinimas klientinėje sistemoje vykdomas transakcijos principu. Jei bent vienas sudedamasis atnaujinimo proceso veiksmas neatliekamas arba atliekamas nesėkmingai, fragmentas neatnaujinamas.

Atnaujinimo procesą sudaro tokie sudedamieji veiksmai:

9) gaunamas centrinės sistemos atsakymas į atnaujinimo užklausą;
10) tikrinama, ar teisinga gauta duomenų struktūra;
11) autentifikuojamas atsakymo siuntėjas, t. y. centrinė sistema;
12) tikrinama, ar atsakyme nėra klaidos pranešimo;
13) dešifruojamas atsiųstas kodo fragmentas;
14) fragmentas išsaugomas tekstiniame faile;
15) sukuriamas naujos versijos įrašas duomenų bazėje su nuoroda į sukurtą failą;
16) pakeičiama fragmento įrašo duomenų bazėje nuoroda į naujausios versijos įrašą.

Paskutinis veiksmas negali būti vykdomas, jei neatlikti visi kiti, arba jei bent vieno iš jų atsakymas yra neigiamas (pavyzdžiui, neteisinga gauta duomenų struktūra). Tokiu būdu užtikrinama, kad esant tinklo sutrikimams,

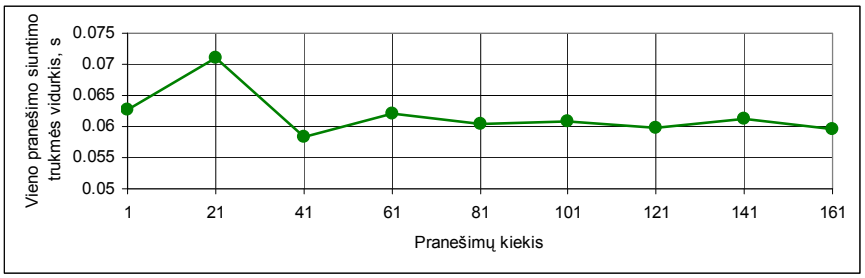
nesankcionuotiems bandymams pakeisti siunčiamus duomenis tarpiniuose mazguose arba sutrikimams pačiuose serveriuose, klientinės sistemos darbas nebus sutrikdytas.

4. Eksperimentinis paslaugų sistemos tyrimas

Šiame skyriuje aprašoma pagal sukurtą prototipą realizuotų pavyzdinių sistemų eksperimentinio tyrimo eiga bei rezultatai. Tyrime išskirtos trys atskiros dalys: pranešimų siuntimo trukmės, atnaujinimų vykdymo trukmės bei programinio kodo formavimo trukmės pokyčių tyrimai.

4.1. Pranešimų siuntimo klientinėms sistemoms trukmės tyrimas

Šio etapo tikslas – nustatyti atnaujinimo pranešimų siuntimo iš centrinės į klientinę sistemą trukmės priklausomybę nuo pranešimų kiekio. Ši priklausomybė reikalinga tam, kad, numatant didžiausią galimą klientinių sistemų, naudojančių vieną fragmentą, skaičių (t. y. didžiausią pranešimų apie vieną atnaujinimą gavėjų skaičių), būtų galima apskaičiuoti, koks pranešimų siuntimo kodo vykdymo laikas turi būti nustatytas centrinėje sistemoje. Pranešimo siuntimo trukme čia vadinamas laiko tarpas nuo pranešimo formavimo pradžios iki atsakymo gavimo iš klientinės sistemos.



