KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ TINKLŲ KATEDRA

Ruslanas Sobolevas

Paskirstytos sistemos su testavimo scenarijais kokybės tyrimas

Magistro darbas

Darbo vadovas
Doc. dr. Gytis Vilutis

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ TINKLŲ KATEDRA

Ruslanas Sobolevas

Paskirstytos sistemos su testavimo scenarijais kokybės tyrimas

Magistro darbas

Recenzentas
Doc. dr. Danguolė Rutkauskienė
2011-05-

Darbo vadovas
Doc. dr. Gytis Vilutis
2011-05-30

Atliko
IFM-9/2 gr. stud.
Ruslanas Sobolevas
2011-05-30

Kaunas, 2011

ABSTRACT

TestTool is an e-learning system for testing theoretical and practical student knowledge. Main difference from other knowledge assessment tools is the ability to use interactive graphical components in testing process.

New administrative subsystem for TestTool was developed, it has dedicated resource repository and operation centre modules that make testing process more flexible. It also simplifies development and maintenance of new subsystems and system functions. These modules allow better adaptation of test process to customer needs and reduce implementation and later maintenance costs of various knowledge assessment features.

It is very important to have free e-learning systems that are of the same quality as similar commercial products. This work provides the details of main development and testing stages of TestTool administrative subsystem that helped to achieve positive results of subsequent subsystem quality assessment.

TURINYS

LENTELĖS

PAVEIKSLAI

ĮVADAS

TestTool – tai žinių testavimo sistema, skirta studentų teorinių ir praktinių sugebėjimų vertinimui bei įgūdžių formavimui. Iš kitų testavimo sistemų ją išskiria galimybė kurti grafiniais komponentais paremtus interaktyvius testus.

Paskirstyta sistema su žinių testavimo scenarijais buvo sukurta magistrinio projekto „TestTool sistemos išvystymo tyrimas ir įgyvendinimas“ metu. Plačiau apie šią sistemą bus pasakojama 2 skyriuje. Projekto tikslas buvo išanalizuoti esamą TestTool sistemos versiją, suformuluoti naujus reikalavimus ir sukurti naują TestTool Administravimo posistemę, kuri sudarytų sąlygas lengvam funkcionalumo praplėtimui bei neturėtų defektų būdingų ankstesniems versijoms. Posistemė yra sudaryta iš dviejų paskirstytų modulių, kurių vienas rūpinasi žinių testavimo proceso organizavimu, o kitas vykdo rezultatų, vertinimą, variantų ir kitų mokymosi resursų saugojimą. Tai leidžia tolygiai paskirstyti apkrovas bei tenkina naujai atsiradusius poreikius.

Magistrinio darbo tikslas yra sukurtų projekto metu TestTool administravimo posistemių kokybės įvertinimas.

Darbo uždaviniai:

1. Kokybės termino sampratos analizė.
2. Ankstesnių TestTool administravimo posistemių versijų trūkumų ir defektų analizė.
3. Naudojamų technologijų realizuotoje sistemoje įtaka sistemos kokybei ir bendrai sukurtai sistemai, jų panaudos privalumai ir trūkumai.
4. Sukurtų posistemių dalinis testavimas (plačiau apie testavimo apimtis 4.2 poskyryje).
5. Statinis objektinio programavimo metrikų ir charakteristikų tyrimas siekiant įvertinti sukurtą PĮ ir pateikti galimas tobulinimo rekomendacijas. Tyrimo metu vykdomas palyginimas su senesnę administravimo sistemos versija tam, kad nustatyti pasikeitusią kokybę.

Darbo struktūra:

- 1 skyriuje atliekama kokybės sampratos analizė.
- 2 skyriuje pristatoma sukurta paskirstyta žinių testavimo sistema, analizuojamos ankščiau sukurtų sistemų trūkumai, analizuojamos panaudotos technologijos ir jų teikiama nauda.
- 3 skyriuje pateikiamos paskirstytų modulių specifikacijos, reikalavimai, kūrimo rizikos, apribojimai ir t. t.

- 4 skyriuje yra pateikta nagrinėjamos sistemos architektūra – loginis posistemių vaizdas, sąryšis tarp paketų, pagrindinių klasių veikimo priklausomybės, duomenų vaizdas.
- 5 skyriuje yra aprašomas sistemos testavimo planavimas ir jo rezultatai.
- 6 skyriuje nagrinėjamos objektinės metrikos ir jų reikšmės, panaudoti pagalbiniai įrankiai, kurių pagalba buvo jos gautos. Pateikiamas šių charakteristikų tyrimas, kuriame reikšmės yra lyginamos su rekomenduotina norma ir su ankstesnėmis versijomis, teikiamos gerinimo rekomendacijos.
- 7 skyriuje yra pateiktos apibendrintos išvados ir rekomendacijos.

Vienas iš paskirstytos sistemos su žinių testavimo scenarijais sukūrimo tikslų siejasi su dabartinės žinių testavimo tvarkos pakeitimu į lankstesnę, kuri bus paremtą scenarijais (šablonais). Scenarijai leidžia organizuoti labiau valdomą ir užsakovo poreikius atitinkantį žinių testavimą.

Sukurta Administravimo posistemė turi būti efektyvi ir kokybiška. Kokybės ir efektyvumo paaiškinimas bus plačiau apžvelgtas toliau.

Sukurta Administravimo posistemė turi būti kuriama naudojant programavimo šablonus taip, kad ją būtų galima ateityje nesudėtingai plėsti bei integruoti su kitomis TestTool posistemėmis. Projekto metu buvo surinkti atsiliepimai apie buvusias sistemas ir suformuluoti reikalavimai naujos TestTool versijos kūrimui. Buvo parengtas ir realizuotas detalus TestTool serverio posistemės projektas, kuriame detaliai aprašoma projekto specifikacija, projektavimo dalis, apskaičiuotos kūrimo rizikos, apžvelgtos naudojamos technologijos ir metodai, atliktas realizuotų posistemių testavimas.

Tiriamuoju darbu siekiama įvertinti realizuotos programinės įrangos kokybę per testavimą ir programinės įrangos metrikų interpretavimą. Programinės įrangos metrikos gali parodyti realizuotų sistemų sudėtingumą, tolimesnio palaikymo kaštus ir apskaičiuoti galimus sunkumus norint toliau plėsti sistemą.

