KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Svajūnas Sasnauskas

MagicDraw įrankio VeTIS praplėtimas duomenų saugojimui nutolusiame serveryje

Magistro darbas

Darbo vadovas

prof. dr. R. Butleris

Konsultantas

dokt. E. Šinkevičius

Kaunas

2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Svajūnas Sasnauskas

MagicDraw įrankio VeTIS praplėtimas duomenų saugojimui nutolusiame serveryje

Magistro darbas

Recenzentas
doc. dr. V. Pilkauskas
2012-05-28

Vadovas
prof. dr. R. Butleris
2012-05-28

Konsultantas
dokt. E. Šinkevičius
2012-05-28

Atliko
IFM-0/1 gr. stud.
Svajūnas Sasnauskas
2012-05-28

Kaunas

2012

TURINYS

MagicDraw tool VeTIS expansion for data storage on the remote server

SUMMARY

In this master thesis developed and described WEB service based version control system for storing MagicDraw tool VeTIS projects.

In first section researched existing version control systems. Analyzed what solutions are used in version control systems for problems like: data concurrency, data merging and storage. After research was done, using existing solutions developed WEB service based version control system and expanded MagicDraw tool VeTIS user interface for functionality usage of new web service module.

The investigation section describes the developed WEB service efficiency, when client's accessing VeTIS projects on server simultaneously.

Key words: MagicDraw, VeTIS, version control system, web service

SANTRUMPŲ IR TERMINŲ ŽODYNAS

P2P (angl. Peer-to-Peer) – kompiuterinio tinklo struktūra, kai kiekvienas kompiuteris tinkle gali dirbti kliento ar serverio rėžimu. Tokia struktūra veikiančiam tinklui nereikalingas centrinis serveris.

OMG (angl. Object Management Group) – koncorsiumas nustatantis standartus paskirstytoms objektinėms sistemoms, kuris dabar orientuojasi į modeliavimo (programoms, sistemoms ir verslo procesams) ir modeliais paremtus standartus.

SBVR (angl. Semantics of Business Vocabulary and Business Rules) – OMG priimtas standartas yra pagrindas norint formalia ir detalia natūralia kalba aprašyti sudėtingas struktūras, tokias kaip verslas.

XML (angl. Extensible Markup Language) – yra žymejimo kalba, kuri apibrėžia taisyklių rinkinį dokumentams koduoti formatu kuris yra suprantamas žmonėms ir mašinoms.

XMI (angl. XML Metadata Interchange) – OMG standartas skirtas keistis meta duomenimis naudojantis XML formatu.

UML (angl. Unified Modeling Language) – standartizuota bendros paskirties modeliavimo kalba naudojama objektiškai orientuotos programinės įrangos inžinerijoje.

RLE (angl. Run-length encoding) – paprastas duomenų suspaudimas, pakeičiant dažnai pasikartojančius simbolius į kitus mažiau atminties užimančius simbolius.

WSDL (angl. Web Services Description Language) – XML standartu paremta kalba skirta aprašyti WEB serviso suteikiama funkcionalumą.

WEB servisas (angl. WEB service) – programinė įranga skirta bendrauti mašina į mašina per internetą.

SOAP (angl. Simple Object Access Protocol) – protokolo specifikacija keistis struktūriškai apibrėžta informacija kurią teikia WEB servisas.

JDBC – sąsaja leidžianti Java kalboje parašytai programai naudotis duomenų bazės serverio paslaugomis.

SQL (angl. Structured Query Language) – programavimo kalba skirta valdyti duomenis reliacinėse duomenų bazių valdymo sistemose.

ĮVADAS

VeTIS yra sukurtas Kauno technologijos universitete Informacinių sistemų katedroje. VeTIS − tai įrankis skirtas apibrėžti veiklos terminus ir veiklos taisykles (SBVR), naudojant ribotą natūralią kalbą ir transformuoti juos iš natūralios kalbos į SBVR XMI ir UML.

SBVR yra OMG specifikacija išreikšti verslo žinias kalboje, kuri pirmiausia būtų suprantama verslo žmonėms. Galima teigti, kad SBVR skirta padėti verslo žmonėms suprasti modelius neturint specialių žinių modeliavimo notacijose ar IT įgūdžių.

Dažniausiai naudojamos tekstinės specifikacijos, kad išreikšti veiklos terminus ir veiklos taisykles, o ne diagramas. Nors diagramos yra naudingos, kai reikia pamatyti kaip verslo sąvokos susiję, tačiau diagramos nelabai tinka apibrėžti veiklos terminus ir išreikšti taisykles. SBVR naudoja ribota natūralia kalbą (pvz. anglų) aprašant verslo specifikacijas [15].

Šiuo metu VeTIS turi galimybę išsaugoti aprašytus veiklos terminus ir veiklos taisykles lokaliai (aplinkoje kurioje programa dirba). Dirbant su šiuo įrankiu komandoje, nesuteikiama galimybė efektyviai ir greitai pasidalinti atliktais pakeitimais veiklos terminuose ir taisyklėse. Efektyviai dalintis pakeitimais komandiniame darbe dažniausiai naudojama versijų kontrolės sistemos, kurios leidžia efektyviai dirbti prie to pačio projekto keliems žmonėms.

Pagrindinis darbo tikslas būtų praplėsti VeTIS galimybe saugoti ir pasiekti projektus nutolusiame serveryje. Šiam tikslui įgyvendinti darbas susidės iš esamų versijavimo sistemų tyrimo, serverio modulio sukūrimo, VeTIS įrankio praplėtimo, kad naudotis sukurto serverio modulio funkcionalumu.

Naujam serverio moduliui iškelti pradiniai reikalavimai:

- Modulis turi veikti servisų pagrindu t.y. naudojant WSDL standartą. Prie serverio turi būti galimybę prisijungti klientine programa, nesvarbu kokia programavimo kalba ji būtų parašyta.
- Būtų galimybė prie projekto dirbti daugiau nei vienam žmogui vienu metu ir netrukdyti vienas kitam (t.y. išspręsti problemą kai modifikuojamas tas pats failas tuo pačiu metu) , bei įvertinti tų žmonių atliktus pakeitimus.

- Klientai turi turėti galimybę prieiti prie įvairių projekto versijų, t.y. fiksuoti failų pakeitimus.

Iškelti pradiniai reikalavimai VeTIS įrankio praplėtimui:

- Turi turėti galimybę prisijungti prie pasirinkto serverio;
- Turi turėti galimybę naujo projekto sukūrimui pasirinktame serveryje;
- Turi turėti galimybę duomenų sinchronizavimui (projekto saugojimui ir parsisiuntimui) su pasirinktu serveriu.

Visas darbas susideda iš kelėtos etapų, kurie yra aprašyti skyriais. Pirmajame šio darbo skyriuje yra analizuojamas esamos versijų kontrolės sistemos. Aiškinamasi kokie duomenų suliejimo ir duomenų saugojimo metodai naudojami. Apžvelgiama kaip sprendžiamos duomenų konkurencinio modifikavimo problemos. Taip pat aprašomas Myers *diff* algoritmo veikimo principas, kuris yra pagrindas daugeliui šiuolaikinių duomenų suliejimo algoritmų.

Antrajame skyriuje aprašoma šiame darbe projektuojama sistema, jos paskirtis ir funkcijos.

Trečiajame skyriuje apibrėžiami funkciniai ir nefunkciniai reikalavimai kurie susideda iš serverio dalies ir grafinės sąsajos.

Ketvirtajame skyriuje yra aprašomas serverio dalies projektas. Paminima kokie sprendimai yra naudojami. Pateikiama sistemos architektūra, duomenų bazės schema su saugomų duomenų aprašymu, UML diagramos.

Penktajame skyriuje aprašoma projekto realizacija, kuri susideda iš grafinės sąsajos ir serverio modulio. Pademonstruojama nauji grafinės sąsajos elementai, jų funkcionalumas. Supažindinama su grafinės sąsajos atsakais į įvarius vartotojo veiksmus.

Šeštajame skyriuje atliekamas tyrimas nustatyti kaip sukurtąjį serverio modulį įtakoja prisijungusių klientų skaičius.

1. VERSIJŲ KONTROLĖS SISTEMŲ ANALIZĖ

1.1. VERSIJŲ KONTROLĖS SISTEMOS

Versijų kontrolė (dar žinoma kaip revizijų kontrolė (angl. revision control)), šaltinio kontrolė (angl. source control) ar išeities kodo valdymas (angl. source code management) tai pakeitimų valdymas dokumentuose, programose ar kitokių kompiuteriuose saugomų duomenų [3]. Taip pat galima sutikti *versijų kontrolės* apibūdinimą kaip procesą, kuris apibūdina projekte atliktų pakeitimų išsaugojimą ir gavimą [1].

Versijų kontrolės sistemos dažniausiai naudojamos ten, kur prie projektų dirba ne vienas žmogus. Tokiose sistemose atsiranda problemų su projekto failų apsikeitimų tarp žmonių dirbančių prie vieno projekto, taip pat jose gali būti sunku ar visiškai neįmanoma pamatyti kokie pakeitimai buvo atliekami projekte laikui bėgant. Versijų kontrolės sistemose taip pat iškyla problema kai prie tų pačių duomenų tuo pačiu metu nori dirbti keli žmonės. Versijų kontrolės sistemos išsprendžia šias problemas, leidžia pamatyti atliktus pakeitimus, kas ir kada atliko pakeitimus, taip pat suteikia galimybę grįžti prie ankstesnės projekto versijos, bei turi mechanizmus, kaip išspręsti situacijas, kai prie projekto dirba keli žmonės tuo pačiu metu.

Egzistuoja kelios rūšys versijų kontrolės sistemų. Jos klasifikuojamos pagal tai, kokiu modeliu pagrįsta saugykla:

- Centralizuota versijų kontrolė (angl. Centralized version control) – tai klientas-serveris veikimo modeliu pagrįsta sistema. Joje Serveris tvarko centrinę duomenų bazę ir pagrindinę duomenų saugyklą, vartotojai dirba su duomenų kopija iš serverio lokalioje sistemoje ir padarytus pakeitimus įkelia į serverį.
- Paskirstyta revizijų kontrolė (angl. Distributed revision control) – tai P2P veikimo modeliu pagrįsta sistema, kurioje kiekvienas vartotojas turi veikiančia projekto kopija ir tarpusavyje keičiasi duomenų pataisymai (angl. patch).

Galima paminėti kelias egzistuojančias atviro kodo (angl. open-source) versijų kontrolės sistemas kaip CVS, SVN, „LibreSource“ yra centralizuotos versijų kontrolės sistemos, o GIT, Mercurial, Bazaar, Monotone, Codeville yra paskirstytos versijų kontrolės

sistemos. Egzistuoja ir daugiau įvairiausių versijų kontrolės sistemų, sąrašą galima rasti “Wikipedia“ [4] portale (sąrašas nėra pilnas).

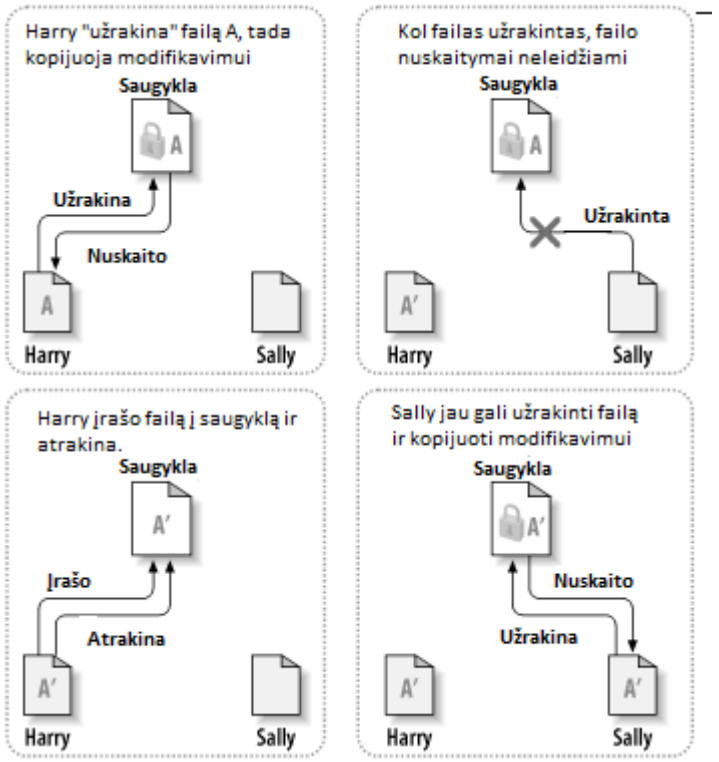
1.2. DUOMENŲ KONKURENCINIO MODIFIKAVIMO MODELIAI

Problemai kai keli vartotojai bando modifikuoti duomenis tuo pačiu metu spręsti versijų kontrolės sistemose yra naudojama keletas pagrindinių metodų. Pagrindiniai šie metodai tai užrakinti-modifikuoti-atrakinti (angl. Lock-Modify-Unlock), bei nukopijuoti-modifikuoti-sulieti (angl. Copy-Modify-Merge) [2].

Abu šie algoritmai turi privalumų ir trūkumų. Vienas didžiausias užrakinti-modifikuoti-atrakinti algoritmo trūkumas yra tas, kad jis užblokuoja duomenų modifikavimą kitiems vartotojams nors gal jie nori dirbti visai prie kitos duomenų dalies negu tas vartotojas, kuris užrakino duomenys. Taip pat gali būti tokia situacija, kad pamirštama nuimti užraktą [3], tokios situacijos labai mažina produktyvumą. Privalumas šio modelio tas, kad neiškyla problemų su duomenų vientisumu, kai failai priklauso vieni nuo kitų, nes dirbant keliems vartotojams vienu metu kas nors gali pakeisti projektą taip, kad paskui nebus galima sulieti į vientisus duomenis. Ši paminėta problema gali iškilti naudojant kopijuoti-modifikuoti-sulieti algoritmą. Tačiau praktikoje kopijuoti-modifikuoti-sulieti dažniausiai kur kas efektyvesnis, nes išspręsti konfliktus duomenyse užtrunka mažiau laiko negu prarandama blokuojant kitus vartotojus [3]. Konfliktai iškyla tada, kai versijų kontrolės sistemose naudojami duomenų suliejimo algoritmai negali tai padaryti automatiškai, iškilusius konfliktus reikia išspręsti rankiniu būdu. Skyreliuose 1.2.1 ir 1.2.2 apžvelgsime šių dviejų modelių veikimo principus.

1.2.1. UŽRAKINTI-MODIFIKUOTI-ATRAKINTI

Užrakinti-modifikuoti-atrakinti metodas leidžia duomenis modifikuoti tik vienam vartotojui tuo pačiu metu. Kad būtų galima modifikuoti duomenis jie pirma turi būti užrakinti. Jeigu duomenys užrakinti, o vartotojai juos nori modifikuoti, jie gali tik peržiūrėti ir laukti kada užraktas bus nuimtas. Grafiškai šį modelį galime matyti paveikslėlyje (1 pav.).

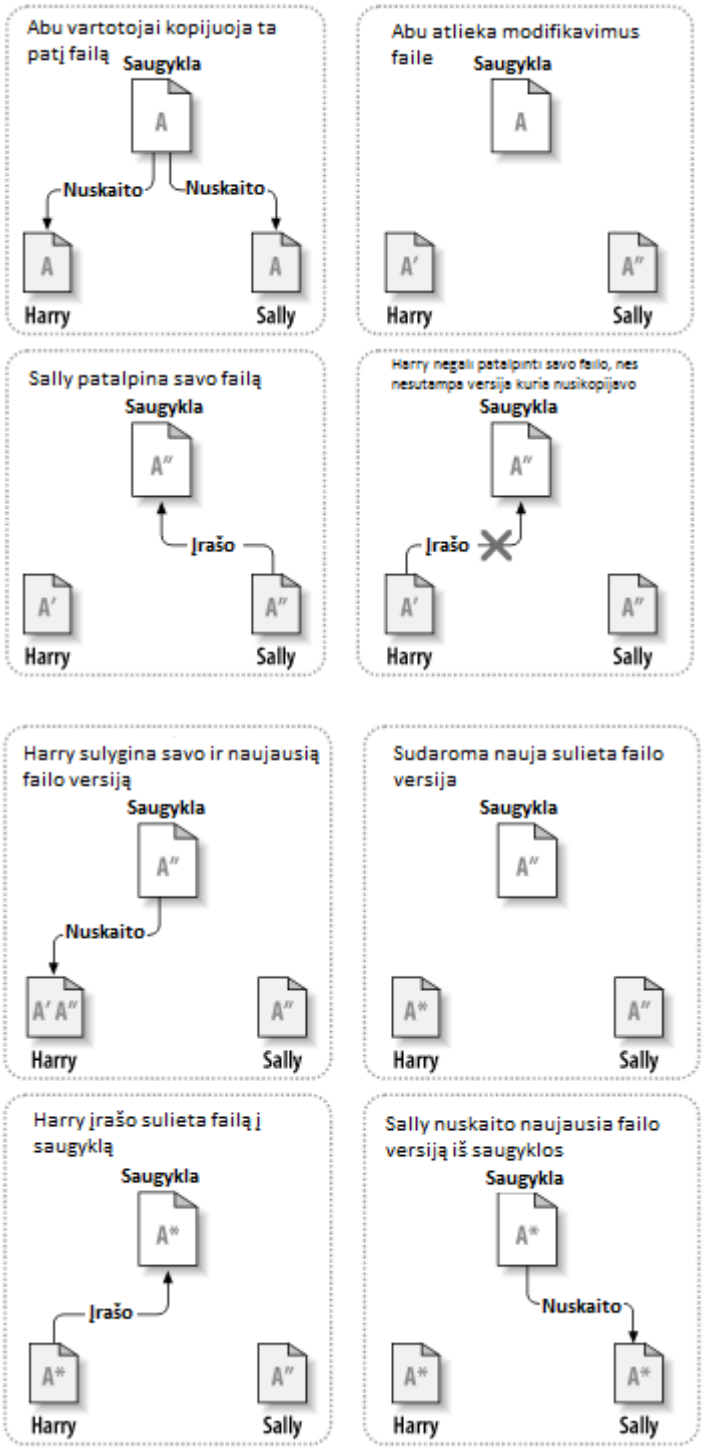


Pav. 1. Užrakinti-modifikuoti-atrakinti metodo modelis [3]

Harry užrakina duomenys A saugykloje ir juos nuskaito, Sally norėdama modifikuoti duomenis bando juos užrakinti, bet negali nes jie jau yra užrakinti Harry. Harry modifikuotus duomenis A‘ įrašo atgal į saugyklą ir atrakina, tuomet Sally gali užrakinti duomenis ir juos modifikuoti.

1.2.2. KOPIJUOTI-MODIFIKUOTI-SULIETI

Kopijuoti-modifikuoti-sulieti algoritmas leidžia modifikuoti duomenis tuo pačiu metu daugiau negu vienam žmogui. Vartotojas atsisiunčia visus duomenis iš saugyklos, modifikavęs įkelia atgal į saugyklą. Jeigu įkėlimo metu nesutampą duomenų versija su ta, kurią vartotojas atsisiuntė modifikavimui, tuomet atsiunčiama nauja versija iš saugyklos ir suliejama su vartotojo pakeitimais ir vėl bandoma įkelti į saugyklą. Grafiškai šis modelis atvaizduotas paveikslėlyje (2 pav.).