5 pav. Vieno pranešimo siuntimo trukmės priklausomybė nuo pranešimų kiekio

5 pav. pateiktas vieno pranešimo siuntimo trukmės priklausomybės nuo pranešimų kiekio grafikas. Esant mažam pranešimų kiekiui bendra trukmė yra labiau paveikiama pašalinių procesų, todėl esama svyravimų. Tačiau matoma, kad pranešimo siuntimo trukmė nusistovi arti 0,06 s. Todėl galima teigti, kad pranešimų siuntimo klientinėms sistemoms trukmė tiesiškai priklauso nuo pranešimų kiekio. Šią priklausomybę galima išreikšti tokia formule:

$$t \approx t_0 \cdot N, \tag{3}$$

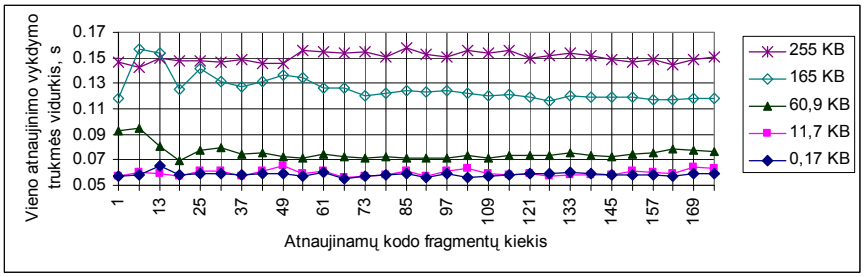
čia t – bendra trukmė, t_0 – vieno pranešimo siuntimo trukmė (šiuo atveju t_0 apytiksliai lygus 0,06 s), N – siunčiamų pranešimų kiekis.

Taigi numatant didžiausią galimą klientinių sistemų, naudojančių vieną fragmentą, kiekį N bei pridedant papildomą laiką t_{papild}, reikalingą įkelto kodo fragmento išsaugojimui centrinėje sistemoje, galima apskaičiuoti, koks turi būti nustatytas leistinas pranešimų siuntimo kodo vykdymo laikas t. Išmatavus t_{papild} paaiškėjo, kad šis įvertis lygus vos kelioms šimtosioms sekundės dalims net ir išsaugant 256 KB apimties kodo fragmentą. Taigi praktiškai kodo vykdymo trukmei apskaičiuoti galima taikyti 3 formulę. Ši trukmė atitiks klientinės sistemos tinklo paslaugos *operacijos vykdymo ir persiuntimo trukmės* dimensiją (4 pav.).

4.2. Atnaujinimų vykdymo klientinėse sistemose trukmės tyrimas

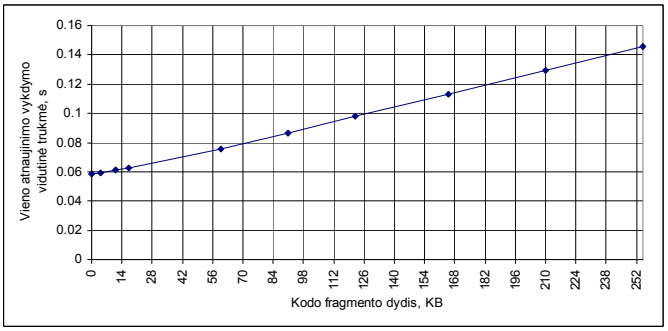
Šiame eksperimentinio tyrimo etape siekta nustatyti atnaujinimų vykdymo klientinėje sistemoje trukmės priklausomybes nuo atnaujinamų kodo fragmentų kiekio bei dydžio. Šios priklausomybės reikalingos tam, kad, numatant didžiausią galimą vienoje klientinėje sistemoje naudojamų kodo fragmentų skaičių bei didžiausią galimą fragmento dydį, būtų galima apskaičiuoti, koks fragmentų atnaujinimų kodo vykdymo laikas turi būti nustatytas klientinėje sistemoje. Į vieno fragmento atnaujinimo trukmę įeina laiko tarpas, skirtas užklausos centrinei sistemai formavimui, siuntimui, atsakymo gavimui, taip pat atsiųsto kodo fragmento išsaugojimui klientinėje sistemoje bei fragmento versijos pakeitimui.

6 pav. pateikta vieno atnaujinimo vykdymo trukmės vidurkio priklausomybė nuo atnaujinamų fragmentų kiekio. Didėjant fragmentų kiekiui vidutinė trukmė apytiksliai nusistovi arba svyruoja apie vieną ar kitą reikšmę. Taigi atnaujinimų vykdymo trukmė tiesiškai priklauso nuo atnaujinamų fragmentų kiekio.