Literatūros apžvalgoje aptariami magistriniame projekte panaudotos technologijos ir metodai, taip pat aprašomas pasirinktas kokybės tyrimo metodas.

1. KOKYBĖS SAMPRATA PROGRAMINĖS ĮRANGOSĮ KURIME IR PALAIKIME

Programinės įrangos (toliau PĮ) kokybė gali būti suprantama įvairiai, dažniausiai tai vartotojo suformuluotų reikalavimų atitikimas. Iš kitos pusės PĮ kokybės kriterijai gali būti įvairūs. Tai ir išeities kodo standartų laikymasis, specifinių taisyklių taikymasis, kurie pagerina kodo skaitomumą bei atnaujinamumą, sumažina jo sudėtingumą ir t. t.

PĮ kokybės faktorius galima padalinti į šiuos kriterijus [1]:

- Suprantamumas – sukurtų posistemių paskirties aiškumas pagal dokumentaciją arba iš posistemių veikimo. Tam, kad sukurtas produktas būtų kokybiškas, labai svarbu, kad jau pradinėje stadijoje įvairūs reikalavimai ir specifikacijos būtų aiškiai apibrėžtos ir suformuluotos. Suprantamumas dar gali būti vadinamas aiškumu.
- Pilnumas – ar visas numatytas funkcionalumas yra realizuotas. Trumpumas – nėra besidubliuojančios arba perteklinės informacijos. Jei tokių dalių rasta, jos turi būti restruktūrizuotos į bendresnes procedūras. Tai liečia ne tik išeities kodus, bet ir susijusią dokumentaciją.
- Portatyvumas – tai lengvumas adaptuoti programą kitai aplinkai: kitai architektūrai, platformai, operacinei sistemai (pavyzdžiui veikimas ir Windows ir Linux operacinėse sistemose) arba jos versijai. Portatyvumas gali būti glaudžiai būti susijęs su suderinamumu, kuris turėtų būti aiškiai apibrėžtas reikalavimuose. Suderinamumas labai įtakoja kitus kokybės rodiklius – gali padidėti sistemos sudėtingumas, ir dėl to pablogėti prižiūrimumo rodykliai[2]. Nuoseklumas – visame programos kode ir dokumentacijoje naudojami bendri sutarimai, formatai ir ženklinimai.
- Prižiūrimumas – tolimesnės programos priežiūros savybė, parodo, kiek pastangų reikia sutelkti norint pakeisti programą, siekiant ištaisyti klaidas, realizuoti naujas galimybes arba funkcijas, siekiant pagerinti charakteristikas arba pritaikyti prie pasikeitusios aplinkos arba reikalavimų. PĮ sudėtingumas yra tiesiogiai susijęs su prižiūrimumu: kuo sudėtingesnė programa, tuo sunkiau yra vykdyti tokios sistemos priežiūrą.
- Testuojamumas – ar programos funkcionalumą įmanoma išbandyti arba patikrinti, ar galima naudoti automatizuotus įrankius sistemos funkcionalumui išbandyti. Paprastai šio rodiklio užtikrinimas prasideda dar projektavimo stadijoje ir vykdomas kūrimo metu.

- Naudojamumas – naudojimo paprastumas ir patogumas, dažniausiai labiau taikytinas grafinėms sąsajoms, tiriamieji objektai yra serverio posistemės, kurios neturi grafinės sąsajos, dėl to šis rodiklis yra nelabai svarbus.
- Patikimumas – tai programos veikimas klaidų ir defektų, o jei tokie randami, juos yra lengva ištaisyti.
- Efektyvumas – PĮ veikimas be žalingo poveikio aplinkinėms programoms, taip pat efektyvus aparatinių resursų naudojimas (atminties, procesoriaus, tinkle pralaidumo, laiko ir t. t.) vykdant savo užduotis.
- Saugumas – ar vartotojo duomenys yra pakankamai saugūs, ar priėjimas prie programinės įrangos valdymo ar duomenų yra apsaugotas? Ar yra realizuoti autentifikacijos, kodavimo ir panašūs mechanizmai?

Kokybės samprata iš vartotojo pozicijos gali būti susijusi su naudojimo patogumu – ar programos nustatymai yra pakankamai aiškūs, ar klaidos yra aiškios, ar programa daro tai, kam ji skirta, ar yra dokumentacija ir ar ji pilna, ar programos sąsajos yra pakankamai aiškios, kad joms nereikia papildomos dokumentacijos, ir ar programos operacijų veikimo uždelsimai tenkina vartotojo lūkesčius.

Ar sukurti moduliai atitinka daugumą apibrėžtų faktorių, susijusių su reikalavimų atitikimu ir veikimo korektiškumu, gali atsakyti įvairiapusis sistemos testavimas. Ar tenkina pasiektas posistemių sudėtingumas ir prižiūrimumo lygis parodo statinė išeities kodų metrikų analizė. Tiriamam projektui iš anksto nebuvo nustatyti specialūs kokybės reikalavimai, jie bus lyginami su norminiais ir ankstesnių versijų kokybe.

Kokybišką sistemą galima sukurti tik tada kai taikomas sklandus ir suderintas kūrimo procesas, kurio sudėtyje būtinai turi būti tokios dalys, kaip reikalavimų surinkimas, jų analizė ir aiškus formulavimas, planavimas, kūrimo kaštų ir rizikos įvertinimas, kuriamų sistemų išsamus projektavimas, suprogramuotų sistemų įvairiapusis testavimas. PĮ kūrimo proceso metų yra labai svarbus pagalbinių priemonių naudojimas, kurios leidžia efektyviai planuoti, valdyti projekto ir programavimo išteklius. Prie tokiu priemonių galima priskirti kodo versijavimo sistemas, kurios savo ruožtu gali būti surištos su projekto valdymo sistemomis bei įvairiais laiko planavimais.