Pav. 2. Kopijuoti-modifikuoti-sulieti metodo modelis [3]

Harry ir Sally nuskaito duomenis A iš saugyklos į savo darbo aplinkas. Harry modifikuoja turimus duomenys ir taip gaunama nauja duomenų versija A‘, kuri saugoma Harry sistemoje. Sally taip pat modifikuoja duomenis savo sistemoje ir turi versiją A‘‘. Sally pirma įkelia savo duomenų versiją į saugyklą. Taigi dabar saugykloje yra duomenų versija A‘‘. Harry bando įkelti savo pakeitimus į saugyklą, bet negali, nes nesutampa duomenų

versija, kurią jis siuntėsi modifikavimui, kad būtų galima sulieti, todėl Harry atsisiunčia naujausia duomenų versiją (tai ką pakeitus įkėlė Sally į saugyklą) į savo darbo aplinką ir sulieja su savais pakeitimais taip gaunama nauja duomenų versija A*. Po suliejimoHarry gali įkelti šiuos duomenis su visais pakeitimais atgal į saugyklą. Sally atsisiunčia naujausią duomenų versiją A* iš saugyklos ir gali matyti visus savo ir Harry atliktus pakeitimus [12].

1.3. DUOMENŲ SULIEJIMO STRATEGIJOS

Visos versijų kontrolės sistemos naudoja kokia nors strategiją sulieti šakoms (angl. branches) ar failams, vienos sistemos leidžia naudoti išorinius trečiųjų šalių (angl. 3rd party) įrankius, kitos naudoja vidinius algoritmus ar abiejų variantų kombinacija.

Duomenims sulieti yra naudojama keletas versijų kontrolės sistemų, pavyzdžiui:

- CVS ir SVN versijų kontrolės sistemose tekstiniams duomenis sulieti naudojamas diff3 išorinis įrankis (GNU Diffutils sudedamoji dalis).
- Mercurial įrankis naudoja vidinį klasikinį trijų pakopų suliejimo (angl. 3-way merge) algoritmą Jeigu iškyla konfliktų bando naudoti išorinį įrankį (jeigu jis yra įdiegtas) [6].
- Git duomenims sulieti turi kelias pasirenkamas strategijas: *resolve, recursive, octopus, ours, subtree* [7].
- Monotone turi vidinį trijų pakopų algoritmą, tačiau leidžia naudoti išorinius įrankius, jeigu iškyla konfliktinių situacijų suliejant duomenis.
- LibreSource duomenų suliejimui naudoja LibreSource Synchronizer kuris turi vidinį algoritmą paremta transformacijomis.

1.3.1. DUOMENŲ SULIEJIMO ĮRANKIAI

Yra įvairiausių duomenų suliejimo įrankių, vieni įrankiai turi grafines vartotojo sąsajas, kiti valdomi tik komandine eilute, treti valdomi internetine sąsaja arba visai neturintys vartotojo sąsajos (programos bibliotekos/įskiepiai). Duomenų suliejimo problemai spręsti plačiausiai naudojami dviejų ir trijų pakopų suliejimo algoritmai, tačiau galime sutikti įrankių kurie naudoja ir kitokius algoritmus, kaip antai paremtus transformacija ar dviejų/trijų pakopų suliejimo algoritmų modifikacijom (pvz. Codeville naudoja dviejų pakopų suliejimą su istorine informacija).

Keletas skirtingas vartotojo sąsajas turinčių įrankių naudojamų versijų kontrolės sistemose duomenims sulieti:

- Diffutils [18] – tik komandine eilute valdomas įrankis, naudojantis dviejų ir trijų pakopų duomenų suliejimo algoritmus.
- LibreSource Synchronizer [8] – grafinė vartotojo sąsają (per WebStart) turintis įrankis, kuris naudoja duomenų suliejimo algoritmą paremta transformavimu.
- Diffuse[9] – grafinę sąsaja turintis įrankis, naudojantis dviejų ir trijų pakopų duomenų suliejimo algoritmus.

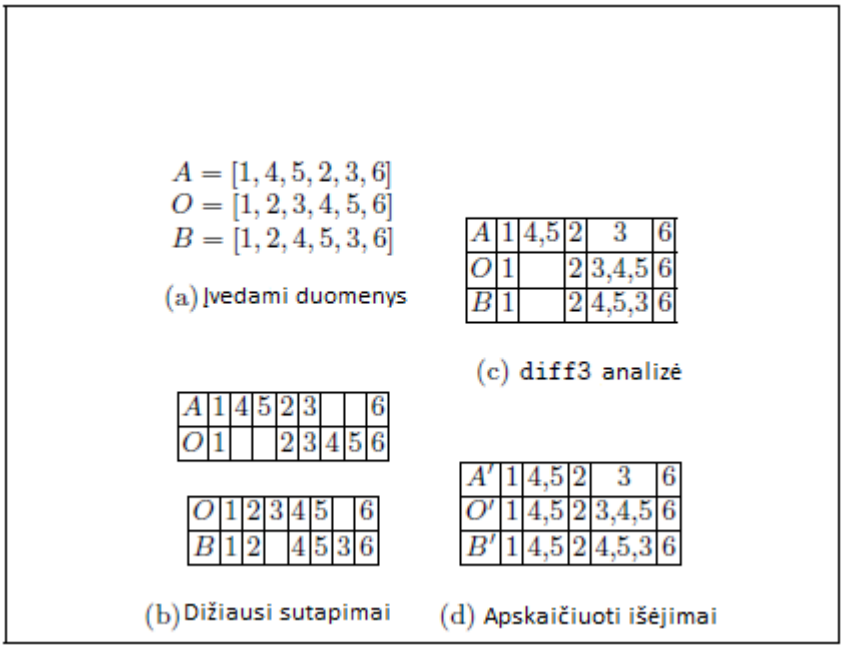
1.4. DUOMENŲ SULIEJIMO METODAI

1.4.1. DVIEJŲ PAKOPŲ METODAS

Dviejų pakopų metodas palygina dvi skirtingas duomenų versijas A ir B, kurios suliejamos į naują vieną versiją D. Šiam suliejimo būdui nereikia duomenų versijos iš kurios buvo nukopijuoti A ir B. Šis metodas turi trūkumų, vienas iš jų yra tai jog jis negali nustatyti ar elementai duomenyse A sukurti, ar duomenyse B ištrinti, o tai nutinka nes nesaugoma informacija iš kur kilo duomenys. Taip pat dviejų pakopų metodas negali nustatyti ar vienas elementas buvo modifikuotas abejuose versijose ar tik vienoje [10].

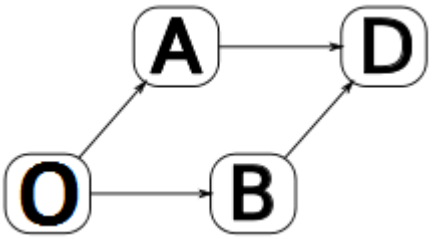
1.4.2. TRIJŲ PAKOPŲ METODAS

Šis metodas tikrina tris failus tarpusavyje (vienas failas yra pradiniai duomenys, o ir kiti du failai A ir B yra modifikacijos iš pradinio failo). Kiekviena modifikacija lyginama su pradiniu failu ir kur nėra persidengusių pasikeitimų sujungiama automatiškai. Jeigu šių failų negali sujungti išmetamas pranešimas operatoriui, kuris nusprendžia kokie pakeitimai bus atliekami. Trijų pakopų metodas neturi trūkumų, kurie buvo paminėti dviejų pakopų metode, nes naudojama duomenų versija iš kurios kilę abu A ir B failai [10]. Metodo veikimo pavyzdį atlikta su diff3 įrankiu galima matyti paveikslėlyje (3 pav.).



Pav. 3. Trijų pakopų suliejimas su Diff3 [5]

Pradinė duomenų versija yra O, kai A ir B yra modifikuotos iš O duomenų versijos. Duomenyse A buvo sukeista vietomis 4, 5 ir 2, 3, o duomenyse B 3 buvo perkeltas po 5. Pirmas dalykas ką diff3 įrankis daro − tai iššaukia dviejų pakopų algoritmu paremta palyginimo įrankį *diff*, kad surastų didžiausius sutapimus (arba ilgiausius bendrinius posekius) tarp O ir A, bei tarp O ir B, kaip parodyta paveikslėlio (b) dalyje (1 pav.). Tada sudaromi nauji blokai, kur O duomenų blokai skiriasi nuo A arba B. Taip pat sujungia esamus duomenų blokus į naujus blokus, kur elementai sutampa, ko pasekoje gaunamos sekos *stabilių* (kai duomenų blokuose ta pati informacija visuose failuose) ir *nestabilių* (kai viename ar abejuose failų duomenų blokuose informacija skiriasi) *gabalų* (autorius vadina juos *chunks* [5]) kaip parodyta paveikslėlio (c) dalyje (3 pav.). Galiausiai tikrinami naujai sudaryti *gabalai* ir nusprendžiama kokie pakeitimai turėtu būti atliekami, kaip parodyta paveikslėlio (d) dalyje (3 pav.). Antras *gabalas* buvo pakeistas tik duomenyse A (įterpiant 4,5), taigi šis pakeitimas gali būti taikomas B duomenims, bet ketvirtame *gabale* A ir B duomenyse skirtingi įrašai, taigi čia *diff3* nieko negali pritaikyti, nes yra konfliktinė situacija. Kur yra konfliktinės situacijos išmetamas pranešimas operatoriui, kuris ir nusprendžia koks pakeitimas turi būti atliekamas. Atlikus visus pakeitimus gaunama nauja duomenų versija D, duomenų srautai atvaizduoti paveikslėlyje (4 pav.).



Pav. 4. Trijų pakopų suliejimo duomenų srautai

1.5. MYER‘S DUOMENŲ SULIEJIMO ALGORITMAS

Šiame skyrelyje aprašomas Myers‘o *diff* algoritmas, remiantis Nicholat Butler straipsniu [16], kuriame vaizdžiai aprašomas algoritmo veikimas.

Eugene W. Myers 1986 m. parašytas straipsnis. „Algorithmica“ žurnale apie *diff* algoritmą yra pagrindas supratimui kaip veikia šiuolaikiniai duomenų suliejimo algoritmai. Myer‘s straipsnio pilnas pavadinimas yra „O(ND) skirtumų algoritmas ir jo variacijos“ (angl. „An O(ND) Difference Algorithm and Its Variations“).

Toliau aprašoma algoritmo apžvalga atlikta su simbolių palyginimu, bet algoritmas gali būti naudojamas bet kokiam duomenų tipui, kuriam galima naudoti lygybės operatorių.

1.5.1. NAUDOJAMI APIBRĖŽIMAI

A ir B failai, kurie naudojami algoritmo įėjime. Algoritmas sugeneruoja instrukcija kaip failą A paversti į B.

Trumpiausias redagavimo scenarijus (angl. Shortest Edit Script - SES) algoritmo surastas scenarijus, turi dvi komandas: trynimas iš A ir įterpimai į B. Šis scenarijus paverčia A failą į B failą.

Ilgiausias bendrinis posekis (angl. Longest Common Subsequence - LCS) – rasti SES yra tas pats kas rasti dviejų failų LCS. LCS yra ilgiausia simbolių seka, kuri gali būti sudaryta iš dviejų failų pašalinant kai kuriuos simbolius.

Kiekvienai failų pora, gali turėti keletą LCS. Pavyzdžiui, turint seką „ABC“ ir „ACB“, yra dvi LCS ilgiu 2: „AB“ ir „AC“. LCS tiesiogiai atitinka SES, taigi galima teigti, kad gali būti ir keletas SES tokiu pačiu ilgiu dviem failams. Algoritmas tiesiog gražina pirmą surasta LCS/SES.

Gyvatės - vienas trynimas ar įterpimas kai paskui eina nulis ar daugiau įstrižainių.

k linijos - linijos prasideda *x* ir *y* ašyse ir yra nubrėžtos įstrižai. Linija prasidedanti (0,0) koordinatėje nusakoma kaip *k=0, k>0* einant į dešine (x-ašimi) ir *k<0* einant žemyn (y-ašimi). Šios linijos yra nusakytos lygybe: *y=x-k*. Tai yra naudinga nes tada reikia tik saugoti x reikšmę konkrečiam taškui, duotoje *k* linijoje ir iš esamos formulės išskaičiuoti *y* reikšmę.

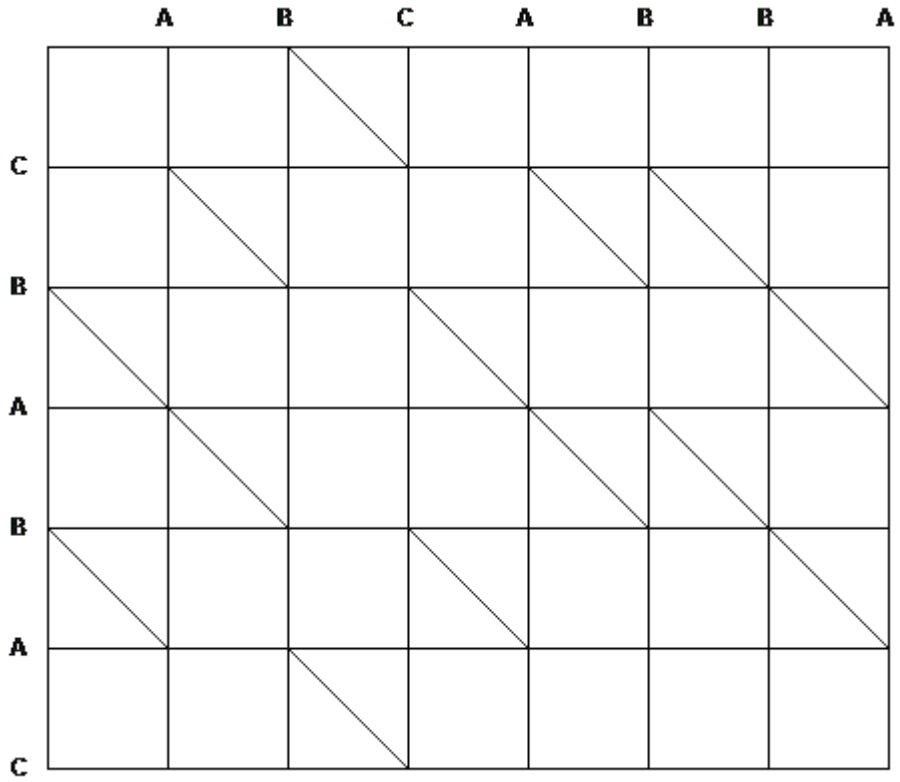
d kontūrai - sudaromas sujungus gyvačių pabaigos taškus su duota d reikšme, kurie parodo kiek skirtumų tarp failų.

1.5.2. ALGORITMO VEIKIMAS

1.5.2.1. Pavyzdiniai duomenys

Pavyzdžiuose duomenys naudojami tokie patys kaip ir Myer‘s straipsnyje. Failas A sudarytas iš „ABCABBA“, o failas B iš „CBABAC“ simbolių sekų. Šios sekos atitinka simbolių masyvus: A[] ir B[]. Masyvo ilgis A[] yra N, o masyvo B[] ilgis yra M.

Duomenys atvaizduoti grafiškai (5 pav.). A[] masyvo simboliai yra išdėstyti *x* ašyje (viršuje, iš kairės į dešine). B[] masyvo simboliai yra išdėstyti Y-ašyje (nuo viršaus į apačia).



Pav. 5. Pavyzdiniai duomenys [16]

Ieškant sprendimo reikia surasti trumpiausią kelią iš viršutinio kairiojo kampo (0,0) į apatinį dešinįjį kampą (7,6).

Galima judėti per viena simbolį horizontaliai arba vertikaliai. Horizontaliai (visada į dešinę) atitinka simbolio trynimą iš A[], o vertikalus judėjimas (visada žemyn) atitinka simbolio įterpimą į B[]. Jeigu yra simbolių sutapimas, tada galima judėti įstrižai (įstrižos linijos pažymėtos 5 pav.), sustojant sutapusios sekos pabaigoje.

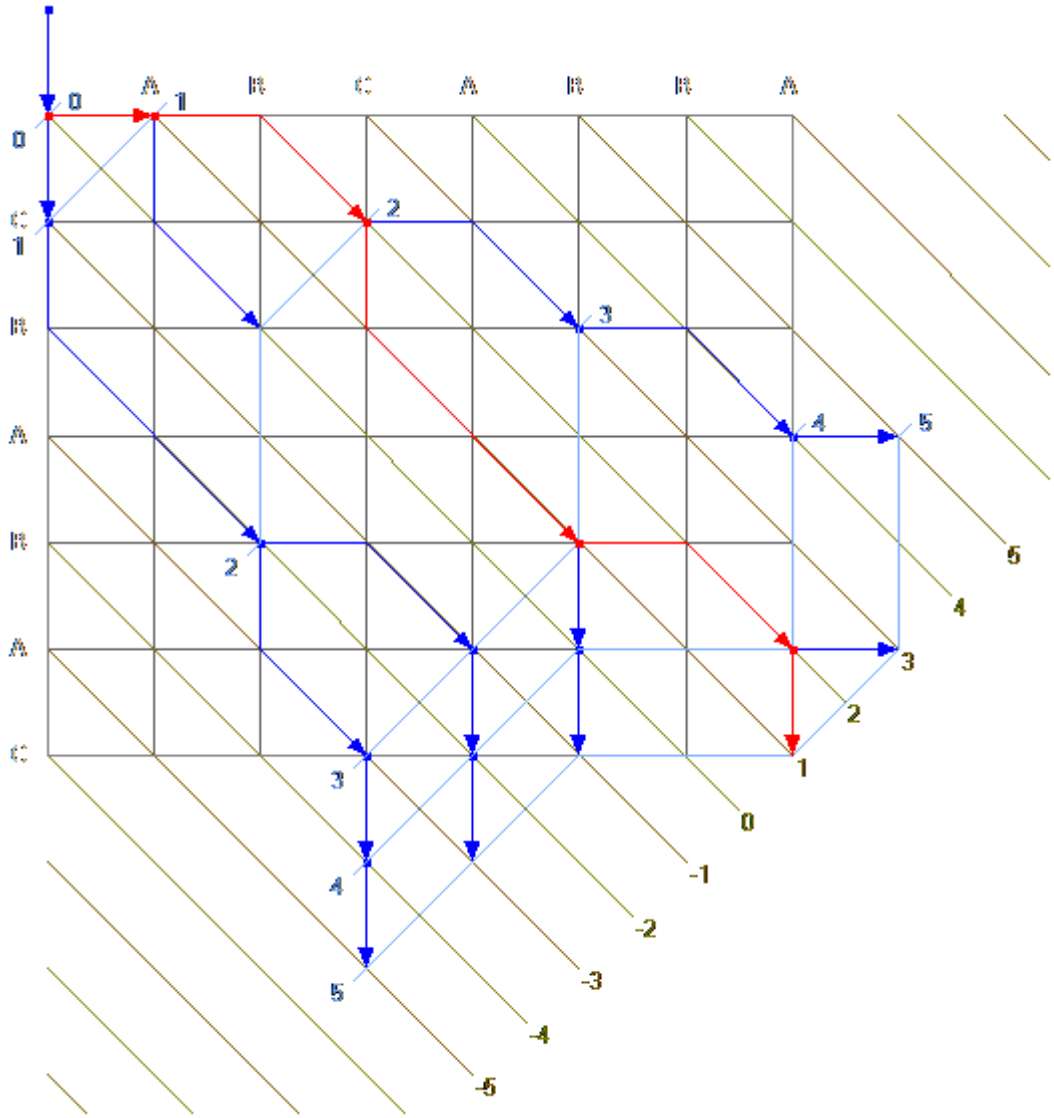
Galutinis sprendimas yra kelias kuris sudarytas iš kuo daugiau įstrižų ėjimų.