6 pav. Vieno atnaujinimo vykdymo trukmės priklausomybės nuo atnaujinamų fragmentų kiekio palyginimas

Įvertinus šį tiesiškumą, atnaujinimo trukmės priklausomybės nuo atnaujinamo fragmento dydžio nustatymui galima naudoti vidutinius nusistovėjusius trukmės įverčius.



7 pav. Vieno atnaujinimo vykdymo trukmės priklausomybė nuo atnaujinamo fragmento dydžio

Vieno atnaujinimo vykdymo trukmės priklausomybė nuo kodo fragmento dydžio parodyta 7 pav. Matoma, kad vidutinė atnaujinimo trukmė nuo fragmento dydžio priklauso tiesiškai, tačiau net ir atnaujinant labai mažos apimties fragmentus, trukmė neartėja prie 0. Taip yra dėl to, kad atnaujinimo metu tarp centrinės ir klientinės sistemų yra apsikeičiama tinklo paslaugų užklausų antraštėmis bei kitais papildomais duomenimis. Susidaro papildomas pastovus dydis – trukmė t_{papild}. Taigi vieno atnaujinimo vykdymo trukmę t_0 galima apskaičiuoti pagal tokią formulę:

$$t_0 \approx t_1 \cdot L + t_{papild}, \tag{4}$$

čia L – atnaujinamo fragmento dydis kilobaitais, t_1 – vieno kilobaito apimties kodo fragmento atnaujinimo trukmė, nesant papildomos trukmės t_{papild}. Dydį t_1 galima rasti iš 7 pav. pateikto grafiko, padalinus didžiausią trukmę iš didžiausio fragmento dydžio.

Remiantis 3 bei 5 formulėmis, galima išvesti bendros atnaujinimų vykdymo trukmės t apskaičiavimui skirtą formulę:

$$t \approx (t_1 \cdot L + t_{papild}) \cdot N, \tag{5}$$

čia L – atnaujinamo fragmento dydis kilobaitais, t_1 – vieno kilobaito apimties kodo fragmento atnaujinimo trukmė, nesant papildomos trukmės t_{papild}, N – atnaujinamų fragmentų kiekis.

Taigi taikant 5 formulę galima apskaičiuoti didžiausią galimą atnaujinimo kodo vykdymo klientinėje sistemoje trukmę. Tam vietoje N reikia įstatyti didžiausią galimą klientinėje sistemoje naudojamų fragmentų kiekį N_{max}, o vietoje L – didžiausią galimą fragmento dydį L_{max}. Rasta trukmė atitiks centrinės sistemos tinklo paslaugos *operacijos vykdymo ir persiuntimo trukmės* dimensiją (4 pav.).

4.3. Programinio kodo formavimo klientinėse sistemose trukmės pokyčių tyrimas

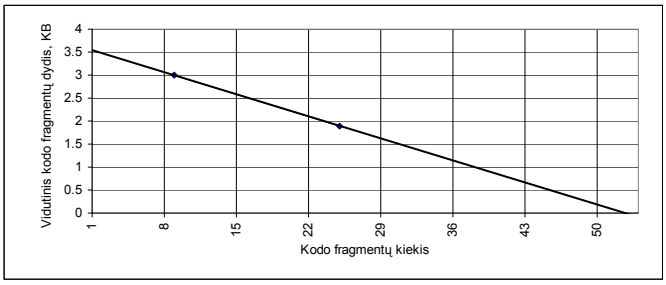
Šiame tyrimo etape siekta nustatyti, ar klientinių sistemų kodą formuojant pagal šiame darbe sukurto modelio principus tenkinamas greito HTML puslapių formavimo reikalavimas. Jo tinkamumo kriterijus – HTML puslapių formavimo trukmė neturi padidėti daugiau kaip 5 %, lyginant su įprastu kodo formavimu.