Šios e-mokymo paskirstytos žinių testavimo sistemos kūrime buvo taikomi programavimo architektūriniai šablonai. Jų panaudojimas leido ne tik užtikrinti maksimalų komponentų dalių pakartotiną panaudojimą, bet ir pagerinti prižiūrimumo rodiklius, supaprastinant PĮ struktūrą, sutrumpinti kūrimo laiką [3]. Iš kitos pusės kiti šaltiniai [4] teigia, kad programavimo šablonų panaudojimas gali ir negatyviai įtakoti kai kuriuos kokybės

rodiklius. Pavyzdžiui mūsų sukurtose posistemėse panaudotas abstraktus fabrikas neigiamai įtakojo tokias kokybės charakteristikas, kaip aiškumas arba suprantamumas. Nepaisant to, jei programavimo šablonai teisingai pritaikyti ir tinka duotai programinei įrangai, jie pagerina PĮ charakteristikas. Iš kitos pusės kai kuriuos kokybės faktorius yra sudėtinga įvertinti dabar, kai kurie rezultatai pasirodys tik po to, kai sukurta PĮ bus naudojama realiose sąlygose pas užsakovą.

2. TIRIAMOS PASKIRSTYTOS E-MOKYMO SISTEMOS APRAŠYMAS

Šioje dalyje bus trumpai pristatyta TestTool žinių testavimo sistema ir sukurti jai moduliai, išdėstyti naujų modulių sukurtimo tikslai ir adresatas. Kaip jau buvo minėta tam, kad sukurti kokybišką PĮ, yra labai svarbų apibrėžti aiškius reikalavimus, o reikalavimų surinkimas yra neįmanomas be praeitų versijų ir panašių sistemų analizės. Labai svarbu aiškiai suprasti ankstesnių versijų trūkumus, kad nustatyti ar naujai sukurtoje PĮ pavykdo juos pašalinti ir įgyvendinti vartotojų norus. Taip pat trumpai apžvelgiami objektinio programavimo šablonai bei programavimo technologijos, kurios yra panaudotos paskirstytos sistemos modulių realizacijoje, jų privalumai ir trūkumai, įtaka sistemos kokybei.

2.1. E-MOKYMO PASKIRSTYTOS SISTEMOS PASKIRTIS

TestTool – tai žinių testavimo sistema, skirta studentų teorinių ir praktinių sugebėjimų vertinimui bei įgūdžių formavimui. Iš kitų žinių testavimo sistemų ją išskiria galimybė kurti grafiniais komponentais paremtus testus.

Magistrinio projekto metu yra realizuota nauja administravimo serverio posistemė, kuriuos pagalba galima iš esmės pakeisti žinių testavimo tvarką, padaryt ją lankstesnę ir labiau valdomą bei tuo pačiu kontroliuojamą.

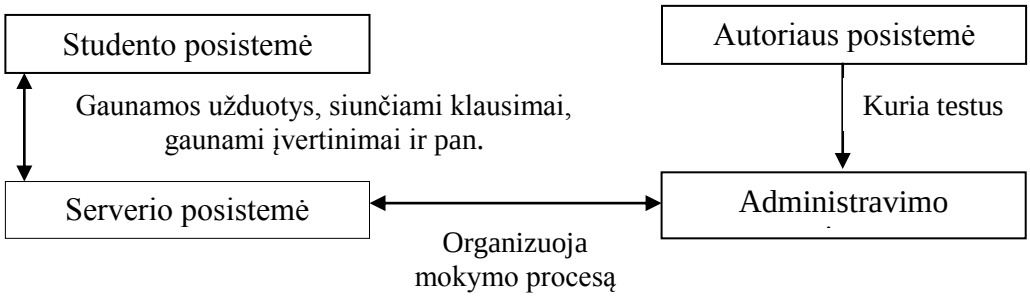
Kūrimo tikslas buvo išnagrinėti PĮ technologijas ir jų pagrindu išvystyti TestTool sistemą, padidinti jos technologiškumą ir funkcionalumą. Sukurtos naujos TestTool 5.3 administravimo posistemės, kurios praplečia 5.1 arba 5.2 sistemų galimybes. Sukurta nauja Testtool Administravimo posistemė sudaryta iš dviejų paskirstytų modulių, kurių vienas rūpinasi žinių testavimo proceso organizavimu, o kitas taikomas rezultatų, vertinimų, variantų ir kitų mokymosi išteklių saugojimui. Tai leidžia ne tik tolygiai paskirstyti apkrovas bet ir užtikrinti resursų pakartotinį panaudojimą.

Pavyko sukurti naują žinių testavimo tvarką, kuri yra lankstesnė, paremta scenarijais (šablonais), pavyzdžiui klausimų paruošimo scenarijai, klausimo variantų siuntimo tvarkos nustatymas pagal scenarijus, atsakymų vertinimo, kuriame nurodomi klausimų svoriai, scenarijus, įtakojantis vertinimo balą ir t. t. Dabar galima kurti testus ir egzaminus lanksčiau panaudojant esamus klausimų variantus ir klausimus.

Nauja Administravimo posistemė yra sukurta naudojant programavimo šablonus taip, kad ją būtų galima ateityje nesudėtingai plėsti bei integruoti su kitomis TestTool posistemis. Tam, kad suprasti ar sukurtose posistemėse pavyko pašalinti ankstesnių sistemų trūkumus, sekančiame skyriuje pateikiame ankstesnių versijų trumpą analizę.

2.2. ŠIUO METU NAUDOJAMOS TESTTOOL SISTEMOS ANALIZĖ

Analizės vykdymo metu buvo naudojama TestTool 5.1 sistemos versija, kurios posistemės yra pavaizduotos 1 pav.



1 pav. TestTool sistema

Įrankis skirtas universiteto vidiniam platinimui. Duomenų srautas tarp atskirų TestTool posistemių yra siunčiamas panaudojant RMI per SSL. Nors kodas yra atviras, lankstumas ribotas dėl prastos architektūros. Mokymosi resursų saugojimui yra naudojama objektinė duomenų bazė Ozone, kuri turi tokius trūkumus:

- Lėtas veikimas;
- Sudėtingas ir specifinis programavimas įgyvendinant galimybes;
- Klaidos programiniame Ozone variklyje, iš ko seka papildomos problemos programuojant;
- Prastas gamintojų palaikymas;
- Projektas senai neatnaujinamas;
- Sudėtinga plėsti ir tobulinti sistemą.