Šių duomenų LCS yra įstrižos kelio sekos, o SES yra horizontalios ir vertikalios kelio sekos. Esamam pavyzdžiui LCS yra 4 simbolių ilgumo, o SES yra 5 skirtumų ilgumo.

1.5.2.2. Godus algoritmas

Apžvelgsime standartini godųjį algoritmą (angl. greedy algorithm), be tiesinės erdvės patobulinimų. Pavyzdiniams duomenims sprendimas grafiškai atvaizduotas paveikslėlyje (6 pav.), kuriame matoma:

- k linijos − tai rudos linijos.
- gyvatės − tai tamsiai mėlyna spalva pažymėtos linijos, o raudona spalva išryškinta sprendinio gyvatė;
- d kontūrai - tai šviesiai mėlyna spalva žymimi skirtumo kontūrai.



Pav. 6. Grafinis sprendimo atvaizdavimas [16]

Algoritmas yra iteracinis. Jis apskaičiuoja tolimiausiai nukeliaujama kelią kiekvienai *k* linijai, kiekvienam nuosekliam *d*. Sprendimas randamas tada, kai kelias pasiekia apatinį dešinės pusės kampinę koordinatę. Pasiekus kampinę koordinatę garantuota, kad tai bus LCS/SES, nes jau buvo apskaičiuotos visos kelio galimybės su mažesniais *d*. Maksimali *d* reikšmė yra kada visai nėra sutapimų, kai N (trynimų) + M (įterpimų).

Ciklas pereiti per *d* reikšmes:

```
kol ( d = 0 ; d <= N + M ; d++ )
```

Vykdydami šį ciklą jo viduje, turime rasti toliausiai siekianti kelia kiekvienai *k linijai*. Duotojo *d, k linijos* kurios gali būti pasiekiamos yra ribose *[-d ... +d]*. *k < 0,* kai visi

ėjimai yra žemyn, o $k > 0$, kai visi ėjimai į dešinę. Svarbus pastebėjimas, kad pabaigos taškai lyginiams d yra ant lyginių *k* linijų ir atvirkščiai.

Vidinis ciklas pereiti per *k linijas* atrodytu taip:

```
kol ( k = -d ; k <= d ; k += 2 )
```

Paskutinė trūkstama dalis yra tolimiausiai siekiančio kelio radimas ant duotos *k linijos*.

Kad patekti ant *k linijos*, pirmiausia reikia eiti žemyn iš *k+1* arba dešinėn iš *k-1*. Jeigu mes išsaugosime tolimiausiai siekianti kelia iš prieš tai apskaičiuotų *d*, mes galime pasirinkti judėjimą, kuris nukels mus į tolimiausia tašką toje k linijoje. Du kraštiniai atvejai, kai $k=-d$, galima judėti tik žemyn ir kai $k=+d$ galima judėti tik dešinėn.

1.6. DUOMENŲ SAUGOJIMO MODELIAI

Versijų kontrolės sistemose failų saugykla ar duomenų bazė saugo pilną projekto istoriją. Saugoti šiuos duomenis su visais atliktais pakeitimais yra keletas modelių: snapshot, delta, weaves.

1.6.1. SNAPSHOT

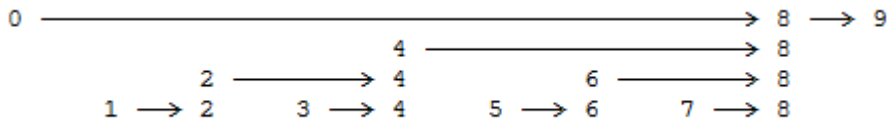
Kiekviena failo versija turi savo kopiją, o failų versijų kopijos nėra susijusios viena su kita [11], [14]. Kad sutaupyti vietos naudojamas elementarus duomenų suspaudimo algoritmas toks kaip RLE (angl. run length encoding) [11]. Tokio tipo saugykloje duomenų pasiekimo laikas yra konstanta, nes nepriklauso nuo kiek failo versijų saugojama, kiek projekte yra šakų ar koks duomenų dydis.

1.6.2. DELTA

Šio tipo saugyklose saugomi skirtumai tarp dviejų failo versijų. Taip sumažinama reikalinga vieta saugoti duomenims. Laikas gauti norima dokumento versija priklauso nuo atliktų pakeitimų kiekio. Yra kelių rūšių metodikos:

- Priekinės deltos (angl. forward deltas) pakeitimai turi būti uždėti ant ankstesnės failo versijos, kad gauti sekančia versiją [11], [14].
- Atgalinės deltos (angl. reverse deltas) saugo pilną naujausio failo versijos kopiją. Kad gauti ankstesnę versija pakeitimai yra uždedami atbuline tvarka [11], [13].

- Praleidžiamos deltos (angl. skip deltas) stipriai apriboja reikiamo kiekio delta atstatyti seną dokumento versiją. Vietoje kad kiekvieną senesnę failo versiją saugoti kaip deltą su sekančia versiją, kai kurios deltos yra apskaičiuojamos su naujesnėmis failo versijomis (praleidžiant kai kurias failo versijas) [11], [14]. Pavyzdys paveikslėlyje (7 pav.) kur skaičius atitinka failo versiją, o rodyklė į kokia versiją apskaičiuotos deltos.



Pav. 7. Praleidžiamos deltos

1.6.3. WEAVES

Visos failo versijos laikomos viename faile, kuriame yra meta duomenys prijungti prie blokų (dažniausiai eilutės), kurie pasako kokiai versijai priklauso blokas. Bet kuri failo versija gaunama nuosekliai nuskaičius visą failą. Išsaugoti naujai failo versijai reikia saugykloje perrašyti visą dokumentą. Daugėjant failo versijų ilgėja ir laikas gauti norimą failo versiją, nes daugėja eilučių skaičius faile [13], [14].

1.7. VERSIJŲ KONTROLĖS SISTEMŲ PALYGINIMAS

Nuo 1.1 iki 1.6 skyriaus buvo apžvelgtos versijų kontrolės sistemos ir jose naudojami metodai iškylančiom problemom (duomenų suliejimas, saugojimas, modifikavimas) spręsti. Ankstesniuose skyreliuose buvo paminėta keletas versijų kontrolės sistemų, dabar palyginsime jų funkcionalumą tarpusavyje lentelėje (1 lentelė).

Lentelė 1. Versijų kontrolės sistemų palyginimas

	Saugyklos modelis	Konkurencinis modelis	Veikia Web serviso pagrindu*	Duomenų saugojimo modelis	Duomenų suliejimo algoritmas	Licenzija/Kaina	OS (Serveris)
CVS	Centralizuotas	KMS	Ne	Delta	Išorinis	GPL/Nemokama	Unix
SVN	Centralizuotas	KMS / UMA	Ne	Delta	Išorinis	Apache, BSD/Nemokama	Unix/Win
LibreSource	Centralizuotas	KMS	Ne	Delta (Operacijos)	Vidinis	GPL/Nemokama	JAVA palaikančios OS
GIT	Paskirstytas	KMS	Ne	Snapshot	Vidinis / Išorinis	GPL/Nemokama	Visos
Mercurial	Paskirstytas	KMS	Ne	Delta	Vidinis / Išorinis	GPL/Nemokama	Unix/Win
Bazaar	Paskirstytas	KMS	Ne	Delta	Vidinis / Išorinis	GPL/Nemokama	Unix/Win
Monotone	Paskirstytas	KMS	Ne	Snapshot/Delta	Vidinis / Išorinis	GPL/Nemokama	Unix/Win
Codeville	Paskirstytas	KMS	Ne	-	Vidinis	BSD/Nemokama	Python palaikančios OS
KMS - Kopijuoti-modifikuoti-sulieti, UMA - Užrakinti-modifikuoti-atrakinti							

* Įdiegiamas kaip WEB servisas nereikia įdiegti atskirai ir veikia WSDL pagrindu.

1.8. ANALIZĖS IŠVADOS

Apžvelgtose versijų kontrolės sistemose matome, kad yra naudojama keletas pagrindinių metodikų vartotojų konkurencijai, duomenų saugojimui, duomenų suliejimui ir kitoms problemoms spręsti.

Vartotojų konkurencijos problemai spręsti yra naudojamos dvi metodikos: kopijuoti-modifikuoti-sulieti ir užrakinti-modifikuoti-atrakinti.

Versijų kontrolės sistemose duomenų saugojimui dažniausiai sutinkamos delta paremtos duomenų saugyklos.

Duomenų suliejimui naudojami du pagrindiniai principai, tai yra dviejų arba trijų pakopų algoritmai.

Apžvelgus versijų kontrolės sistemų matome, kad serverinė programinė įranga turi būti įdiegta atskirai ir turi tik priedus į WEB serverius, kad palaikytų internetinę sąsają. Yra ir saugyklos principu veikiančių sistemų, viena iš jų yra GIT. GIT neturi serverinės dalies, nes duomenys saugomi viešose saugyklose, kaip tinkliniame diske (angl. network drive) ar per WEB serverį prieinama saugyklą.

2. SPECIFIKACIJA

2.1. PROJEKTUOJAMAS SERVERIO MODULIS

Projektuojamas serverio modulis, kuris suteiks galimybę VeTIS projektus saugoti nutolusiame serveryje. Kiekvienas projektas turės pakeitimų istorijas, o tai leis palengvinti veiklą darbo grupėms dirbančioms su VeTIS projektais. Taip pat modulio paslaugos bus prieinamos, ne vienai bet visoms klientinėms programoms, kurios gali naudotis WSDL sąsaja, taip nepririšant vartotojų prie konkrečios klientinės programos.

Šiuo metu rinkoje esančios versijų kontrolės sistemos serverio dalyje saugo duomenis failų sistemoje, o skirtumų skaičiavimą tarp duomenų versijų atlieka klientinės programos. Šis projektuojamas modulis duomenų skirtumų skaičiavimus atliks pats, kas leis vartotojams patiems pasirašyti elementarias klientines programas, kurios pasiektų modulio teikiamas paslaugas per WSDL sąsają. O visa tai leis nesirūpinti projektų duomenų integralumo serveryje užtikrinimu, pavyzdžiui kai yra atnaujinamas serverio modulis ar jo duomenų suliejimo algoritmai. Taip pat serverio modulis dirbs WEB serviso pagrindu ir naudos MySQL duomenų bazę, kas praktiškai leis jį naudoti bet kokiame WEB serveryje.

Projektuojama programa susidės iš WEB serviso realizuoto „Java“ programavimo kalba, bei klientinės programos dalies, kuri bus integruota į VeTIS, o tai leis naudotis šio serverio modulio paslaugomis.

2.2. PROJEKTUOJAMO SERVERIO MODULIO PASKIRTIS

Projektuojamas modulio pagrindinė paskirtis suteikti galimybę VeTIS projektus saugoti versijomis nutolusiame serveryje. Šis sprendimas leis prieiti prie projektų iš bet kurios pasaulio vietos pasinaudojant interneto prieigą (jei tai viešai publikuojamas serveris, kuriame veikia šis modulis). Šio serverio modulio teikiamos paslaugos yra pateikiamos naudojant WSDL sąsają. Prie serverio bus galima prisijungti naudojantis įvairiomis programavimo kalbomis suprogramuotais klientais (pavyzdžiui: PHP, Python, C++, Java ir kt.).

2.3. PROJEKTUOJAMO SERVERIO MODULIO FUNKCIJOS

Sukurtas serverio modulis leis klientams prieiti prie šių operacijų:

- Autentifikuotis – leis vartotojui identifikuotis sistemoje. Pasinaudojus vartotoju vardu ir slaptažodžiu, taip gaunamas unikalus leidimo numeris „passport“ kuriuo

naudojantis prieinama prie serverio operacijų (kur reikalinga vartotojo identifikacija).

- Sukurti projektą – leis vartotojui sukurti naują projektą serverio duomenų bazėje, kurioje bus saugomi vieno VeTIS projekto failų duomenys. Vienas projektas susidės iš trijų failų: taisyklių, žodyno ir žodyno antraštės.
- Užrakinti projektą – nurodytas projektas bus užrakinamas, o jo duomenis galės atnaujinti tik jį užrakinęs vartotojas. Jeigu projektas bandomas užrakinti, kai jis jau yra užrakintas kito vartotojo ar vartotojas neturi tam teisių projekto užrakinti savo vardu nepavyks.
- Atrakinti projektą – jeigu projektas yra užrakintas jis gali būti atrakintas tik to vartotojo, kuris ji užrakino.
- Nusiųsti projektą į serverį – VeTIS projekto duomenys nusiunčiami į serverį išsaugojimui nurodytame serverio projekte Projektas išsaugojamas serveryje kaip naujausią projekto versiją, o duomenys priimami tik iš to vartotojo, kuris užrakinęs projektą.
- Parsisiųsti projektą – jeigu vartotojas turi teisę peržiūrėti projektą, tai bus leidžiama atsisiųsti bet kurią nurodyto projekto versiją ir gauti visus keturis duomenų failus.

3. REIKALAVIMAI

3.1. FUNKCINIAI REIKALAVIMAI

Reikalavimai serverio moduliui:

1. Serveris turi suteikti galimybę prieiti prie teikiamų savo paslaugų išorinėms klientinėms programoms (toliau klientas) per WSDL sąsają.
2. Serveris turi suteikti galimybę klientui prieiti prie jo paslaugų. Tam turi būti reikalaujama vartotojo identifikacija.
3. Serveris turi suteikti galimybę klientui įrašyti, modifikuoti VeTIS projektų duomenys jei vartotojas turi tam reikalingas teises ir uždrausti priėjimą tiems, kurie jų neturi.
4. Serveris turi išsaugoti visus projekte atliktus pakeitimus ir žinoti, kas ir kada juos atliko.
5. Serveris turi suteikti galimybę klientui gauti, bet kuria projekto versiją.
6. Serveris turi leisti atlikti modifikacijas tik vienam projektui ir tik vienam klientui vienu metu.

VeTIS įrankio grafinės sąsajos praplėtimo reikalavimai:

1. Grafinė sąsaja turi suteikti galimybę prisijungti prie nurodyto serverio.
2. Grafinė sąsaja turi suteikti galimybę parsisiųsti, bet kurią norimą projekto versiją.
3. Grafinė sąsaja turi suteikti galimybę sukurti naują projektą serveryje.
4. Grafinė sąsaja turi suteikti galimybę administruoti projektą.

3.2. NEFUNKCINIAI REIKALAVIMAI

Reikalavimai serverio moduliui:

1. Serverio modulis turi veikti serverinėse sistemose, kur palaikomi WEB servisai parašyti „Java“ programavimo kalba.
2. Serverio modulis turi būti pasiekiamas per WSDL sąsają.

3. Serverio modulio naudojami duomenys turi būti saugomi MySQL duomenų bazėje.
4. Serverio modulis turi dirbti stabiliai. Įvertinti galimas klaidas, kaip antai: gavus klaidingus duomenys ar komandas, nestabdyti modulio darbo, o ignoruoti klaidas ir tęsti darbą toliau (nebent jos butų kritinės).
5. Serverio modulis turi būti pasiekiamas per internetą ar vietinį tinklą.
6. Serverio modulis turi užtikrinti stabilų darbą daugiau nei vienam klientui tuo pačiu metu.

VeTIS įrankio grafinės sąsajos praplėtimo reikalavimai:

1. Grafinė sąsaja turi integruotis į dabartine VeTIS grafinę sąsają.
2. Naujas funkcionalumas neturi trukdyti esamam funkcionalumui.
3. Klientinė programa turi dirbti stabiliai (įvedus klaidingus duomenys informuoti vartotoją ir leisti jas pataisyti).

4. SERVERIO MODULIO IR GRAFINĖS SĄSAJOS PROJEKTAS

4.1. NAUDOJAMI SPRENDIMAI

Pagrindinės problemos kuriant WEB servisą yra parinkti tokį duomenų suliejimo metodą, kad būtų išlaikytas duomenų vientisumas. Taip pat reikia išlaikyti kuo didesnę spartą su kuo mažesnėmis duomenų saugojimo sąnaudomis. Reikia įvertinti situaciją, kad turi būti apsaugota projektų priklausomybę vienas nuo kito, kai modifikuojamas vienas projektas, o jo duomenys susieti su kitais projektais.

Duomenų suliejimo metodas – duomenų skirtumams apskaičiuoti ir juos vėl sulieti naudojama *google-diff-match-patch* biblioteka [19]. Šios bibliotekos autorius yra Neil Fraser. Bibliotekos paskirtis surasti skirtumus tarp dviejų tekstinių duomenų, sudaryti delta variacijas. Naudojant šias deltas ir vieną iš tekstinių duomenų, sukonstruoti kitus tekstinius duomenys. Bibliotekos algoritmas remiąsi Myers diff algoritmo pagrindu (aprašytas 1.5 skyrelyje), kuris laikomas geriausiu bendros paskirties algoritmu. Šios bibliotekos duomenų formato pavyzdžius kurie išsaugojami serverio duomenų bazėje galima rasti 1 priede.

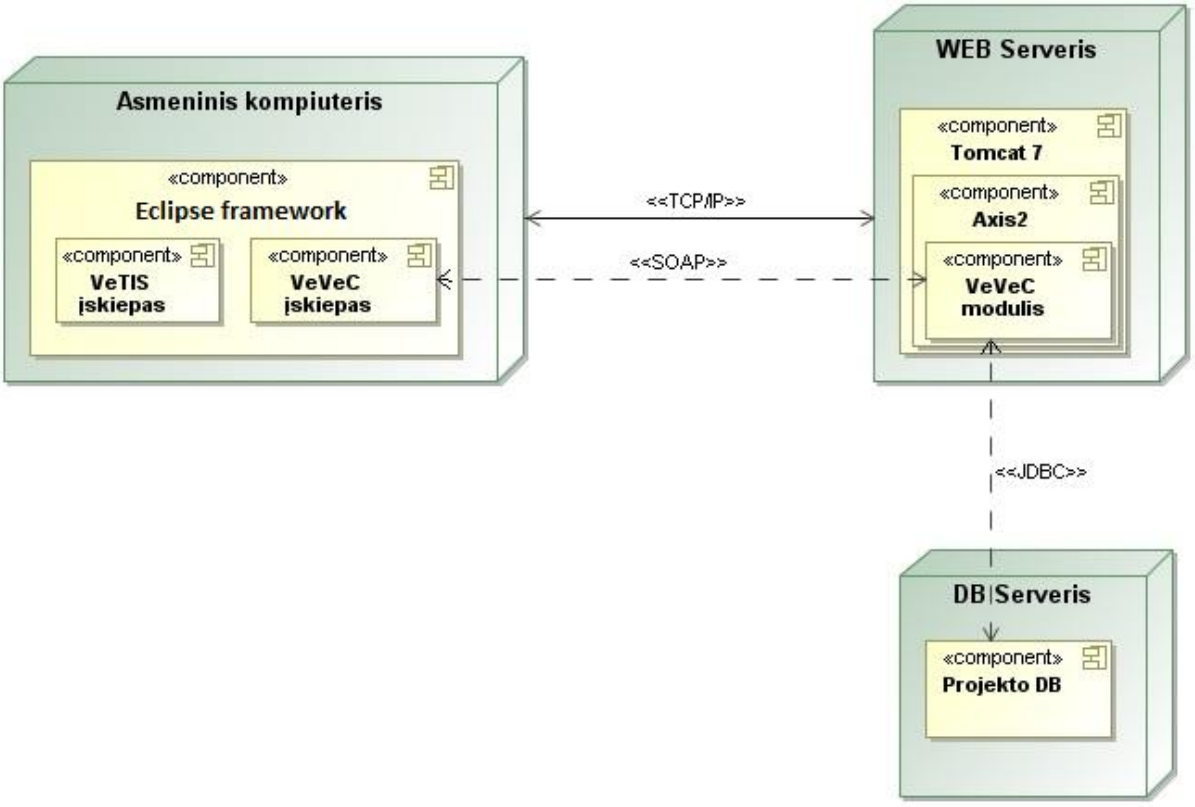
Duomenų transformavimas – objektų transformavimui į XML naudojama XStream biblioteka [20]. Jos pagalba bet koks objektas paverčiamas į XML struktūra ir gali būti vėl gražinamas į objektą. Taip transformavus duomenis supaprastina sudėtingų duomenų struktūrų apsikeitimą tarp kliento ir serverio.