Siekta nustatyti pokyčių priklausomybes nuo fragmentų dydžio bei kiekio, kad būtų galima numatyti ribas, iki kurių tinkamumo kriterijus tenkinamas. Iškeltos dvi hipotezės:

1) kuo didesni kodo fragmentai, tuo santykinis kodo formavimo trukmės padidėjimas mažesnis;
2) kuo daugiau kodo fragmentų, tuo santykinis kodo formavimo trukmės padidėjimas mažesnis.

Pirmoji hipotezė logiškai kyla iš to, kad didesnių fragmentų įtraukimo trukmė yra didesnė nei mažesnių, todėl ji užima santykinai didesnę dalį nei duomenų bazės užklausos, kurios ir lemia matuojamą trukmės padidėjimą. Antrosios hipotezės atsiradimo priežastis yra tame, kad norint gauti naujausių kodo fragmentų versijų failų vardus stengiamasi į duomenų bazę kreiptis tik vieną kartą, todėl didesnis fragmentų kiekis sumažina kiekvieno iš jų failo vardo išgavimui skirtą trukmę.

8 pav. parodyta tiesė, kurios apatinėje dalyje esančios kodo fragmentų kiekio bei vidutinio dydžio kombinacijos aukščiau minėto tinkamumo kriterijaus netenkins, t. y. santykinis kodo formavimo trukmės padidėjimas bus didesnis nei 5 %. Pavyzdžiui, klientinėje sistemoje naudojant 36 fragmentus, kurių dydžio vidurkis 1 KB, tinkamumo kriterijus nebus tenkinamas, o naudojant tą patį kiekį 1,5 KB dydžio fragmentų kriterijus bus tenkinamas. Ši tiesė patvirtina abi iškeltas hipotezes.



8 pav. Greito HTML puslapių formavimo reikalavimo tinkamumo kriterijaus (formavimo trukmė neturi padidėti daugiau kaip 5 %) tenkinimo ribos

5. Išvados

Užsakovų informacijos sistemų atnaujinimas reikalauja didelių laiko sąnaudų ir yra nepatogus galutiniams vartotojams, nes sutrikdo jų darbą, todėl nuspręsta sukurti tinklo paslaugomis grindžiamą centralizuoto atnaujinimo modelį, kuris teigiamai paveiks sistemų kūrimo bei atnaujinimo procesą.

Atlikus egzistuojančių atnaujinimo sistemų, tinklo paslaugų standartų bei architektūros analizę nuspręsta, jog standartinė paslaugų architektūra netinka, tad atsisakyta UDDI registro. Taip sumažinamas paslaugų užklausų kiekis, vykdymo trukmė bei klaidų tikimybė.

WSDL modelio išplėtimas kokybinėmis charakteristikomis leidžia jį naudoti ne tik užklausoms generuoti, bet ir įvertinant paslaugų kokybę. Tokios charakteristikos gali būti papildomas privalumas derybose su užsakovu. Bendru atveju jos leistų paslaugų naudotojams lengviau išsirinkti vieną iš analogiškų paslaugų.

Tyrimas parodė, kad pranešimų siuntimo bei atnaujinimų vykdymo trukmė tiesiškai priklauso nuo pranešimų kiekio (pastaroji – ir nuo fragmentų dydžio). Tai leidžia lengvai apskaičiuoti didžiausią galimą pranešimų siuntimo ar atnaujinimų vykdymo trukmę.

Tyrimas taip pat parodė, kad puslapių formavimo užsakovų sistemose trukmė padidėja ne daugiau kaip 5 %, jei naudojamų fragmentų kiekis didesnis nei 50, arba dydžio vidurkis didesnis nei 3,5 KB. Kai fragmentų kiekis ir dydžio vidurkis mažesni, trukmės padidėjimas gali būti didesnis nei 5 %.