Dėl objektinės duomenų bazės Ozone ir sistemos architektūros dabar naudojama TestTool 5.1 versija veikia lėtai, turi klaidų bei netikslumų, turi ribotas studentų žinių vertinimo galimybes, vartotojo sąsaja nėra intuityvi, trūksta lankstumo. Praplėsti sistemai reikia daug laiko ir žmoniškųjų resursų.

5.1 versija buvo naudojama iki 2010 metų. Po to ją pakeitė 5.2 versija, kurioje buvo padaryti nežymus patobulinimai, kurie leido naudoti duomenų bazę MySQL. 5.2 taip pat kaip ir 5.1 turi daug defektų ir nepatogumų:

- Programa veikia nestabiliai;
- Perteklinis resursų panaudojimas, kuris seka prie kitos PĮ nekorektiško veikimo ir viso aparatinio serverio funkcijų ir paslaugų vykdymo sustabdymui;
- Programos funkcionalumas neatitinka šiuolaikiškų užsakovų reikalavimų;
- Funkcionalumas ribotas ir turi defektų;
- Klaidingas servisų veikimas;
- Perteklinis kodas ir sudėtinga kodo struktūra;

- Sudėtingas (praplėsti sistemai reikia daug laiko ir žmoniškųjų resursų) arba neįmanomas naujų funkcijų įdiegimo procesas;
- Sudėtingas serverio administravimas;
- Nėra aiškios dokumentacijos;

2.3. SUKURTOS PASKIRSTYTOS SISTEMOS TIKSLAI, NAUJUMAS IR ADRESATAS.

Dauguma žinių patikrinimo arba kitaip tariant žinių testavimo sistemų apsiriboja paprastais tekstiniais testais. TestTool sistema teikia testavimo galimybes, paremtas grafiniais komponentais. Išnagrinėjus esamos sistemos privalumus ir trūkumus įmonėje pereinama prie naujos sistemos versijos, kuri turi būti lanksti, greita ir lengvai praplečiama pagal poreikius. Po siūlomo projekto realizacijos TestTool testavimo organizavimo funkcinė bazė taptų turtingesnė. Atsirastų galimybė nustatyti testavimo tvarką (kompleksiniai testai, klausimų variantų scenarijai), lankstų vertinimą (vertinimo scenarijai), panaudojant įvairius kriterijus (klausimų įverčiai, priklausomybė nuo ankščiau gautų pažymių ir t. t.).

Projekto tikslas yra pagerinti TestTool sistemos funkcionalumą, pašalinti netikslumus, realizuoti naujas galimybes. Tai leistų išplėsti potencialių vartotojų ratą. Patobulinti moduliai ir naujai sukurti moduliai bus integruoti į TestTool 5.3 versiją. Sistema bus naudojama įvairių mokomųjų dalykų arba darbo sričių asmenų žinių patikrinimui.

Programinės įrangos užsakovas yra Kazys Baniulis, kurio organizacija yra kompiuterinių tinklų katedra. Kompiuterinių tinklų katedra yra nepelno siekianti valstybinė studijų ir mokslo įstaiga, priklausanti Kauno technologijos universiteto Informatikos fakultetui. Kauno technologijos universitetas turi Lietuvos Respublikos Konstitucijos ir Aukštojo mokslo įstatymo nustatytą specialų statusą. Pagrindinė veikla aukštasis išsilavinimas, mokslinė, kultūrinė ir šviečiamoji veikla.

2.4. PANAŠŪS E-MOKYMO SPRENDIMAI PASAULYJE

Žinių testavimo sprendimai dažniausiai paremti paprastų tekstinių arba grafinių blokų taikymu be galimybės manipuliuoti objektais ekrane. Šie moduliai įeina į sudėtingesnes struktūras, turinčias ir kitų funkcijų, tiesiogiai nesusijusių su žinių testavimų. Tokie produktai pateikiami vartotojams kaip kompleksinės mokymosi sistemos, iš kurių galime išskirti Moodle, WebCT.

WebCT[5] – komercinė mokymosi aplinka su integruotais moduliais ir įrankiais. Kadangi aplinka mokama, tai uždeda tam tikrus apribojimus naujų modulių rašymui ir

pritaikymui. Esamų modulių naudojimas yra intuityvus ir nesudėtingas, administravimas pakankamai lankstus.

Moodle[6][7] – atvira, modulinė objektiškai orientuota dinaminė mokymosi aplinka, kuri parašyta panaudojant PHP programavimo kalbą ir SQL duomenų bazes. Moodle platinama pagal GNU GPL licenciją, kuri leidžia kitiems programuotojams plėsti sistemą. Kadangi Moodle turi modulinę architektūrą, tai galima prijunginėti įvairius modulius (pvz. kurso elementus, įvairius blokus, filtrus, ataskaitos ir t. t.).

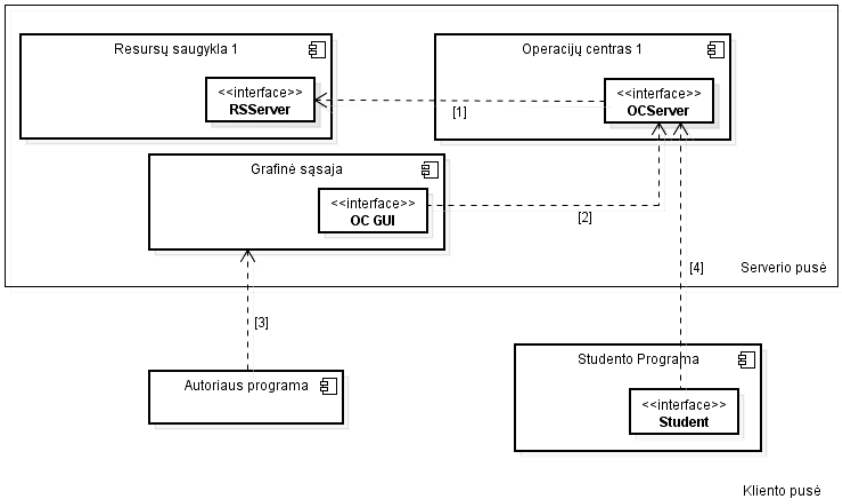
2.5. E-MOKYMO SISTEMŲ POREIKIS IR SITUACIJOS LIETUVOJE ĮVERTINIMAS

96% (2009m. duomenys pagal Statistikos departamentą) Lietuvos įmonių naudoja kompiuterius. Dabar Lietuvoje asmenų žinių testavimui naudojamos tokios pat priemonės kaip ir pasaulyje. Nors didesnės įmonės kartais naudoja savo siaurai sričiai pritaikytus įrankius, kurie sukuriami įmonės viduje. Paprastai tokie įrankiai neišsiskiria ypatingu funkcionalumu, bet patenkina įmonės keliamus reikalavimus žinių testavimui.