4.2. BENDRA SISTEMOS ARCHITEKTŪRA

Visa sistema susideda iš trijų pagrindinių dalių: asmeninio kompiuterio, WEB serverio ir duomenų bazių serverio. Asmeniniame kompiuteryje įdiegta Eclipse aplinka su VeTIS ir VeVeC įskiepais. VeVeC plėtinys bendrauja su serverio VeVeC moduliu SOAP protokolu. Tarp jų duomenys keliauja XML formate.

Serverio VeVeC modulis atlieka tik skaičiavimus, o visus duomenys saugo duomenų bazių serveryje. Su duomenų baze susijungiama JDBC sąsaja, per kuria modulis įkelia ir pasiima duomenis naudodamasis SQL užklausomis.

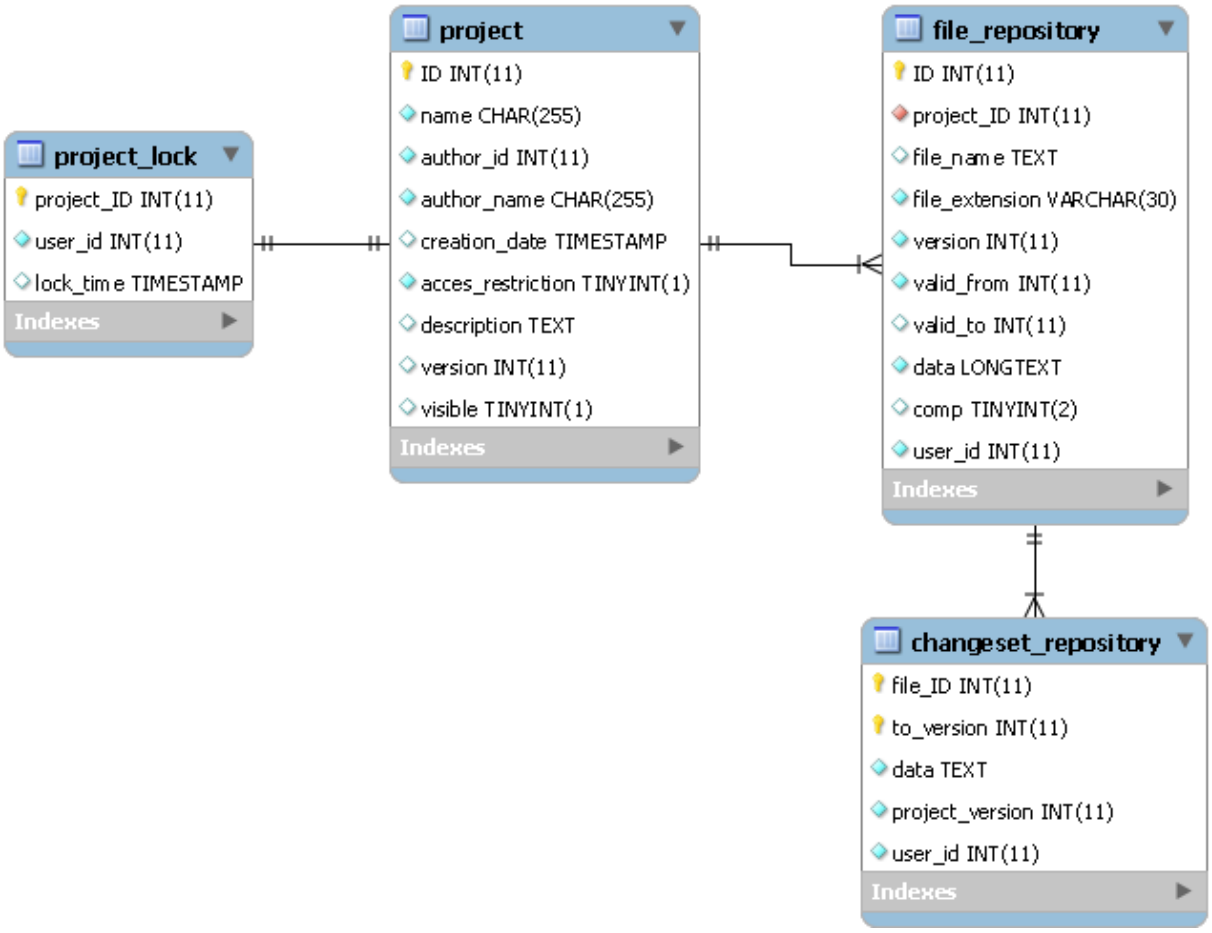
„*VeVeC įskiepas*“ yra projektuojama grafinė sąsaja su reikalinga logika naudotis „*VeVeC moduliu*“. „*VeVeC modulis*“ yra projektuojamas WEB servisas.



Pav. 8. Loginė sistemos architektūra

4.3. DUOMENŲ BAZĖS SCHEMA

Projektas turi vieną duomenų bazę, kurios schema pateikta paveikslėlyje (9 pav.) Minėtoje schemoje pateikti lentelių pavadinimai ir saugomi įrašai, bei atvaizduoti ryšiai tarp šių lentelių.



Pav. 9. „vetis_versioncontrol“ duomenų bazės schema

4.4. DUOMENŲ APRAŠYMAS

Duomenų bazę „vetis_versioncontrol“ naudosime saugoti VeTIS projektams, taip pat pakeitimų rinkiniams, o tai leis atkurti bet kurią projekto versiją.

Duomenų bazė sudaryta iš keturių lentelių „project_lock“, „project“, „file_repository“ ir „changeset_respository“. Lentelę „project“ sudaro tokie atributai:

- ID – esybės identifikavimui, jos išskyrimui iš kitų tos esybės objektų, vartojamas raktas, kuris vienareikšmiškai apibrėžia bet kurį esybės objektą (atributą);
- name – atributas saugojantis projekto pavadinimą, kuris yra suteiktas vartotojo.
- author_id – atributas saugojantis vartotojo identifikacinį numerį, kuris sukūrė tą projektą.
- author_name - atributas saugojantis vartotojo vardą, kuris sukūrė tą projektą.

- creation_date – atributas saugojantis kada projektas buvo sukurtas duomenų bazėje.
- access_restriction – atributas saugojantis projekto priėjimo leidimus.
- description – atributas saugojantis projekto apibūdinimą.
- version – atributas saugojantis įrašą, kuris nurodo kiek versijų yra saugoma.
- visible – atributas saugojantis įrašą, kuris nurodo ar projektas matomas projektų paieškose.

Lentelė „project_lock“ saugoja įrašus apie projektus kurie yra šiuo metu užrakinti. Ją sudaro tokie atributai:

- project_id – atributas saugojantis projekto identifikavimo numerį.
- user_id – atributas saugojantis vartotojo identifikacinį numerį, kuris užrakino projektą.
- lock_time - atributas saugojantis laiką kada projektas buvo užrakintas.

Lentelė „file_repository“, saugoja projektų failų duomenys. Ją sudaro tokie atributai:

- ID – esybės identifikavimui, jos išskyrimui iš kitų tos esybės objektų, vartojamas raktas, kuris vienareikšmiškai apibrėš, bet kurį esybės objektą (atributą);
- project_id – atributas saugojantis ID numeri iš „project“ lentelės, kad identifikuoti kuriam projektui šis įrašas priklauso.
- file_name – atributas saugojantis failo vardą.
- file_extension – atributas saugojantis failo plėtinį.
- version – atributas saugojantis failo versijos numerį.
- valid_from - atributas saugojantis įrašą nuo kurios projekto versijos jis priklauso projektui.

- valid_to - atributas saugojantis įrašą iki kurios projekto versijos jis priklauso projektui.
- data – atributas saugojantis failo duomenys.
- comp – atributas saugojantis įrašą, kuris nusako ar „data“ lauko įrašui yra panaudota kompresija.
- user_id - atributas saugojantis įrašą, vartotojo id, kuris įrašė failo duomenis.

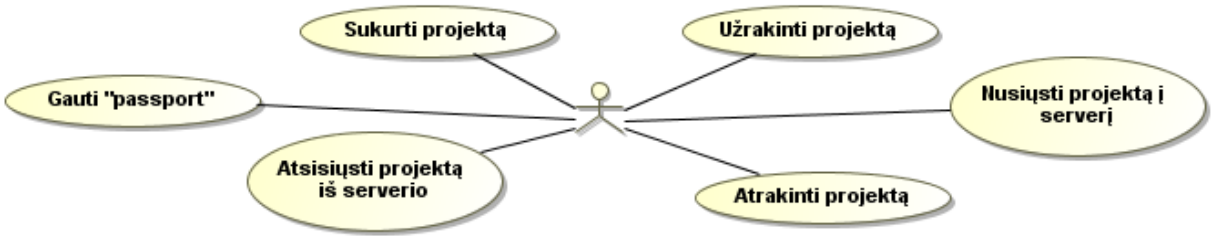
Lentelė „changeset_repository“, saugoja projektų failų skirtumų delta rinkinius. Ją sudaro tokie atributai:

- file_ID – esybės identifikavimui, jos išskyrimui iš kitų tos esybės objektų, vartojamas raktas, kuris apibrėš bet kurį esybės objektą (atributą). Šis įrašas sutampa su ID įrašais iš „file_repository“ kas nusako jų tarpusavio ryšį.
- to_version – atributas saugojantis įrašą į kokią versiją delta keičią failą.
- data – atributas saugojantis failo deltas.
- project_version – atributas saugojantis įrašą, kuris nusako kokiai projekto versijai priklauso failas, sudarytas iš saugojamų deltų.
- user_id - atributas saugojantis įrašą, kuris nusako vartotojo id kuriam priklauso sudarytas failas.

4.5. SERVERIO MODULIO IR GRAFINĖS SĄSAJOS UML DIAGRAMOS

4.5.1. PANAUDOJIMO ATVEJAI

Pagrindiniai veiksmai kuriuos gali atlikti vartotojas grafinėje aplinkoje, kad pasinaudoti WEB serverio paslaugomis kurios aprašytos 2.3 skyrelyje.



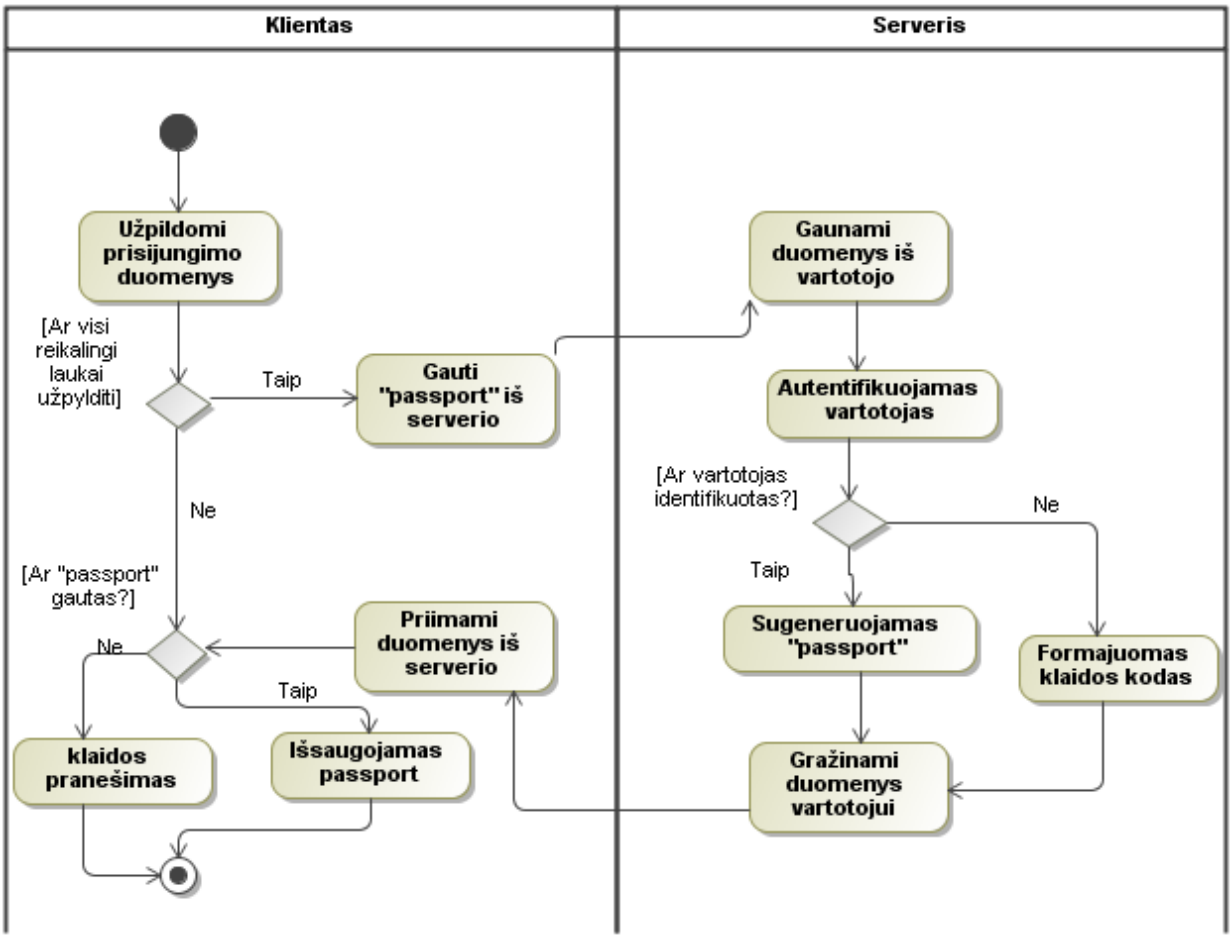
Pav. 10. Panaudojimo atvejai

4.5.2. VEIKLOS DIAGRAMOS

Veiklos diagramos pateiktos paveikslėliuose (11-16 pav.) kiekvienam panaudojimo atvejui kuris buvo pateiktas ankstesniojo skyrelio paveikslėlyje (11 pav.).

Gauti „passport“

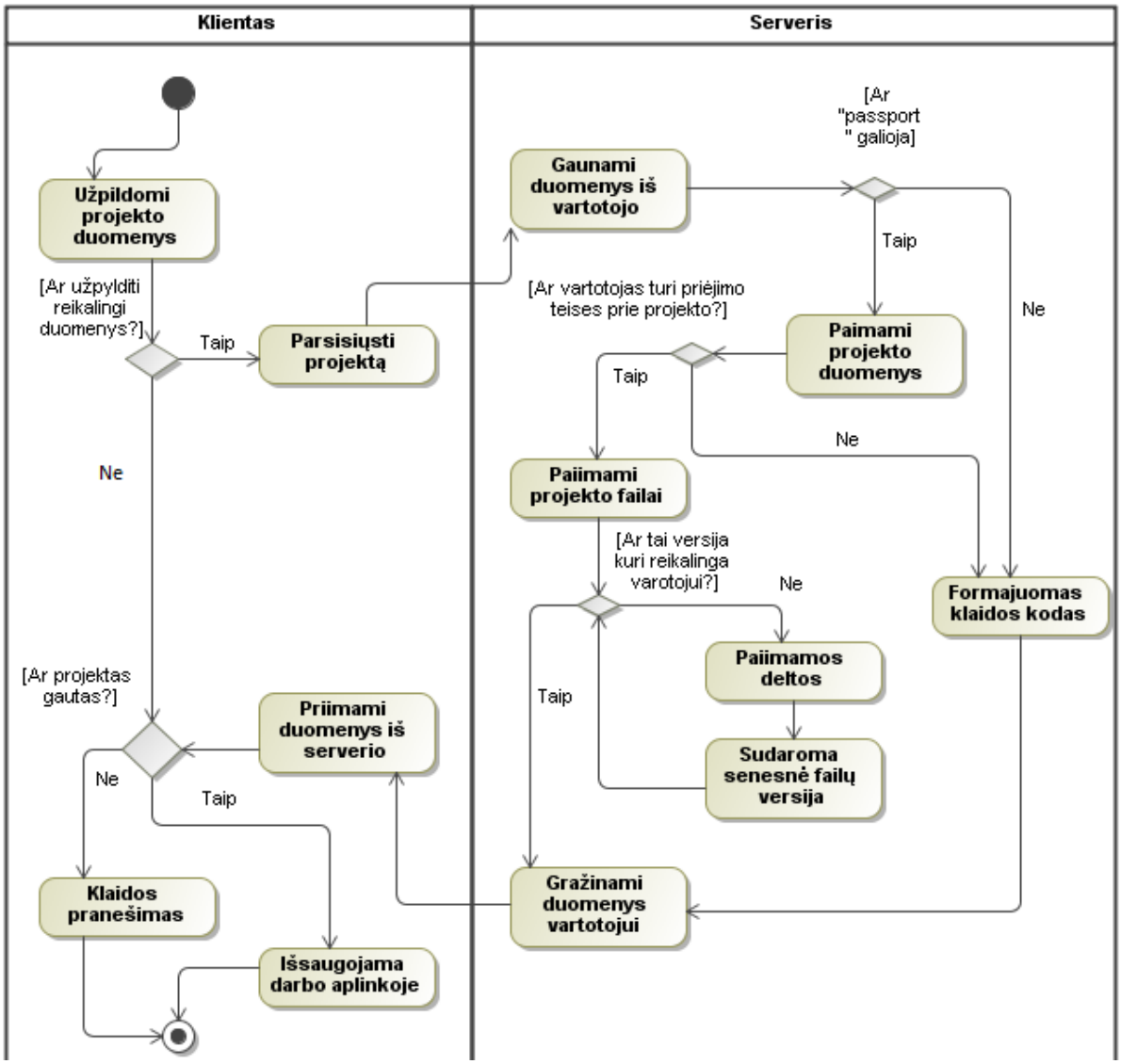
Veiklos diagrama pavaizduota paveikslėlyje (11 pav.) parodo vykdoma veiksmų seka, nuo to laiko kai vartotojas sužadina veiksmą „*gauti passport*“. Šis veiksmas atliekamas kai vartotojas nėra prisijungęs prie serverio ir norima naudotis jo paslaugomis, bet prieiti prie paslaugų reikalingas „*passport*“.