Šis tyrimas įgalins informacijos sistemas kuriančias organizacijas centralizuotai atnaujinti įdiegtas užsakovų sistemas. Tai sumažins jų administravimo laiką, atnaujinimas vyks nepertraukiant sistemų veikimo ir netrukdant galutiniams vartotojams.

6. Literatūra

[1]. **Alonso G., Casati F., Kuno H., Machiraju V.** Web Services: concepts, architectures and applications. *Berlin [etc.]: Springer*. 2004, 354 p.

[2]. **Box D., Ehnebuske D., Kakivaya G., Layman A. ir kt.** SOAP: Simple Object Access Protocol. [interaktyvus]. 2000, balandis [žiūrėta 2006-10-22]. Prieiga per internetą:http://static.userland.com/xmlRpcCom/soap/ SOAPv11.htm.

[3]. **Cerami E.** WSDL Essentials. [interaktyvus]. 2003, vasaris [žiūrėta 2006-10-22]. Prieiga per internetą: http://www.developer.com/services/article.php/1602051.

[4]. **Christensen E., Curbera F., Meredith G., Weerawarana S.** Web Services Description Language (WSDL) 1.1. [interaktyvus]. 2001, kovas [žiūrėta 2006-10-22]. Prieiga per internetą: http://www.w3.org/TR/wsdl.

[5]. **D'Ambrogio A.** A Model-driven WSDL Extension for Describing the QoS of Web Services// *IEEE International Conference on Web Services (ICWS'06): tarptautinės konferencijos pranešimų medžiaga [Chicago, 2006 m. rugsėjo 18-22 d.]*. Chicago, 2006, p. 789-796.

[6]. **David J. L.** Creating and Consuming Web Services With PHP. [interaktyvus]. 2004, kovas [žiūrėta 2007-06-20]. Prieiga per internetą: http://webservices.xml.com/pub/a/ws/2004/03/24/phpws.html.

[7]. **Erl T. – SOA Systems, Inc.** SOA: Principles of Service Design. *Indiana: RR Donnelley*. 2007, liepa, 608 p.

[8]. **Forbis.** IB Pay. Atsiskaitymai per FORPOST*Internet Banking. Informacija pardavėjams. [interaktyvus]. – [žiūrėta 2007-10-12]. Prieiga per internetą: http://www.sampo.lt/files/el.prekyba.ibpay.LT.pdf.

[9]. **He H.** What Is Service-Oriented Architecture. [interaktyvus]. 2003, rugsėjis [žiūrėta 2006-10-16]. Prieiga per internetą: http://www.xml.com/pub/a/ws/2003/09/30/soa.html.

[10]. **Kodali R. R., JavaWorld.com.** What Is Service-Oriented Architecture. [interaktyvus]. 2005, birželis [žiūrėta 2006-10-16]. Prieiga per internetą: http://www.javaworld.com/javaworld/jw-06-2005/jw-0613-soa.html.

[11]. **Nichol S.** Introduction to NuSOAP. [interaktyvus]. 2004, lapkritis [žiūrėta 2007-06-20]. Prieiga per internetą: http://www.scottnichol.com/nusoapintro.htm.

[12]. **Nickull D. – Adobe Systems, Inc.** Service Oriented Architecture Whitepaper. [interaktyvus]. 2005, vasaris [žiūrėta 2006-10-16]. Prieiga per internetą:http://www.adobe.com/enterprise/pdfs/Services_Oriented_Architecture_from_Adobe.pdf.

[13]. **Object Management Group.** UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms. [interaktyvus]. 2006, gegužė [žiūrėta 2007-11-24]. Prieiga per internetą: http://www.omg.org/docs/formal/06-05-02.pdf.

[14]. **The OpenSSL Project.** OpenSSL dokumentacija. [interaktyvus]. – [žiūrėta 2007-10-12]. Prieiga per internetą: http://www.openssl.org/docs/apps/openssl.html.

[15]. **W3C Working Group.** Web Services Architecture. [interaktyvus]. 2004, vasaris [žiūrėta 2006-10-22]. Prieiga per internetą: http://www.w3.org/TR/ws-arch/.