Iš viešai prieinamų Lietuvoje sukurtų žinių testavimo sistemų galima būtu paminėti testuok.lt [8]. Testuok.lt – komercinė žinių testavimo sistema, nors mokslo įstaigoms yra suteikta galimybė naudoti šią sistemą nemokamai. Visgi tai apriboja šio produkto naudojimo sritis, nes jis negali būti adaptuojamas prie įvairesnių vartotojo poreikių, galima naudotis tik jau realizuotomis galimybėmis. Naudotis įrankių lengva ir paprasta.

2.6. SUKURTŲ E-MOKYMO SISTEMOS MODULIŲ ANALIZĖ

2.6.1. Naujai sukurtos TestTool 5.3 versijos funkcionalumo santrauka



2 pav. Apibendrinta TestTool sistemos struktūra

Sukurtos posistemės komponentų diagrama parodyta *2* pav. Šioje schemoje 5.3 versijos serverio posistemės pažymėtos pilkai, jame OC – operacijų centras rūpinasi žinių testavimo

proceso organizavimu, pilnai išpildo studento ir administravimo funkcijas. RS - resursų saugykla, RVS - rezultatų ir vertinimo saugykla. Žymėjimai diagramoje:

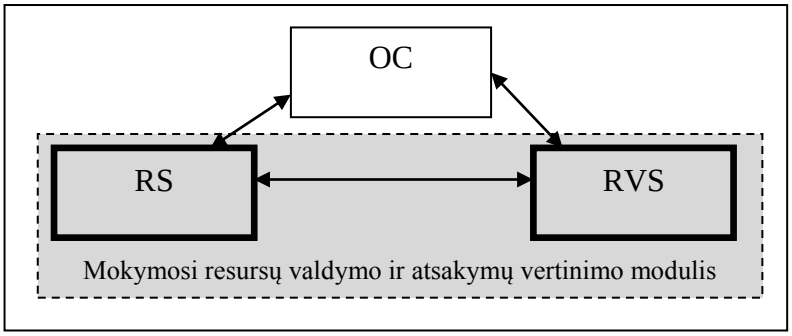
1 – Saugo / gauna mokymosi resursus.

2 – Valdomas testavimo procesas.

3 – Siunčiami / parsiunčiami klausimai.

4 – Gaunamos užduotys / siunčiami studento atsakymai.

Schemoje žemiau parodytas detalizuotas resursų saugyklos modulis bei ryšis tarp OC ir RS sudėtinių modulių.



3 pav. TestTool serverio posistemė

„Resursų saugykla“ ir „rezultatų ir vertinimo saugykla“ (išskirta pilkai) sudaro mokymosi resursų valdymo ir atsakymų vertinimo modulį. Nors šią posistemę sudaro du moduliai, jie įgyvendinami kartu ir šiuo metu veikia kaip neatskiriamas modulis, todėl jis bus vadinamas tiesiog kaip Resursų saugykla (RS). Resursų saugyklos funkcijos išvardintos 1 lentelėje.

1 lentelė. Resursų saugyklos funkcionalumas

Resursų saugykla	
Resursų saugykla	Rezultatų ir vertinimo saugykla
Atlieka įvairių mokymosi resursų saugojimą, atnaujinimą, arba grąžinimą pagal užklausą. Atlieka tarpininko funkciją tarp sistemos komponentų ir duomenų bazės. Mokymosi resursų paruošimas panaudojant scenarijus (gali būti taikomi įvairių konvertavimo funkcijų realizacijai) Resursų paieška	Atlieka studento atsakymų, testo klausimų ir variantų saugojimą, atnaujinimą, grąžinimą pagal užklausą. Atlieka studento atsakymo vertinimą ir rezultato saugojimą. Variantų vertinimas atliekamas panaudojant vertinimo scenarijus.

Variantų vertinimui naudojami vertinimo scenarijai, kurių sudėtyje yra 5.1 administravimo sistemos vertinimo modulis.

OC modulio funkcijos: Administratorių arba super-administratorių valdymas, kursų tvarkymas, grupių tvarkymas, resursų saugyklų tvarkymas, resursų saugyklų funkcijų vykdymas pagal pasirinktą resursų saugyklą (RS klientas), klausymų, testų, egzaminų tvarkymas, klausimų variantų pateikimo organizavimui gali būti taikomas specialiai realizuotas scenarijus, visų tipo objektų paieškos galimybė.

Papildomos galimybės:

- Klausimo variantų (egzaminų scenarijus) grąžinimas studento programai panaudojant specialius egzamino scenarijus.
- Galimybė nustatinėti naudojamus scenarijus, lankstus vertinimas remiantis vertinimo scenarijais ir klausimų svoriais.
- Priėjimas prie resursų saugyklų per operacijų centrą.
- Lengvas mokymosi resursų paruošimo, vertinimo šablonų diegimas.
- Galimybė instruktoriams priskirti >1 resursų saugyklą.
- Klausimų svoriai, vertinimas pagal svorius (jei užsakovas realizavo atitinkamą scenarijų).

2.6.2. Sistemos funkcijų išplėtimo galimybės ir naudojamos technologijos

Sistemos funkcijas galima lengvai plėsti kuriant įvairius klausimų vertinimo, mokymosi medžiagos ir/arba klausimų paruošimo scenarijus. Šie scenarijai gali naudoti esamą posistemių funkcionalumą arba didinti funkcionalumą pagal užsakovo pageidavimus.