Pav. 11. Veiklos diagrama panaudojimo atvejui „*Gauti passport*“

Atsisiųsti projektą iš serverio

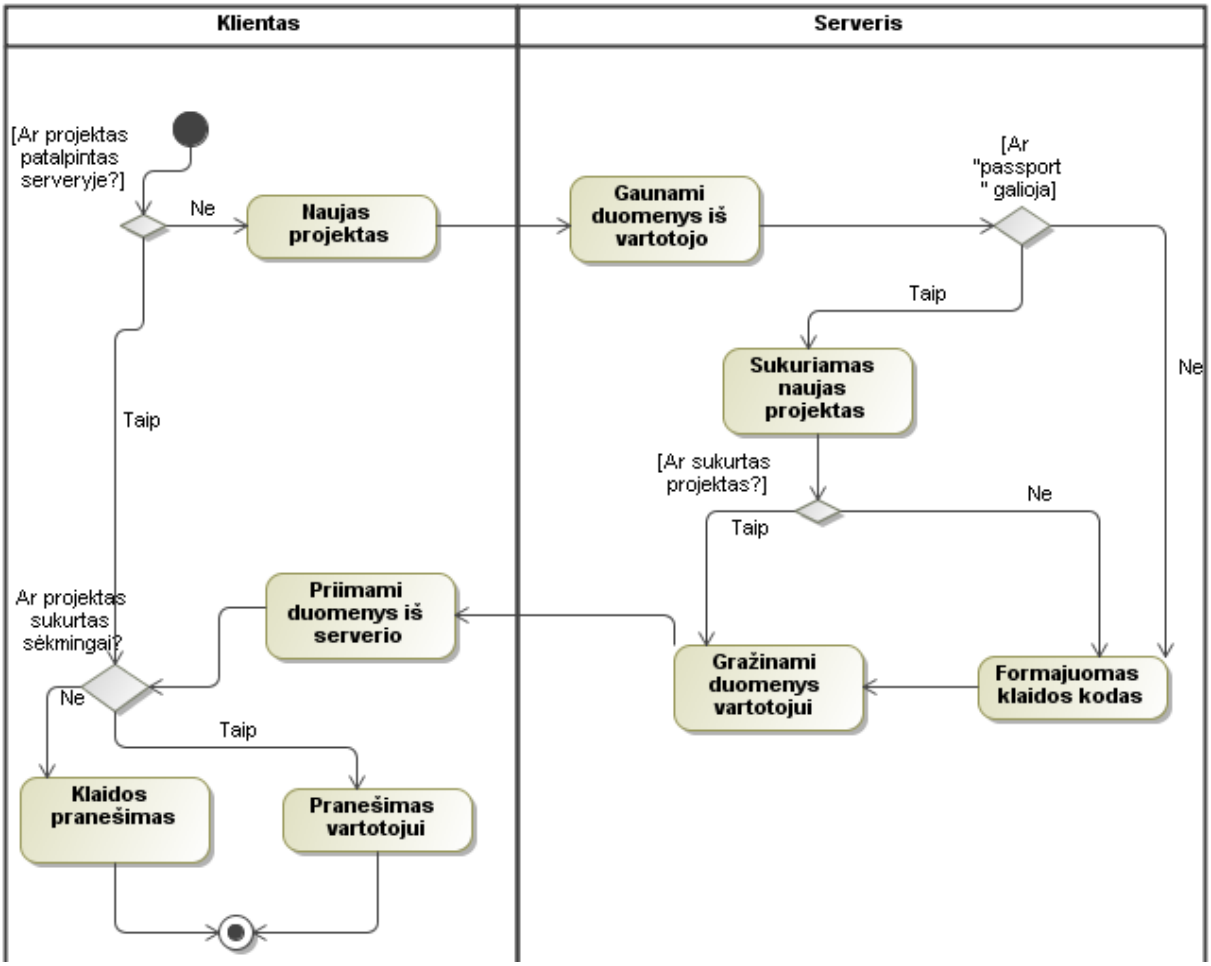
Veiklos diagrama pavaizduota paveikslėlyje (12 pav.) parodo vykdoma veiksmų seka, nuo to laiko kai vartotojas sužadina veiksmą „*atsisiųsti projektą iš serverio*“. Norint sužadinti šį veiksmą reikia turėti „*passport*“ arba pirmiausia atlikti veiksmą „*gauti passport*“.



Pav. 12. Veiklos diagrama panaudojimo atvejui „*Atsisiųsti projektą iš serverio*“

Sukurti projektą

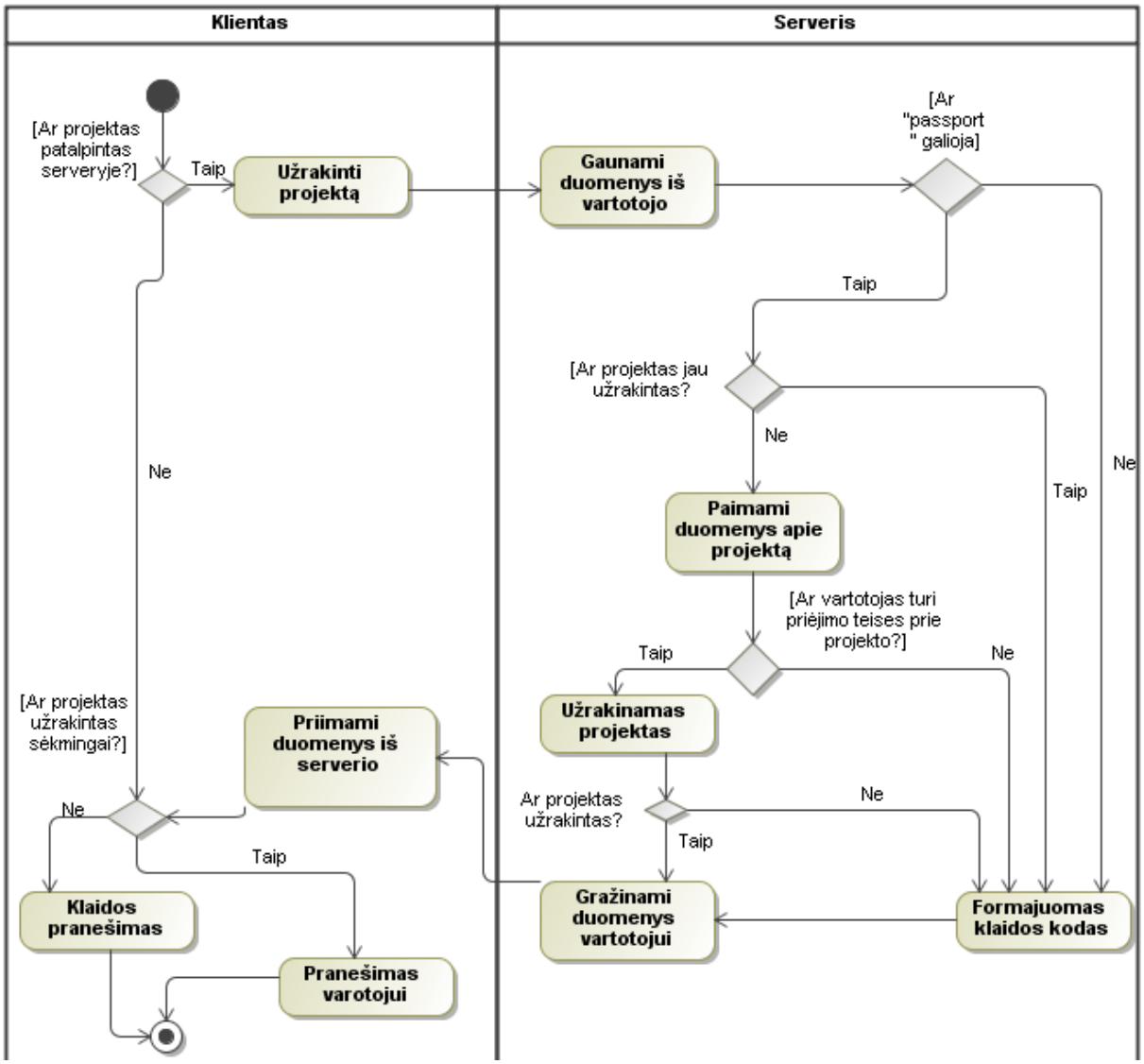
Veiklos diagrama pavaizduota paveikslėlyje (13 pav.) parodo vykdoma veiksmų seka, nuo to laiko kai vartotojas sužadina veiksmą „*sukurti projektą*“. Norint sužadinti šį veiksmą reikia turėti „*passport*“ arba pirmiausia atlikti veiksmą „*gauti passport*“.



Pav. 13. Veiklos diagrama panaudojimo atvejui „*Sukurti projektą*“

Užrakinti projektą

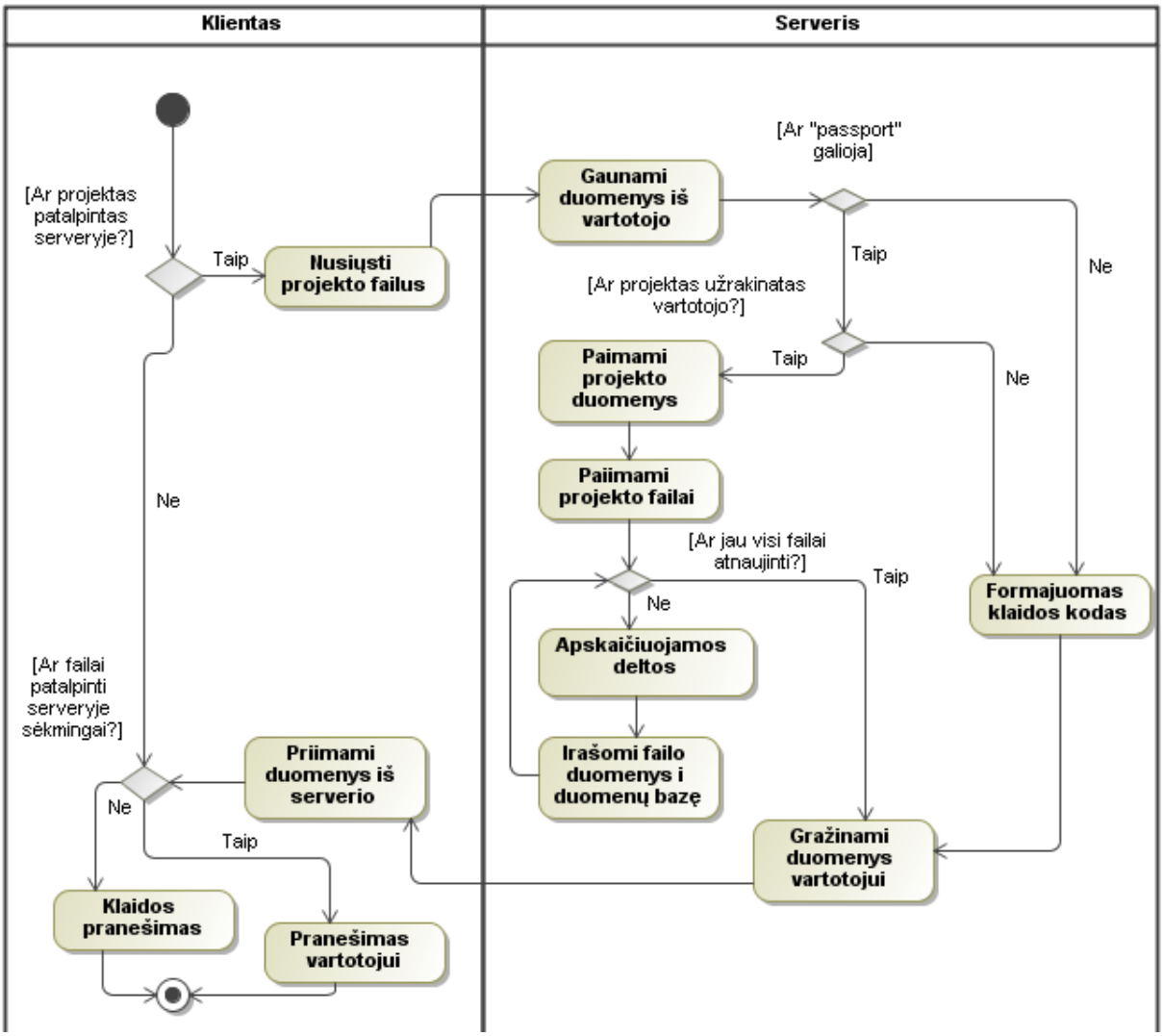
Veiklos diagrama pavaizduota paveikslėlyje (14 pav.) parodo vykdoma veiksmų seka, nuo to laiko kai vartotojas sužadina veiksmą „*užrakinti projektą*“. Norint sužadinti šį veiksmą reikia turėti „*passport*“ arba pirmiausia atlikti veiksmą „*gauti passport*“. Taip pat projektas turi būti darbo aplinkoje sukurtas veiksmu „*sukurti projektą*“ ar „*atsisiųsti projektą iš serverio*“.



Pav. 14. Veiklos diagrama panaudojimo atvejui „*Užrakinti projektą*“

Nusiųsti projektą į serverį

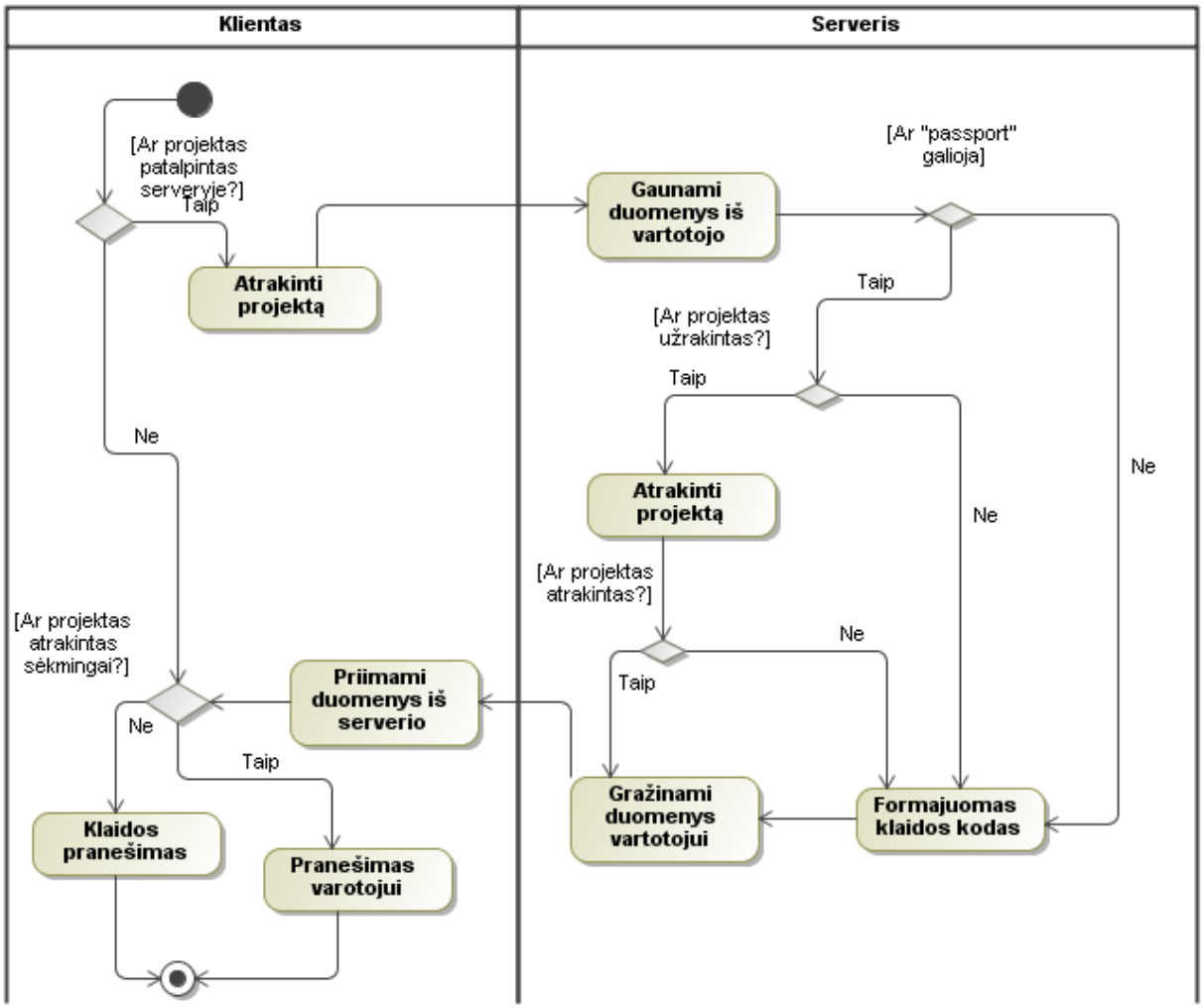
Veiklos diagrama pavaizduota paveikslėlyje (15 pav.) parodo vykdoma veiksmų seka, nuo to laiko kai vartotojas sužadina veiksmą „*nusiųsti projektą į serverį*“. Norint sužadinti šį veiksmą reikia turėti „*passport*“ arba pirmiausia atlikti veiksmą „*gauti passport*“. Taip pat projektas turi būti darbo aplinkoje sukurtas veiksmu „*sukurti projektą*“ arba „*atsisiųsti projektą iš serverio*“.



Pav. 15. Veiklos diagrama panaudojimo atvejui „*Nusiųsti projektą į serverį*“

Atrakinti projektą

Veiklos diagrama pavaizduota paveikslėlyje (16 pav.) parodo vykdoma veiksmų seka, nuo to laiko kai vartotojas sužadina veiksmą „*atrakinti projektą*“. Norint sužadinti šį veiksmą reikia turėti „*passport*“ arba pirmiausia atlikti veiksmą „*gauti passport*“. Taip pat projektas turi būti darbo aplinkoje sukurtas veiksmu „*sukurti projektą*“ arba „*atsisiųsti projektą iš serverio*“.