Maksimaliam suderinamumui tarp kuriamų ir esamų sistemos modulių užtikrinimui naudojama Java programavimo kalbą. Todėl nagrinėjant įrankius, technologijas ir priemones, atsižvelgėme tik į tokius, kurie yra suderinami su Java programavimo kalba. Pritaikytas praktikas ir technologijas plačiau apžvelgsime tolesniuose skyreliuose.

2.6.3. OC modulio architektūros technologiniai sprendimai

Per paskutinius metus objektinės programinės įrangos kūrimas ženkliai pasikeitė [9]. Tobulėjo technologijos, bei to pasekmėje programavimo kalbos. Atsirado daug naujų programavimo strategijų ir šablonų, anotacijų ir pagalbinių priemonių, kuriuos padeda kurti efektyvias aplikacijas, bei programinius kompleksus[10]. Kuriamai programinei įrangai, ypač verslo sektoriuje, visuomet buvo keliami aukšti kokybės bei sąnaudų reikalavimai. Tokiems poreikiams tenkinti naudojama Java EE (Java Enterprise Edition) su įvairiais architektūriniais šablonais (angl. design patterns). Realizuojant kuriamą projektą naudosime Java SE su tam tikrai užduočiai tenkinančius architektūriniais šablonais, kurie naudojami su Java EE. Priežastis kodėl naudojami architektūriniai šablonai [11] yra tame, kad tai leidžia daug kartų

panaudoti sprendimus ir nustatyti bendrą terminologiją tarp skirtingų programos modulių. Jie (šablonai) suteikia programuotojams aukštesnio lygio galimybes sprendžiant problemą ir architektūros kūrimo procesą ir jo dizainą iš objektiškos pusės.

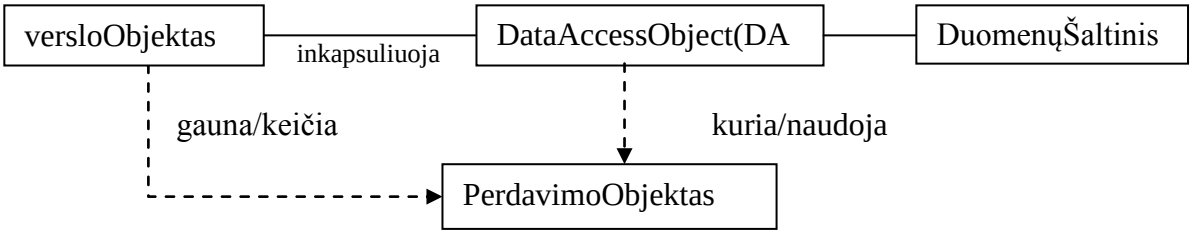
Modulių architektūroje panaudojamas Data Access Object (duomenų priėjimo objektas), Transfer Access Object (TO, plačiau 2.6.5 skyriuje) architektūrinius šablonus.

2.6.4. Architektūrinis šablonas Data Access Object

Priėjimo mechanizmas darbui su duomenų šaltiniu realizuotas remiantys DAO (angl. Data Access Object). DAO[12] – tai vienas iš standartinių J2EE architektūrinių modelių (angl. pattern). Aplamai DAO dažniausiai naudojamas tam, kad atskirti žemo lygio (angl. low-level) duomenų priėjimo logiką nuo aukšto lygio verslo logikos (angl. high-level business logic).

Panaudojant DAO dirbama su paprastesne sąsaja. Taip pat DAO paslepia duomenų šaltinio realizaciją nuo klientų. Tai leidžia pridėti lankstumą serveriui ir kitiems programuotojams, yra nebūtina žinoti Hibernate karkaso (kuris yra naudojamas modulių sudėtyje) ir to, kaip yra realizuotas sąryšis su duomenų šaltiniu.

Praktiškai DAO veikia kaip funkcija-adapteris tarp komponento ir duomenų šaltinio. Todėl keičiant duomenų šaltinio realizacijos schemas DAO sąsaja nesikeičia ir neįtakoja duomenų šaltinio klientus arba verslo-komponentus. 4 pav. parodyta klasių diagrama su santykiais DAO modelyje.



4 pav. Teorinė klasių diagrama, parodanti santykius DAO modelyje

VersloObjektas – parodo duomenų klientą. Tai objektas, kuriam iš šaltinio reikia gauti duomenis arba juos saugoti. DataAccessObject – pirminis DAO šablono komponentas, abstrahuoja priėjimo prie duomenų realizaciją, teikia skaidrų priėjimą prie duomenų šaltinio. VersloObjektas taip pat perduoda atsakomybę už pakrovimo operacijų ir saugojimo operacijų vykdymą DAO objektui. DuomenųŠaltinis – parodo duomenų šaltinio realizaciją (pvz. duomenų bazę arba kitą sistemą). PerdavimoObjektas – objektas, kuris naudojamas duomenų

perdavimui, DAO taip pat gali naudoti jį duomenų gražinimui iš kliento. DAO gali priimti duomenis iš kliento objekte PerdavimoObjektas (apie tai kaip yra realizuojamas šis objektas plačiau 2.6.5 skyriuje) duomenų pakeitimui duomenų šaltinyje.

DAO realizacija turi sekančius komponentus: DAO gamyklinę (angl. factory) klasę, DAO sąsają (angl. interface), konkrečią klasę, kuri realizuoja DAO sąsają, duomenų perdavimo objektai (angl. Data transfer objects) arba kartais vadinami reikšmės objektais (angl. value object).

„Abstrakti gamykla“ yra sąsaja objektų šeimų kūrimui be konkrečių klasių nurodymo. Kiekvienos šeimos objektai turi būti logiškai surišti tarpusavyje. Klientas kuria objektus gamyklos pagalba ir tokiu būdu manipuliuoja su objektais aukštesniam abstrakcijos lygyje. Modulio architektūroje naudojamas būtent „Abstraktaus gamyklos“ šablonas, nes jis turi daugiau galimybių plėsti aplikacijai.