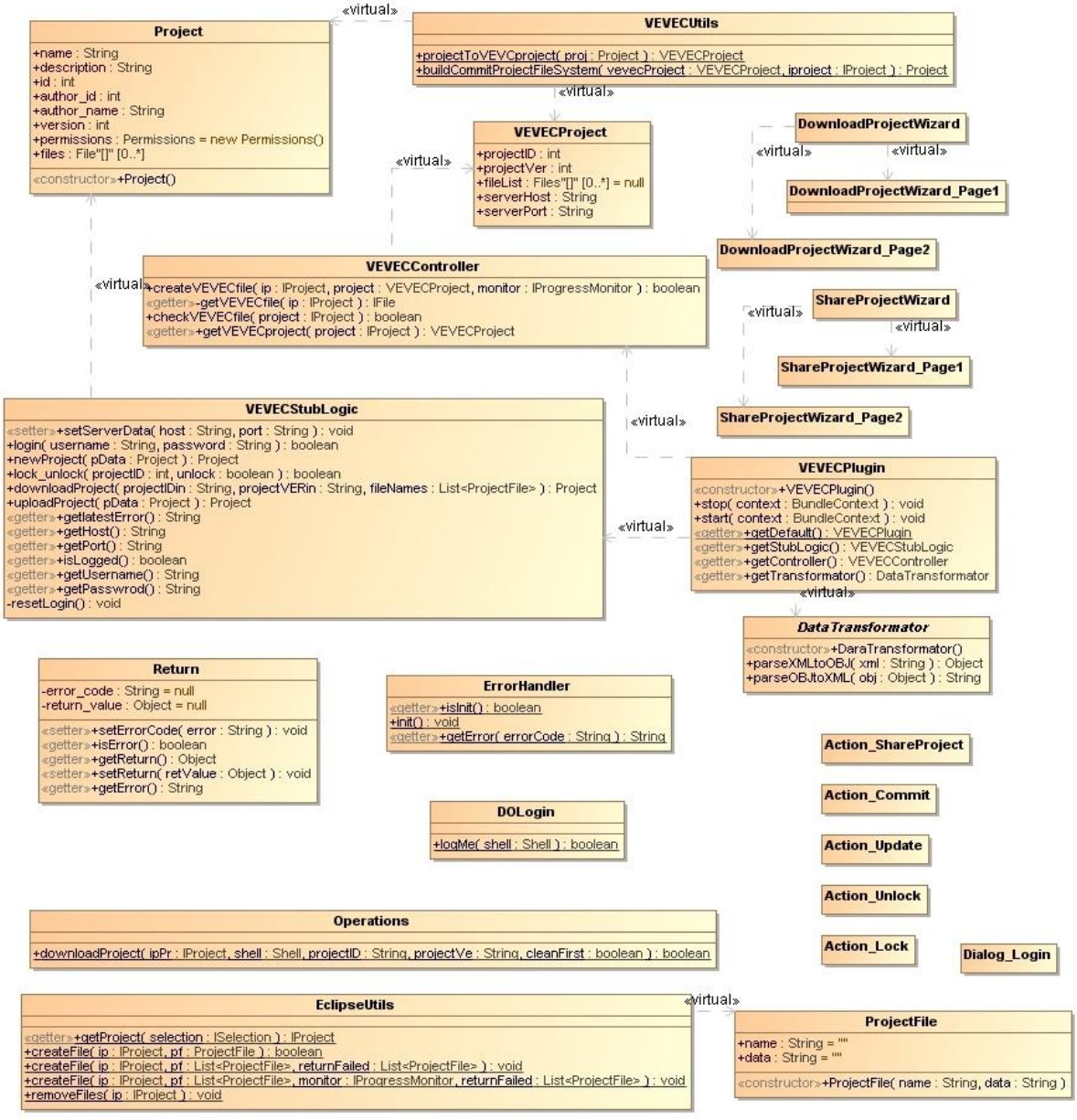
Pav. 16. Veiklos diagrama panaudojimo atvejui „*Atrakinti projektą*“

4.5.3. KLASIŲ DIAGRAMOS

Grafinės sąsajos klasių diagrama

Klasė „*VEVECPlugin*“ aktyvuoją sukurta plėtinį, kuris integruojasi į VeTIS grafinę sąsaja ir leidžia pasinaudoti nauju funkcionalumu.

Visa pagrindinė logika bendrauti su serverio modulio yra „*VEVECStubLogic*“ klasėje, šioje klasėje visi metodai skirti naudotis per WSDL sąsaja prieinamomis serverio paslaugomis. Iš serverio gautus „String“ duomenis su „*DataTransformator*“ klasės metodais sukuriamas „*Return*“ objektas toks koks buvo išsiųstas iš serverio.



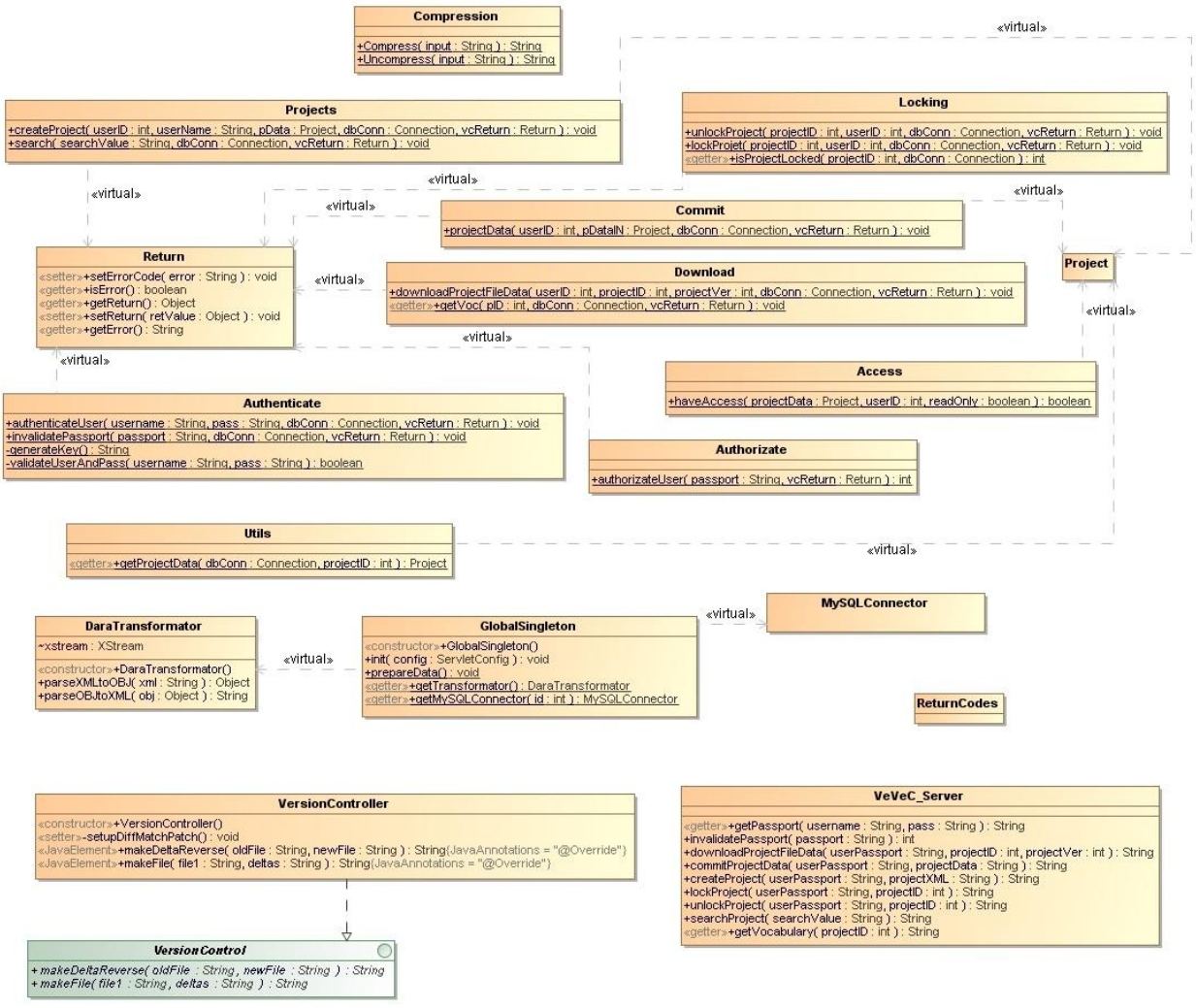
Pav. 17. Grafinės sąsajos klasių diagrama

Serverio modulio klasių diagrama

Klasė „GlobalSingleton“ iššaukiama startavus serverio moduliui, joje yra sukuriamas ir laikomas MySQL adapterio objektas „MySQL_Connector“ jungiantis į projekto duomenų bazę. Taip pat sukuriamas ir saugomas objektas „DataTransformator“ skirtas objektus transformuoti į XML ir atvirkščiai.

Klasės „VeVeC_Server“ metodai yra prienami per sugeneruota WSDL sąsaja. Iš šios klasės iššaukiami statiniai vykdymo metodai esantys klasėse: „Projects“, „Locking“, „Commit“, „Download“, „Authenticate“ klasėse.

Duomenys vartotojui gražinami konvertavus „Project“ objektą su saugomais kintamaisiais į „String“ duomenų formatą, kur objekto duomenys saugomi XML struktūroje.



Pav. 18. Serverio modulio klasių diagrama

5. SERVERIO MODULIO IR GRAFINĖS SĄSAJOS REALIZACIJA

5.1. SERVERIO MODULIO VEIKIMO APRAŠYMAS

Serverio dalies modulis yra patalpinamas į WEB serverį su Tomcat 7 moduliu. WEB servisas turi WSDL sąsaja su prieinamomis operacijomis:

- getPassport – identifikuojamas vartotojas ir suteikiamas unikalus raktas, leidžiantis naudotis serverio paslaugomis.
- unlockProject – atrakinti projektą.
- commitProjectData – nusiųsti projekto failus į serverį.
- createProject – sukurti projektą.
- downloadProjectFileData – parsisiųsti projekto failus.
- lockProject – užrakinti projektą.

Kad būtų galima naudotis serverio paslaugomis vartotojas turi pirma identifikuotis. Vartotojo identifikavimas atliekamas naudojantis (getPassport), jeigu pateikiamas teisingas varotojo vardas ir slaptažodis suteikiamas unikalus raktas identifikuojantis vartotoją sistemoje. Visos serverio funkcijos naudojamos su gautuoju raktu.

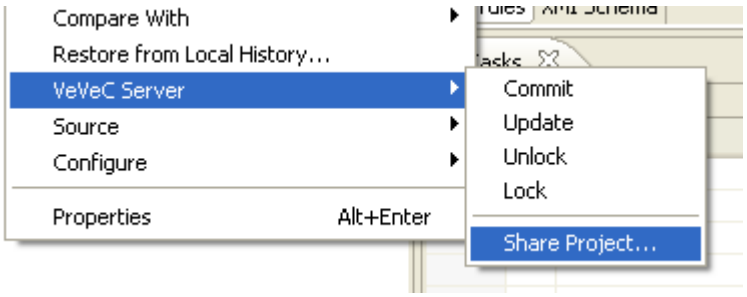
Kad galėtume išsaugoti naujo projekto duomenys serveryje pirmiausia sukuriame projektą (createProject), tada turime užrakinti projektą (lockProject), kad galėtumėme siųsti projekto failus į serverį (commitProjectData). Baigus darbą su projektu turime jį atrakinti (unlockProject), kad prie jo galėtų prieiti kiti vartotojai.

Į serverį nusiųstas projektas yra išsaugomas duomenų bazėje, jeigu prieš tai buvo kita versija yra apskaičiuojamos deltos, kas leidžia esant poreikiui iš naujos versijos grįžti į ankstesnę. Norint gauti pirmąją projekto versiją, o saugomas projektas yra penktos versijos, tokiu atveju serverio modulis uždės deltas atvirkštine tvarka ant naujausios versijos. Pavyzdžiui, kad gautume 4 versiją ant 5 dokumento versijos bus uždedamos deltos, kurios buvo apskaičiuotos tarp 4 ir 5 versijų. Sekantis žingsnis, kad gauti 3 versija ant 4 versijos bus uždėtos deltos gautos skaičiuojant skirtumas tarp 3 ir 4 versijos ir t.t. kol bus sudaryta 1 projekto versija.

5.2. PRAPLĖSTOS VETIS GRAFINĖS SĄSAJOS APŽVALGA

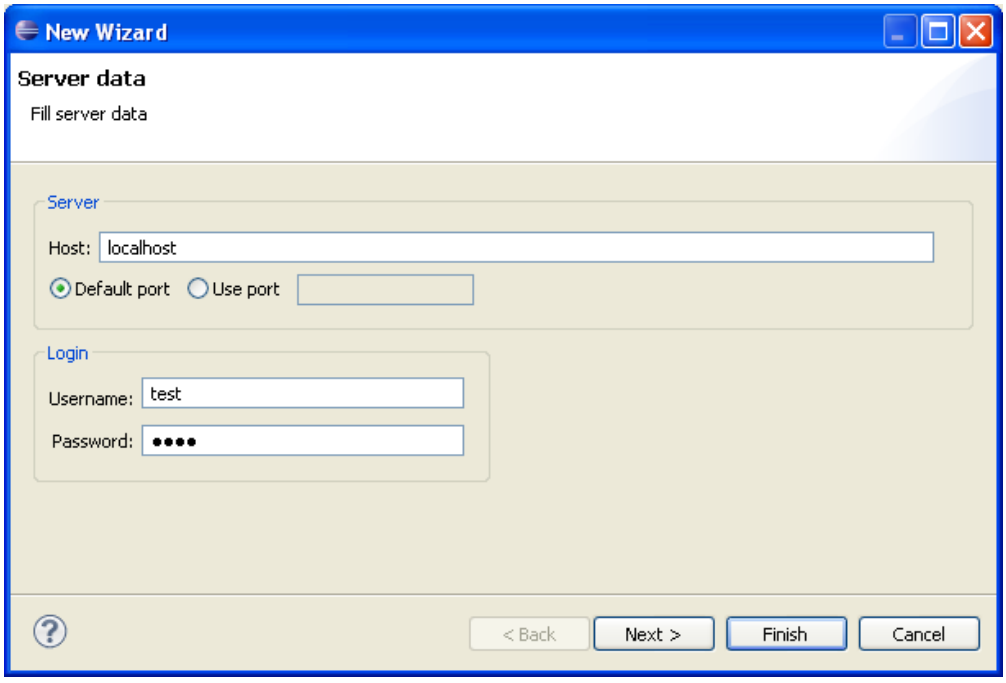
5.2.1. NAUJO PROJEKTO SUKŪRIMAS SERVERYJE

Vartotojas norėdamas patalpinti savo projektą serveryje, spaudžia ant projekto dešinį pelės mygtuką ir iš „*VeVeC Server*“ srities turi pasirinkti „*Share Project...*“, kaip parodyta paveikslėlyje (19 pav.).



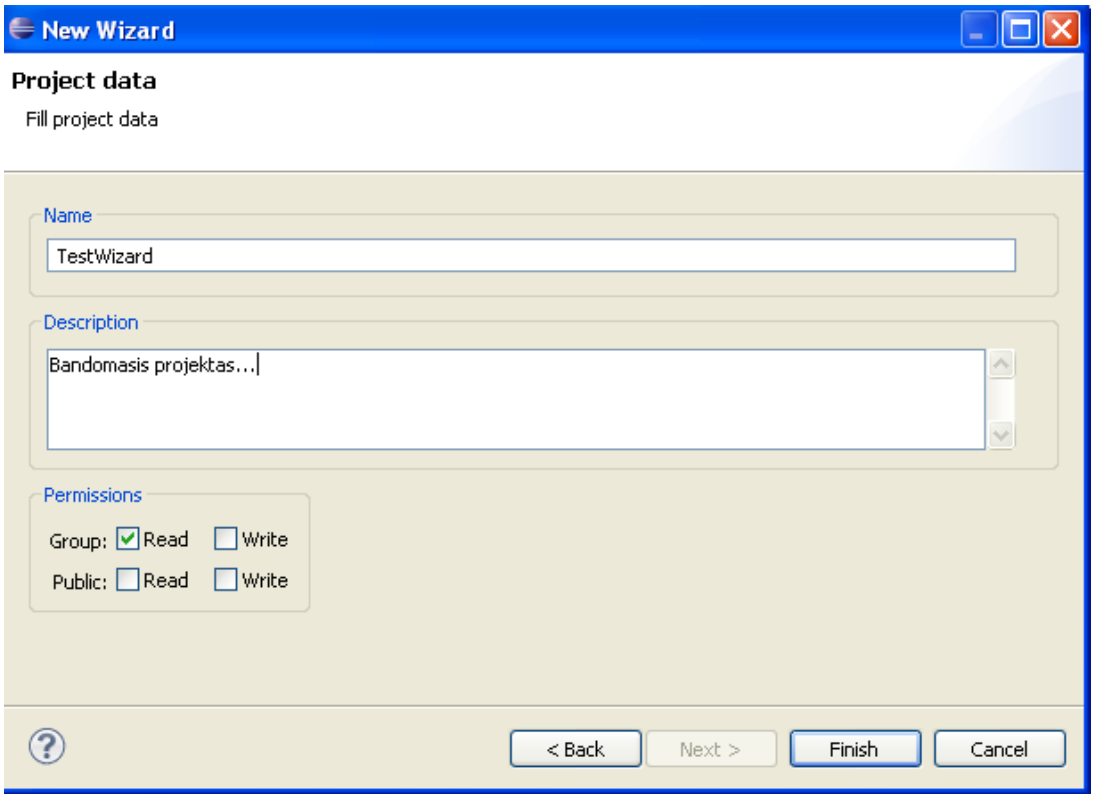
Pav 19. „*Share Project...*“ meniu pasirinkimas

Atsidaro vedlio pirmasis langas, kurį galima matyti paveikslėlyje (20 pav.). Šiame lange reikia nurodyti serverio adresą ir prievadą (jeigu serveris naudoja kita, t.y. ne 8080 prievadą). Taip pat turi būti užpildyti prisijungimo duomenys (projektas susiejamas su vartotoju). Atlikus visus šiuos veiksmus spaudžiama „*Next*>“.



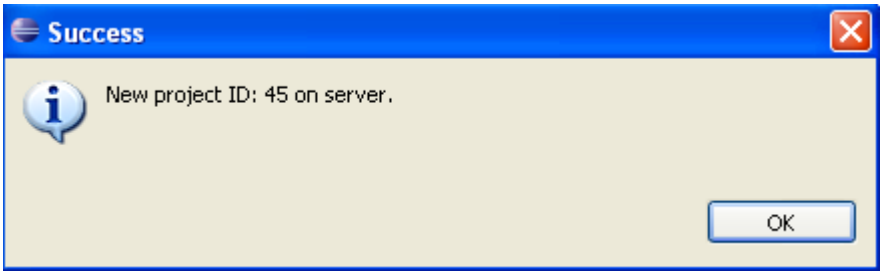
Pav. 20. Vedlio langas su serverio nustatymais

Atsidaro antrasis vedlio langas, kuris pateiktas paveikslėlyje (21 pav.). Šiame lange užpildomi duomenys apie projektą (pavadinimas, aprašymas, leidimai).



Pav. 21. Vedlio langas su projekto nustatymais

Paspaudus „*Finish*“ vartotojas gaus pranešimą ar pavyko sukurti projektą ar ne, jeigu nepavyko bus pranešta dėl ko nepavyko. Jeigu duomenis serveryje sėkmingai užregistruoti bus išvestas pranešimas kokiu ID projektas išsaugotas (22 pav.).



Pav. 22. Pranešimas apie sėkminga projekto išsaugojimą

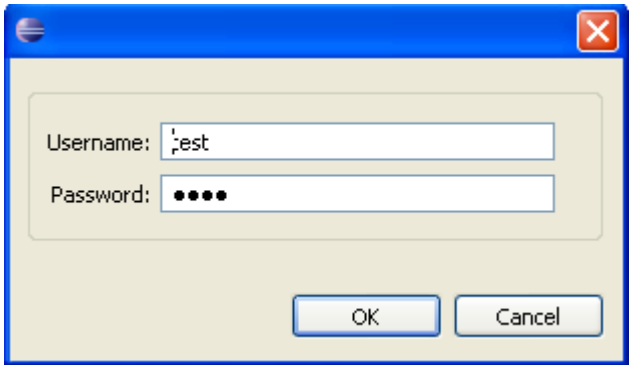
Vartotojas su gautu projekto ID numeriu gali parsisiųsti savo projektą iš serverio į kitą kompiuterį arba į tą patį tik kitu vardu (darbinėje aplinkoje).

5.2.2. PROJEKTO UŽRAKINIMAS IR ATRAKINIMAS

Norint išsaugoti projekto failus serveryje projektas visų pirma turi būti užrakintas. Projekto užrakinimas atliekamas paspaudus dešinį pelės mygtuką ant projekto ir pasirinkus „*VeVeC Server*“ srities parinktį „*Lock*“ užrakinti, kaip aprodyta paveikslėlyje (23 pav.). Norint atrakinti projektą (atrakinti projektą gali tik tas pats vartotojas kuris jį užrakino) spaudžiama „*Unlock*“.

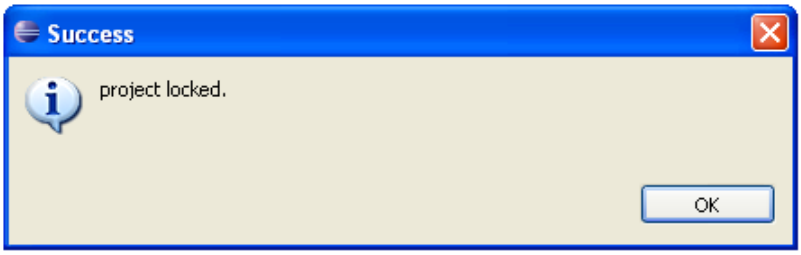
Pav. 23. „Lock“ meniu

Paspaudus „*Lock*“ arba „*Unlock*“ vartotojas gaus pranešimą ar operacija buvo sėkminga ar ne. Jeigu vartotojas nėra prisijungęs prie to serverio su kuriuo yra susietas projektas, tokiu atveju iššoks prisijungimo langas, kaip pavaizduota paveikslėlyje (24 pav.).



Pav. 24. Prisijungimo langas

Jeigu buvo panaudoti teisingi prisijungimo duomenys, tai bus bandoma užrakinti/atrakinti projektą, o vartotojas informuotas ar operacija atlikta sėkminga ar ne. Sėkmės atveju bus matomas langas, kaip paveikslėlyje (25 pav.).



Pav. 25. Sėkmingo užrakinimo pranešimas

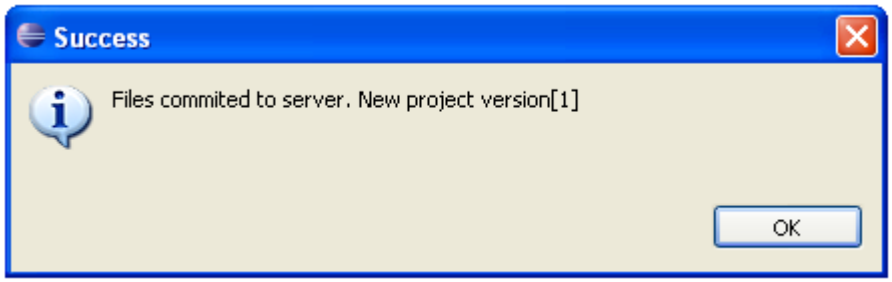
5.2.3. DUOMENŲ TALPINIMAS SERVERYJE

Norėdami nusiųsti projekto failus į serverį iš „*VeVeC Server*“ meniu pasirenkame „*Commit*“ matoma paveikslėlyje (26 pav.).

Pav. 26. „Commit“ meniu pasirinkimas

Jeigu vartotojas yra neprisijungęs išvedamas prisijungimo langas, kaip buvo aprašyta ankstesniame skyrelyje ir pavaizduota paveikslėlyje (24 pav.).

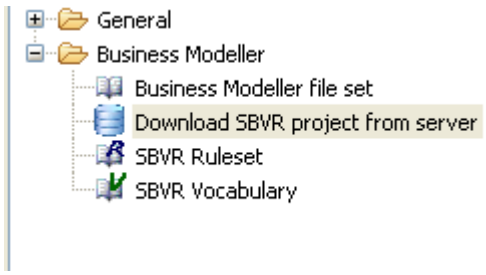
Sėkmingai nusiuntus failus vartotojui parodomas pranešimas apie sėkminga failų išsaugojimą ir kokia versija jie buvo išsaugoti (27 pav.). Nesėkmės atveju parodomas pranešimas kodėl nepavyko atlikti operacijos.



Pav. 27. Pranešimas apie sėkmingą failų išsaugojimą serveryje

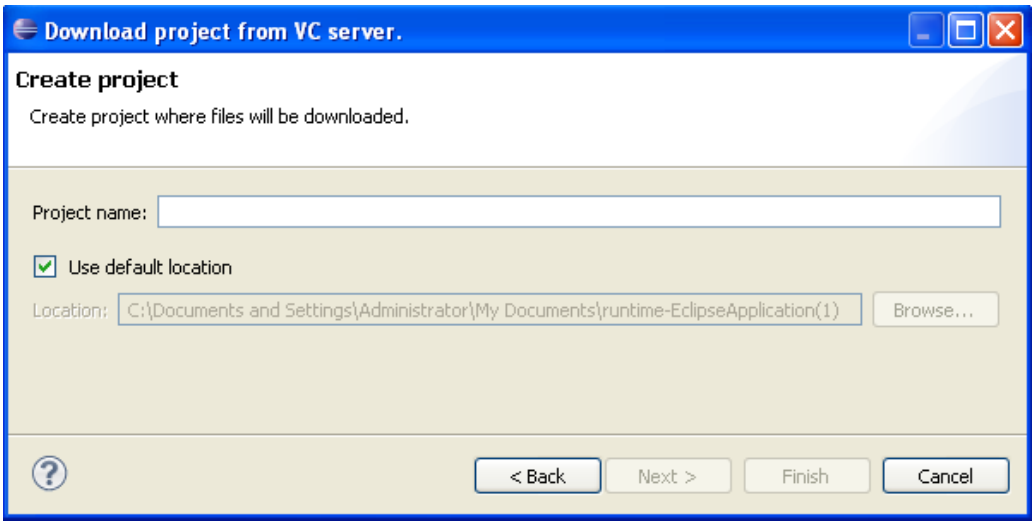
5.2.4. PROJEKTO PARSISIUNTIMAS IŠ SERVERIO

Norint parsisiųsti naują projektą iš serverio reikia atlikti šios veiksmu., Iš naujų projektų sąrašo pasirenkama „*Donwload SBVR Project from server*“, kuris yra „*Business Modeller*“ skiltyje, kaip matoma paveikslėlyje (28 pav.).



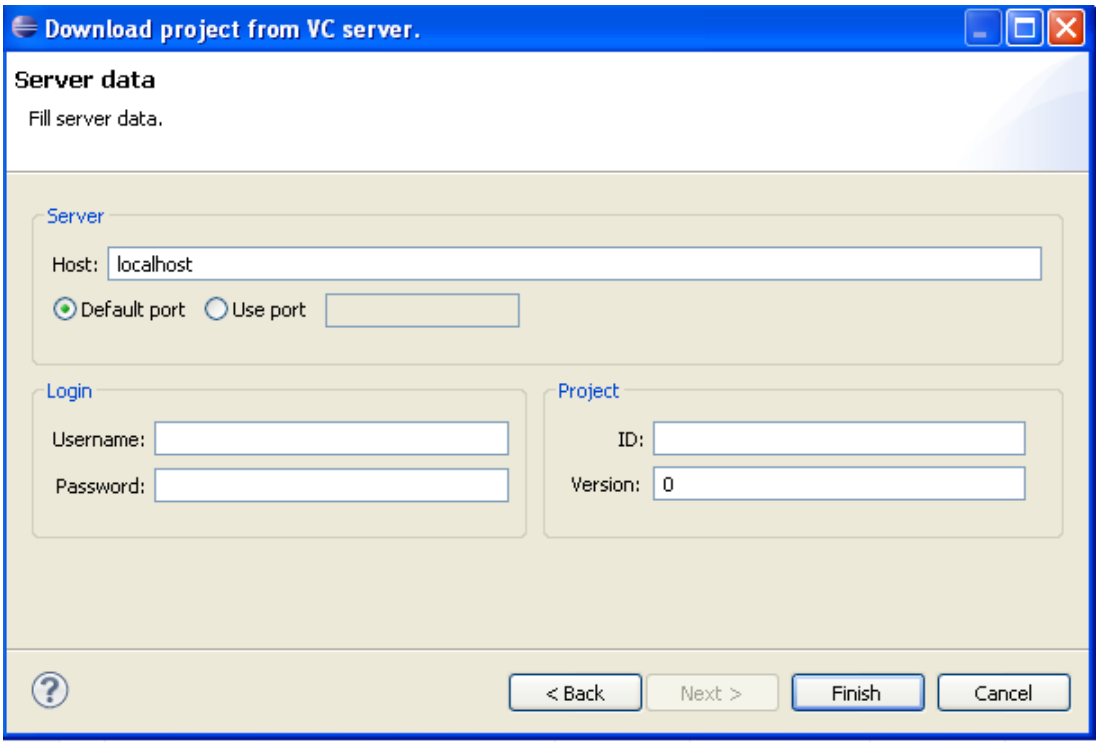
Pav. 28. Projekto parsisiuntimo iš serverio meniu

Pasirinkus šį menių iškviečiamas vedlys. Pirmasis vedlio langas yra skirtas projekto pavadinimo pasirinkimui, bei nurodymui kur jis bus išsaugotas (29 pav.).



Pav. 29. Projekto pavadinimas darbo aplinkoje

Antrasis vedlio langas (30 pav.) yra skirtas serverio nustatymams. Čia nurodomi prisijungimo duomenys prie serverio, bei projekto ID ir versija kurią norima parsisiųsti. „*Version*“ skiltį palikus „0“ bus parsiųsta naujausia projekto versija.

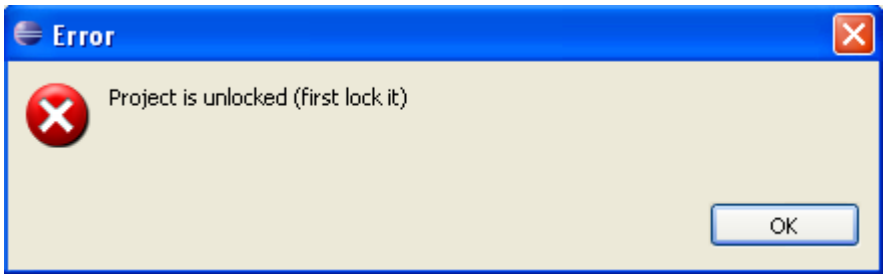


Pav. 30. Prisijungimas prie serverio, serverio adreso nustatymai ir projekto pasirinkimas

Paspaudus „*Finish*“ projektas yra parsiunčiamas ir įrašomas į darbo aplinką. Jeigu nepavyksta parsiųsti parodomas pranešimas kodėl nepavyko.

5.2.5. PRANEŠAMŲ KLAIDŲ SĄRAŠAS

Klaidos atveju yra matomas langas su „*Error*“ antrašte (31 pav.).



Pav. 31. Klaidos pranešimo langas

Naudojantis „*VeVeC*“ serverio paslaugomis galime sutikti viena iš šių pranešimų pateiktų lentelėje (2 lentelė).

Lentelė 2. Klaidų kodai su pranešimais ir aprašymais.

Grafinės sąsajos klaidos		
Kodas	**Pranešimas**	**Priežastis**
LERR_001	Unable connect to server	Negali susijungti su serveriu nurodytu adresu ar prievadu.
LERR_002	Server responce error.	Iš serverio gautas atsakas yra nesuprantamas ar klaidingas.
LERR_003	Specified bad login data.	Nurodyti klaidingi prisijungimo duomenys (palikti tušti prisijungimo laukai).
LERR_004	Need to be logged in first.	Norint naudotis pasirinkta funkcija pirmiausiai reikia prisijungti.
LERR_005	Bad input data.	Įvesti duomenys yra netinkami (ar tušti) nes neužpildyti (ar klaidingai užpildyti) privalomi laukai.
Iš serverio gaunami klaidų pranešimai		
Kodas	**Pranešimas**	**Priežastis**
ERR001	Server database error.	Serveris turi problemų su duomenų baze.
ERR002	Specified bad parameters in function	Serveriui nusiųsti funkcijos parametrai yra netinkami.
ERR101	Bad login data.	Neatpažintas nurodytas vartotojas.
ERR102	Invalid passport	Naudojamas negaliojantis „passport“

ERR201	No access to project	Neturima leidimo prieiti prie nurodyto projekto.
ERR202	Project already locked.	Projektas jau yra užrakintas.
ERR203	Project version requested not exist.	Prašoma projekto versija neegzistuoja.
ERR204	Project not exists with this ID	Projektas su nurodytu ID neegzistuoja.
ERR206	Project is unlocked (first lock it)	Projektas visų pirma turi būti užrakintas.
ERR207	Project already locked by you.	Projektas jau yra užrakintas. Pranešama kai tas pats vartotojas bando vėl užrakinti užrakinta ta patį projektą.
ERR300	Unknown lock	Tikrinant projekto užrakinimą iškilo klaida.
ERR301	Unknown server error	Serveris dirba klaidingai

6. EKSPERIMENTINIS SERVERIO MODULIO SPARTOS TYRIMAS

6.1. EKSPERIMENTO TIKSLAS

Eksperimento tikslas nustatyti WEB modulyje naudojamu metodu vykdymo laiką ir kaip jis kinta, esant įvairiam apkrovimui, t.y. kai WEB modulis aptarnauja daugiau nei vieną klientą tuo pačiu metu.

6.2. EKSPERIMENTO METODIKA

Serverio modulis paleistas Eclipse SDK aplinkos vidiniame Tomcat v7.0.12 serveryje su Axis2 v1.6.1 palaikymu. Naudojama duomenų bazė MySQL v5.5.20.

Serverio operacinė sistema Windows XP SP3 paleista WMware virtualioje mašinoje su išskirta 1.3 GB operatyviąja atmintimi, centrinis procesorius Intel i3-370M.

Kiekvienoje WEB serverio modulio tiriamų metodų pradžiose ir pabaigose įterptas papildomas laiko matavimo kodas, kuris apskaičiuoja kiek laiko užtruko metodo vykdymas.

Kontroliuoti kaip klientai kreipiasi į serverį buvo parašyta testavimo programa, kuri atlieka iš anksto nustatyta veiksmų seką:

- prisijungia prie serverio;
- sukuria naują projektą;
- užrakina projektą;
- sugeneravus dokumentą iš atsitiktinių simbolių (8 KB didumo) įkelia į serverį kaip pirmąja versiją;
- atsisiunčia pirmąją projekto versiją ir atlikus atsitiktinius pakeitimus duomenyse nusiunčia atgal į serverį kaip naują versija (šis veiksmas kartojamas 50 kartų);
- atrakina projektą.

Eksperimentas atliekamas su testavimo programa, kuri kreipiasi į WEB moduli imituodama klientą, eksperimentas kartojamas penkis kartus pradedant vieno vartotojo

imitavimu ir vis pridedant viena papildomi vartotoją, kol pasiekiama kad penki vartotojai vienu metu vykdo aprašyta veiksmų seką.

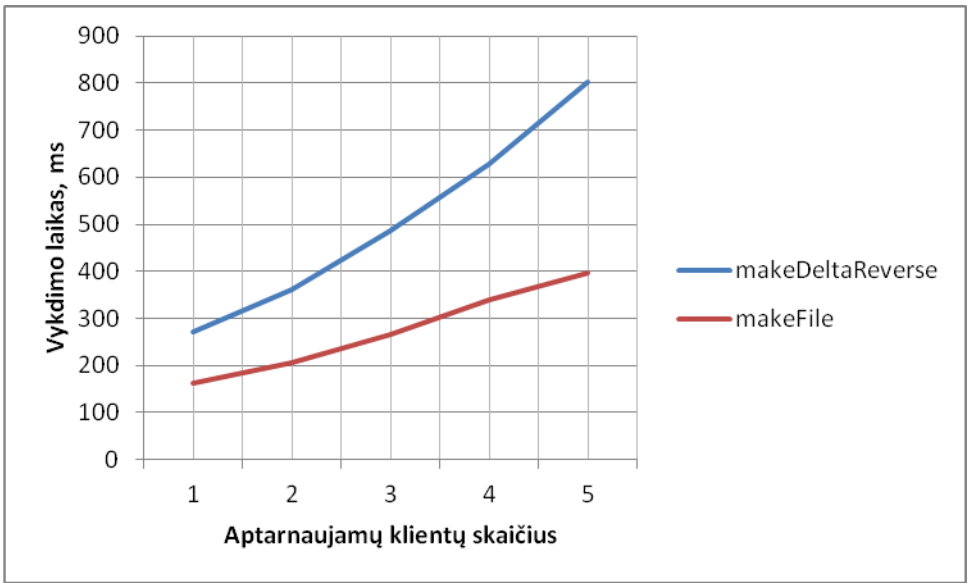
6.3. EKSPERIMENTO REZULTATAI

Eksperimente analizuoti metodai pateikti lentelėje (3 lentelė). Čia pateikti trumpi jų aprašymai nurodant kokia to metodo paskirtis.

Lentelė 3. Analizuojami metodai

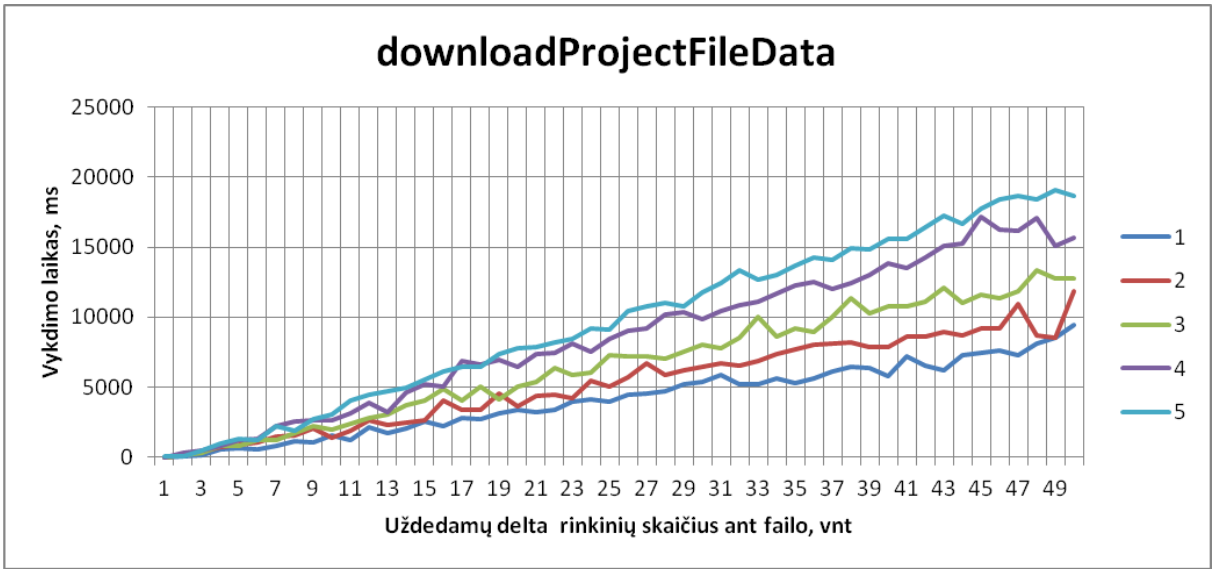
Metodo pavadinimas	**Paskirtis**
projectData	Nauji duomenis įrašomi į duomenų baze, bei paskaičiuojamos deltos tarp naujos ir jau esamos duomenų versijos.
getProjectData	Paimami duomenys iš DB lentelės „project“ su nurodytu projekto ID numeriu.
makeDeltaReverse	Apskaičiuojamos deltos tarp dviejų failų naudojantis *google-diff-match-patch* biblioteka.
makeFile	Iš deltų ir failo, sudaroma senesnė dokumento versija naudojantis *google-diff-match-patch* biblioteka.
downloadProjectFileData	Iš duomenų bazės paimami visi projekto failų duomenys ir jeigu vartotojas prašo sudaroma norima versija (senesnė).