Konkretizuosim DAO funkcijas sukurtoje sistemoje:

- Visas priėjimas prie duomenų bazės sistemoje vykdomas per DAO.
- Kiekvienas DAO egzempliorius atsako tik už vieną pirminį objektą arba esybę, t. y. jei objektas turi nepriklausomą gyvenimo ciklą, tai jis privalo turėti savo DAO. Nors realiai ne visur šis teiginys yra išpildomas, kadangi esybės yra glaudžiai susijusios.
- DAO atsako už objekto kūrimo, skaitymo (pagal pirminį raktą), trynimo, atnaujinimo operacijas.
- DAO gali leisti užklausas, kurie remiasi kriterijum, kuris skiriasi nuo pirminio rakto. (pvz. paieškos metodas, kuris grąžins kolekciją.)

DAO modelio naudojimo privalumai:

- Suteikia skaidrumą. Objektai gali naudoti duomenų šaltinį, nežinant apie konkrečias realizacijos detales. Priėjimas yra skaidrus, nes realizacijos detalės paslėptos DAO viduje.
- Palengvina migraciją. DAO objektų lygis leidžia lengvai pereiti aplikacijai prie kitos duomenų bazės, nes klientai nežino sąryšio tarp duomenų bazės veikimo realizacijos. Iš čia seka, kad pakeitimai bus reikalingi tik DAO lygyje.

- Sumažina kodo sudėtingumą. Kitų komponentų kodas supaprastėja, nes visi veiksmai susiję su duomenų bazę realizuojami DAO lygyje, o ne komponentuose. Tai padidina programavimo našumą.

2.6.5. Architektūrinis šablonas Transfer Object

Esmė ir kylanti problema[13]:

Kai dvi sistemos (arba aplikacijos) bendrauja tarpusavyje susiduriama su tokia situacija, kai aplikacijos klientai tarpusavyje keičiasi duomenimis (šie duomenys dar bus vadinami verslo-duomenimis) su modulių komponentais. Kai kurie iš serverio verslo-komponentų grąžina duomenis klientui, klientas kviečia gavimo (angl. get) metodus tiek kartų, kol negaus visų reikiamų atributų reikšmių.

Duomenų valdymo komponentai naudojami daugelių vartotojų vienu metu, tuo pat metu jie yra transakcijiniai (angl. transactional) objektai ir pateikia išlaikomuosius (angl. persistence) duomenis. Duomenų valdymo komponentas teikia atributų reikšmes panaudodamas metodą *accessor* (kartais angl. vadinamas *getter* metodu arba *get*), kiekvienam atributui.

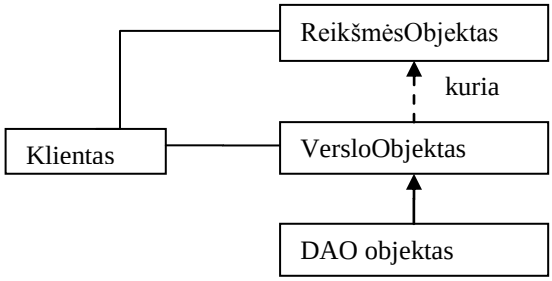
Kiekvienas objekto metodo iškvietimas ir sesijos komponentui, ir duomenų valdymo komponentui potencialiai yra nuotolinis. Tokiu būdu šitie nuotoliniai iškvietimai panaudoja tinklo lygį nepriklausomai nuo atstumo tarp kliento ir serverio komponento (net jei klientas ir komponentas randasi toje pačioje JVM, OS arba fiziniam kompiuteryje).

Šioje vietoje kyla šios problemos:

- Augant šių metodų iškvietimo skaičiui, sistemos greitaveika mažėja. Bet naudoti keletą *get* metodų, kurie grąžina vieno atributo reikšmę yra neefektyvu.
- Jei sesijos metu kliento komponentas persiunčia duomenų objektą, kuris savo ruožtu duomenų pasiekiamumui panaudoja papildomas klases, tai šio atveju šios klasės būtinos kliento pusėje. Tokiu būdu padidėja kliento dydis bei sprendimas tampa neracionalus, kadangi dalis serverio failų turi būti kliento pusėje.

Šią problemą spendžiame, panaudojant perdavimo objekto (angl. Transfer Object) programavimo šabloną, kuris inkapsuliuoja verslo-duomenis. Duomenų perdavimui ir išskleidimui perdavimo objektas naudoja vieną metodo iškvietimą. Kai klientas kreipiasi į komponentą užklausdamas verslo-duomenis, šis komponentas gali sukurti perdavimo objektą, įrašyti į jį savo atributų reikšmes ir perduoti klientui.

Paprasčiausioje formoje perdavimo objekto esmę pavaizduoja klasių diagrama 5 pav.



5 pav. Šablono „Perdavimo objektas“ klasių diagrama

Kaip parodyta šioje diagramoje, Perdavimo objektas kuriamas pagal komponento reikalavimą ir grąžinamas nutolusiam klientui.

Čia *Klientas* yra klientinės sistemos komponentas arba įvairūs verslo objektai (angl. BusinessObject).

VersloObjektas šiame programavimo šablone vykdo rolę, kuri gali būti vykdoma sesijos komponentu, duomenų valdymo komponentu arba Duomenų Priėjimo objektų (angl. Data Access Object, DAO). Verslo Objektas atsako už Perdavimo Objekto kūrimą ir grąžinimą klientui pagal užklausą. Verslo Objektas taip pat gali priimti duomenis iš kliento kaip Perdavimo objektą ir naudoti šios duomenis atnaujinimo operacijoms.

Perdavimo objektas (angl. *Transfer Object*) yra serializuotas Java-Objektas. Klasė PerdavimoObjektas gali realizuoti konstruktorių, kuris priima visus reikiamus atributus Perdavimo Objekto kūrimui. Konstruktorius gali priimti visas duomenų valdymo komponentų atributų reikšmes, kurių saugojimui sukurtas Perdavimo Objektas. Dažniausiai Perdavimo Objektas realizuojamas taip, kad po sukūrimo jo negalima būtų pakeisti, bet ši realizacija gali būti skirtinga priklausomai nuo sistemos-kliento poreikių, bet tada Perdavimo Objektas turi savyje turėti atitinkamus metodus tokiems veiksmams realizuoti.