Metodų *makeDeltaReverse* ir *makeFile* vykdymo laiko priklausomybė nuo prisijungusių klientų skaičiaus pateikta paveikslėlyje (32 pav.). Iš diagramos matoma, kad sudaryti deltas užtrunka kur kas ilgiau, nei iš deltų sudaryti failą. Pagal grafiko duomenis matosi, kad aptarnaujamų klientų skaičius turi kur kas stipresnę įtaka ilgėjančiam vykdymo laikui deltų sudaryme, nei failo sudaryme.



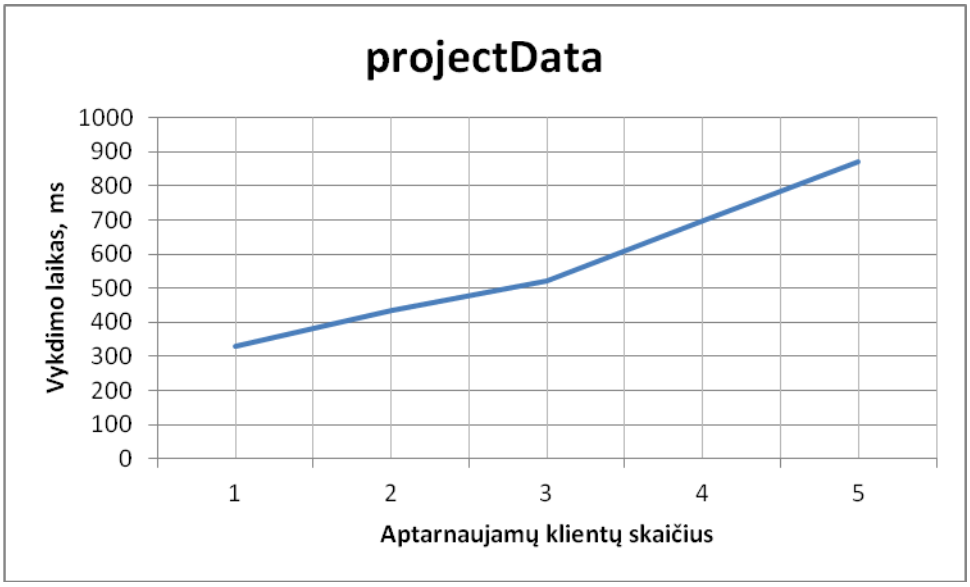
Pav. 32. *makeDeltaReverse* ir *makeFile* vykdymo laiko priklausomybė nuo prisijungusių klientų skaičiaus

Metodo *downloadProjectFileData* analizė pateikta paveikslėlyje (33 pav.). Grafike pateikiama kokia įtaka turima metodo vykdymo laikui, kai vienu metu aptarnaujama nuo 1 iki 5 klientų. Iš grafo matosi jeigu klientai siunčiasi naujausią versiją (uždedamų delta skaičius yra 0), aptarnaujamų klientų skaičiaus įtakos praktiškai nesimato. Tačiau jeigu klientai siunčiasi senesnes failo versijas (grįžti 50 failo versijų atgal), vieno vartotojo aptarnavimas užtrunka apie 9 sekundes, tačiau šis laikas stipriai išauga, kai aptarnaujama 5 klientus vienu metu net iki 19 sekundžių.



Pav. 33. *downloadProjectFileData* vykdymo laiko kitimas

Metodo *projectData* vykdymo laiko augimas yra tolydus. Kai aptarnaujamas vienas klientas metodo vykdymas užtrunka apie 0,3 sekundės ir tolydžiai didėja iki 0,89 sekundės, kai aptarnaujami 5 klientai vienu metu.



Pav. 34. *projectData* metodo vykdymo laiko kitimas

6.4. EKSPERIMENTO IŠVADOS

Atlikus eksperimentą pastebėta, kad metodų kurie iššaukia DB užklausą ir pateikia duomenis, bet neatlieka jokių papildomų skaičiavimų, vykdymo laikas yra trumpas 0,01 sekundės. Jeigu tokie metodai įšaukiami eilę kartų, vykdymo laiko neišeina išmatuoti, nes matuojant milisekundėmis vykdymo laikas gaunamas 0 milisekundė, dėl duomenų laikymo atmintyje.

Metoduose kur reikalinga atlikti daug skaičiavimų, tokiame kaip *downloadProjectFileData*, vykdymo laikas smarkiai išauga, net iki 20 sekundžių, kai reikia sudaryti senesnę projekto versiją ir aptarnauti 5 klientus vienu metu, kurie irgi prašo iš serverio senesnės projekto versijos.

7. IŠVADOS

1. Magistriniame darbe analizuotose versijų kontrolė sistemos, pastebėta kad praktiškai visos jų naudoja tas pačias metodikas duomenų konkurenciniam modifikavimui ir duomenų suliejimo problemoms spręsti.

2. Nerasta nei vienos versijų kontrolės sistemos, kuri veiktų kaip WEB servisas ir duomenis saugotų *MySQL* duomenų bazėje, bei duomenų skirtumus apskaičiuotų serveryje, o ne kliente.

3. Analizės metu nustatyta, kad visų šiuolaikinių duomenų suliejimo algoritmų pagrindas yra Myers 1986 m. „Algorithmica“ žurnalo straipsnyje aprašytas algoritmas.

4. Sukurta programinė versijų kontrolės sistema, kuri veikia WEB serviso pagrindu parodė, kad tokia sistema funkcionuoja, tačiau reikalauja daugiau resursų aptarnaujant klientus. Išaugusios resursų sąnaudos dėl to, kad visi pagrindiniai (skirtumų radimas tarp failų ir failo sudarymas iš deltų) skaičiavimai realizuoti serveryje.

5. Pastebėtas sistemos privalumas kur visi skaičiavimo metodai realizuoti serveryje, toks kad keičiant naudojamas technologijas duomenims sulieti ar saugoti, nereikia atnaujinti klientinės programinės įrangos.

6. Sukurta versijos kontrolės sistemą galima naudoti ne vien tik VeTIS projektams, bet visiem projektams kurie sudaryti iš tekstinių duomenų.

LITERATŪRA

[1] Vesperman J. Essential CVS. JAV: O'Reilly Media, 2003. 336 p.

[2] Collins-Sussman B, Fitzpatrick B. W, Pilato C. M Version Control with Subversion. JAV: O'Reilly Media, 2008. 320 p.

[3] Vassia Atanassova, Net Models of Conflict Resolution Approaches in Version Control Systems. Part 1: Lock-Modify-Unlock, p. 11

[4] Comparison of revision control software. [žiūrėta 2011-08-04]. Prieiga per internetą: http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

[5] Khanna S., Kunal K., Pierce B. C. A Formal Investigation of Diff3 – 27: Foundations of Software Technology and Theoretical Computer Science (FSTTCS) pranešimų medžiaga [New Delhi, India, Gruodžio 12-14 d.]. New Delhi, 2007, p. 485-496

[6] Frequently Asked Questions//Mercury. [žiūrėta 2011-08-04]. Prieiga per internetą: http://mercurial.selenic.com/wiki/FAQ

[7] git-merge(1)//Git. [žiūrėta 2011-08-05]. Prieiga per internetą: http://www.kernel.org/pub/software/scm/git/docs/git-merge.html

[8] Libresource. [žiūrėta 2011-08-10]. Prieiga per internetą: http://dev.libresource.org/

[9] Diffuse. [žiūrėta 2011-08-11]. Prieiga per internetą: http://diffuse.sourceforge.net

[10] Altmanninger K., Seidl M., Wimmer M., A survey on model versioning approaches//International Journal of Web Information Systems. ISSN: 1744-0084, Nr. 5, p.271 – 304

[11] Monotone documentation//Monotone [žiūrėta 2011-09-07]. Prieiga per internetą: http://wiki.monotone.ca/

[12] Vassia Atanassova, Peter Georgiev. Generalized net models of conflict resolution approaches in version control systems. Part 2: Copy-modify-merge – 10: International Workshop on Generalized Nets pranešimų medžiaga [Sofia, Bulgaria, 2009 m. Gruodžio 5 d.]. Sofia, 2009, p. 14-21.

[13] Walter F. RCS—a system for version control// Software—Practice & Experience. Nr 15. p.637-654

[14] Greg Hudson. Notes on keeping version histories of files [žiūrėta 2011-09-07]. Prieiga per internetą: http://web.mit.edu/ghudson/thoughts/file-versioning

[15] VeTIS User Guide [žiūrėta 2012-03-15]. Prieiga per internetą: http://www.magicdraw.com/files/manuals/VeTISUserGuide.pdf

[16]Nicholas Butler. Investigating Myers' diff algorithm: Part 1 of 2 [žiūrėta 2012-03-25]. Prieiga per internetą: http://www.codeproject.com/Articles/42279/Investigating-Myers-diff-algorithm-Part-1-of-2

[17] Eugene W. Myers. An O(ND) Difference Algorithm and Its Variations//Algorithmica (1986) Nr. 1 p.251-266

[18] GNU diffutils//Gnu. [žiūrėta 2011-08-05]. Prieiga per internetą: http://www.gnu.org/s/diffutils/manual/

[19] google-diff-match-patch. [žiūrėta 2011-10-12]. Prieiga per internetą: http://code.google.com/p/google-diff-match-patch/

[20] XStream. [žiūrėta 2012-02-18]. Prieiga per internetą: http://xstream.codehaus.org/

PRIEDAI

1 Priedas. SERVERYJE TALPINAMO PROJEKTO DUOMENŲ VERSIJAVIMO APŽVALGA

Serveryje buvo sukurtas naujas projektas vardu „TestSBVR“ ir užrakintas vartotojo, kurio ID 5. Toliau buvo paimtas realus VeTIS projektas (vienas projektas susideda iš trijų failų: taisyklių, žodyno ir žodyno antraštės). Apžvalgoje bus pateiktas tik vienas projekto failas, kuris aprašo verslo taisykles „.rules“, nes deltų skaičiavimo operacijos atliekamos atskirai kiekvienam failui, o failai saugomi duomenų bazėje, tad nėra priklausomi vienas nuo kito.

Pradinė projekto *„.rules“* duomenų failo versiją, kuri buvo įrašyta į serverį kaip pirmoji versija, pavaizduota 1 lentelėje. Čia sunumeruotos failo eilutės. Šie eilučių numeriai serveryje nesaugomi, jie pateikiama tik kad vėliau būtų galima lengviau matyti kur ir kas buvo keičiama).

Lentelė 1. .rules duomenų failas

Eil.	*TestSBVR.rules*
1	It is necessary that a loan owns exactly one bail.
2	
3	It is possible that a bail is_owned_by at most one loan.
4	
5	It is possible that a debtor gets at most 3 loan.
6	
7	It is necessary that the loan is_got_by exactly one debtor.
8	
9	
10	It is possible that a branch is_included_in at most one parent_bank.
11	
12	It is necessary that the parent_bank includes at least 4 and at most 10 branch.
13	
14	
15	It is necessary that an account is_included_in exactly one bank.
16	
17	
18	It is necessary that a person owns at least 1 account.
19	
20	It is necessary that a person owns at most 5 account.
21	
22	It is necessary that the account is_owned_by exactly one person.
23	
24	
25	It is necessary that the loan is_given_by exactly one bank.
26	
27	
27	It is necessary that the owner owns at least 2 real_estate.
29	
30	It is necessary that the real_estate is_owned_by exactly one owner.
31	
32	
33	It is permitted that a debtor requests a loan.
34	
35	It is obligatory that a bank gives a loan.
36	
37	It is obligatory that a bank checks_validity_of a loan.
38	
39	It is necessary that an amount of_the loan is_not_greater_than 10000.
40	

41 42	It is necessary that a loan is_a reliable_loan if each loan is_got_by debtor is_returned loan.

Šiam failui atliksime serija įvairių pakeitimų, tokių kaip: įrašų modifikavimas, eilučių pašalinimas, naujų eilučių įrašymas ir eilučių bei žodžių perstūmimas.

7.1.1. ĮRAŠŲ MODIFIKAVIMAS

Bus atliktas duomenų failo įrašų modifikavimas t.y. esamo įrašo pakeitimas kitu. Duomenyse bus pakeistas penktos eilutės įrašas, bei trisdešimt devintos eilutės įrašas, kaip pavaizduota 2 lentelėje.

Lentelė 2. Atliktos įrašų modifikacijos

5 eilutė	
Prieš	It is possible that a debtor gets at most 3 loan.
Po	It is possible that a debtor gets at most 5 loan.
39 eilutė	
Prieš	It is necessary that an amount of_the loan is_not_greater_than 10000.
Po	It is necessary that an amount of_the loan is_not_greater_than 20000.

Po pakeitimo dokumentas yra išsaugotas kaip antroji (2) versija, kad būtų galima grįžti į pirmąją (1) versiją serveryje saugomos deltos:

=156 -1 +3 =906 -1 +1 =105

7.1.2. EILUČIŲ ŠALINIMAS

Bus atliekamas devynioliktos ir dvidešimtos eilučių šalinimas. 3 lentelėje pateikti įrašai kurie buvo pašalinti.

Lentelė 3. Pašalinti įrašai

Eil.	Šalinamas įrašas
19 20	It is necessary that a person owns at most 5 account.

Po pakeitimo dokumentas yra išsaugomas kaip trečioji (3) jo versija. Kad būtų galima grįžti į antrąją (2) versiją serveryje saugomos deltos:

=538 +a person owns at most 5 account.%0D%0A%0D%0AIt is necessary that
=574

7.1.3. EILUČIU PERSTUMIMAS

Bus atliekamas eilučių nuo 31-39 perkėlimas už 42 įrašo. Rezultate 40-42 eilutės taps 31-33, o 31-39 taps 34-42 eilutėmis, tai pavaizduota 4 lentelėje.

Lentelė 4. Perstumtos eilutės

Eil.	Rezultatas po eilučių sukeitimo
31	
32	
33	It is necessary that a loan is_a reliable_loan if each loan is_got_by debtor is_returned loan.
34	
35	
36	It is permitted that a debtor requests a loan.
37	
38	It is obligatory that a bank gives a loan.
39	
40	It is obligatory that a bank checks_validity_of a loan.
41	
42	It is necessary that an amount of_the loan is_not_greater_than 20000.

Po pakeitimo dokumentas yra išsaugotas kaip ketvirtoji (4) jo versija. Kad būtų galima grįžti prie trečiosios (3) versijos serveryje saugomos deltos:

=794 -100 =218 +%0D%0A%0D%0A%0D%0AIt is necessary that a loan is_a reliable_loan if each loan is_got_by debtor is_returned loan.

7.1.4. EILUČIŲ ĮRAŠYMAS

Įterpsime naują eilutę į dokumento pradžią (pirmąją poziciją) eilutės turinys:

It is necessary that a person owns at most 5 account.

Po pakeitimo dokumentas yra išsaugotas kaip penktoji (5) jo versija. Kad būtų galima grįžti į ketvirtąją (4) versiją serveryje saugomos deltos:

-57 =1112

7.1.5. REZULTATAS

Po įvairių duomenų modifikavimų operacijų serveryje saugoma naujausia TestSBVR.rules failo versija, kuri pažymėta kaip 5 versija. Visas duomenų failas pateiktas 5 lentelėje.

Lentelė 5. Penkta .rules failo versija

Eil.	TestSBVR.rules saugomas serveryje
1	It is necessary that a person owns at most 5 account.
2	
3	It is necessary that a loan owns exactly one bail.
4	
5	It is possible that a bail is_owned_by at most one loan.
6	
7	It is possible that a debtor gets at most 5 loan.
8	
9	It is necessary that the loan is_got_by exactly one debtor.
10	
11	
12	It is possible that a branch is_included_in at most one parent_bank.
13	
14	It is necessary that the parent_bank includes at least 4 and at most 10 branch.
15	

16	
17	It is necessary that an account is_included_in exactly one bank.
18	
19	
20	It is necessary that a person owns at least 1 account.
21	
22	It is necessary that the account is_owned_by exactly one person.
23	
24	
25	It is necessary that the loan is_given_by exactly one bank.
26	
27	
27	It is necessary that the owner owns at least 2 real_estate.
29	
30	It is necessary that the real_estate is_owned_by exactly one owner.
31	
32	
33	It is necessary that a loan is_a reliable_loan if each loan is_got_by debtor is_returned loan.
34	
35	
36	It is permitted that a debtor requests a loan.
37	
38	It is obligatory that a bank gives a loan.
39	
40	It is obligatory that a bank checks_validity_of a loan.
41	
42	It is necessary that an amount_of_the loan is_not_greater_than 20000.

Norint įsitikinti ar buvo teisingai apskaičiuoti skirtumai (deltos) irišsaugoti duomenų bazėje, o algoritmai teisingai iš deltų surenka dokumentą, visų pirma iš serverio parsisiųsime pirmąją dokumento versiją, kuri bus surinkta iš naujausio failo ir serijos deltų, kurios buvo pateiktos prie kiekvienos modifikacijos. Gauta pirmoji dokumento versija pateikta 6 lentelėje.

Lentelė 6. Iš deltų sudaryta pirmoji .rules failo versija

Eil.	Sudaryta pradine (1) TestSBVR.rules versija
1	It is necessary that a loan owns exactly one bail.
2	
3	It is possible that a bail is_owned_by at most one loan.
4	
5	It is possible that a debtor gets at most 3 loan.
6	
7	It is necessary that the loan is_got_by exactly one debtor.
8	
9	
10	It is possible that a branch is_included_in at most one parent_bank.
11	
12	It is necessary that the parent_bank includes at least 4 and at most 10 branch.
13	
14	
15	It is necessary that an account is_included_in exactly one bank.
16	
17	
18	It is necessary that a person owns at least 1 account.
19	
20	It is necessary that a person owns at most 5 account.
21	
22	It is necessary that the account is_owned_by exactly one person.
23	
24	
25	It is necessary that the loan is_given_by exactly one bank.
26	
27	
27	It is necessary that the owner owns at least 2 real_estate.
29	
30	It is necessary that the real_estate is_owned_by exactly one owner.
31	
32	

```
33  It is permitted that a debtor requests a loan.
34
35  It is obligatory that a bank gives a loan.
36
37  It is obligatory that a bank checks_validity_of a loan.
38
39  It is necessary that an amount of_the loan is_not_greater_than 10000.
40
41
42  It is necessary that a loan is_a reliable_loan if each loan is_got_by debtor
    is_returned loan.
```

Palyginus rezultatas pateiktus 6 ir 1 lentelėse matosi kad jie yra identiški. Atlikus bandymus su duomenų versijavimu matome, kad projektų išsaugojimas versijomis serveryje dirba korektiškai. Atlikus 5 skirtingus pakeitimų rinkinius, buvo galima grįžti į ankstesnę projekto versiją turint naujausią projekto versijos variantą ir ant jos paeiliui uždedant deltas. Gautas rezultatas yra be duomenų praradimo ir iškraipymo. Tuo galime įsitikinti sulyginę 1 ir 6 lentelėse pateiktus dokumentų duomenis.