Realizavimo strategijos:

- Strategija Atnaujinamas Perdavimo Objektas (angl. Updatable Transfer Objects) ir strategija Keletas Perdavimo Objektų (angl. Multiple Transfer Objects). Šios dvi strategijos panaudojamos tada, kai komponentas realizuotas kaip sesijos komponentas arba duomenų valdymo komponentas. Būtent šią strategiją ir naudojame sukurtų modulių architektūroje.
- Strategija Esybė Paveldi Perdavimo Objektą (angl. Entity Inherits Transfer Object) ir strategija Perdavimo Objektų Fabrikas (angl. Transfer Object Factory) naudojamos tik tais atvejais kai Verslo Objektas realizuotas kaip duomenų valdymo komponentas. Panašūs būdas naudojamas buvusioje TestTool versijoje.

Kadangi dauguma sukurtų paskirstytos posistemės modulių komponentų realizuoti kaip duomenų valdymo komponentai, tai naudojame keletą atnaujinamų perdavimo objektų.
Šio būdo naudojimo privalumai yra:

- Supaprastinamas duomenų valdymo komponentas ir nuotolinė sąsaja.
- Perduodama daugiau duomenų kviečiant mažesnį kiekį užklausų.
- Sumažinamas kliento dydis.
- Sumažinamas tinklo srautas.
- Sumažinamas kodo dubliavimas.

Šis šablonas turi ir neigiamų pusių:

- Gali būti teikiami pasenę „perdavimo objektai“.
- Dėl sinchronizacijos ir versijų kontrolės realizacijų gali padidėti aplikacijos sudėtingumas.
- Lygiagretus priėjimas kartais gali būti nekorektiškas.

2.6.6. Duomenų saugojimo technologijos ir Hibernate

Serializacija (angl. Serialization) – yra Java duomenų ir objektų perdavimų integruotas mechanizmas.

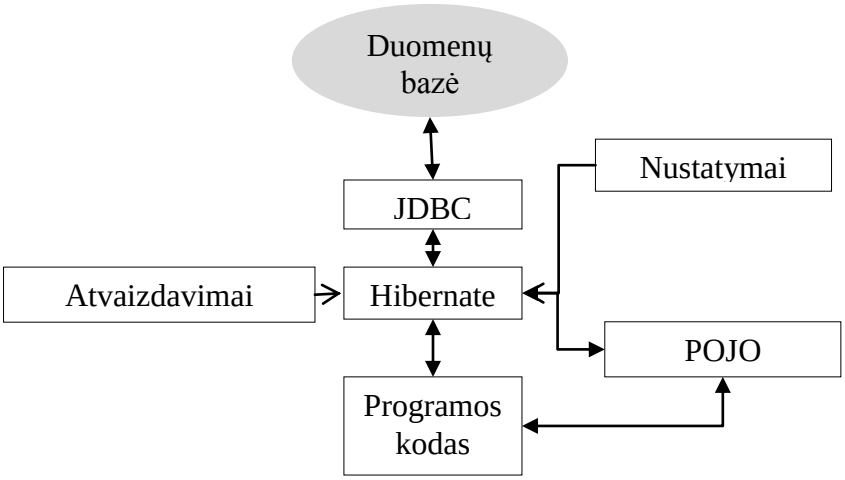
Duomenų išlaikomumas (angl. Persistence)[14] tai veiksmas, kuris Java kalboje paprastai reiškia duomenų saugojimą reliacinėje duomenų bazėje panaudojant SQL.

Java Database Connectivity (JDBC)[15] – Aplikacijų programinė sąsaja (angl. Application programming interface, API) buvo kuriamas darbui su reliaciniais duomenų bazėmis. Šios technologijos minusas yra tame, kad nėra mechanizmų, kurie atvaizduotų reliacinius duomenis į objektus, todėl kompensuojant šitą trūkumą žymiai padidėja kodo apimtis.

ORM (angl. Object relational mapping), tai automatizuotas ir skaidrus duomenų objektų atvaizdavimas iš Java aplikacijos į reliacines duomenų bazės lenteles, panaudojant meta duomenis, kurie aprašo atitikmenis tarp tų objektų ir duomenų bazės. Java Persistence API[2] (sutrump. angl. JPA) apjungia savyje objektų serializaciją su galimybe dirbti su objektais pagal objektinio programavimo modelį. Tuo pačiu lieka galimybė kombinuoti priėjimą prie duomenų reliaciniame duomenų lygyje (pagal ORM) kaip JDBC technologijoje, visa tai leidžia pasiekti didesnę greitaveiką ir lankstumą, palyginus su JDO (angl. Java Data Objects), specifikacija, kuri yra viena iš progresuojančių technologijų, kuri leidžia naudoti ne tik reliacines, bet ir objektines duomenų saugyklas.

Java Pesistence API turi daug įvairių realizacijų, viena iš jų yra Hibernate[16] JPA. Hibernate skirta ORM užduočių sprendimui.

Java Hibernate yra gana nesudėtingas įrankis su daugybę galimybių ir funkcijų, kurios palengvina programavimo eigą. Jis teikia skaidrų palaikymą duomenų išliekamumui (angl. persistence) paprastiems (trump. angl. POJO) Java objektams. Hibernate ne tik išsprendžia užduotis, susijusias su Java klasių ir duomenų bazės lentelių surišimu, bet ir teikia įrankius automatinei lentelių generacijai ir atnaujinimui. Tokiu būdu taupomas laikas, kuris paprastai skiriamas rankiniam SQL ir JDBC kodo rašymui. Prisijungimo prie duomenų bazės bei kiti hibernate nustatymai įrašomi specialiam xml formato faile. Transakcijos realizuojamos Hibernate‘o transakcijomis ir integruotu sesijos pagal reikalavimą (angl. session-per-request) funkcionalumu. Naudojant Hiberante nereikia J2EE aplikacijų serverio arba kitokių specialių aplinkų. Hibernate rolė Java aplikacijoje parodyta 6 pav.[11].



6 pav. Hibernate vieta aplikacijoje

Čia POJO – (angl. Plain Old Java Object) – paprastas java-objektas, nepaveldėtas nuo kurio nors specifinio objekto ir nerealizuojantis jokių tarnybinių sąsajų neskaitant tų, kurie reikalingi verslo modeliui organizuoti.

„Nustatymai“ – Hibernate konfigūracijos

„JDBC“ – tvarkyklė, kurią Hibernate naudoja priėjimui prie duomenų bazės (schemoje „Duomenų bazė“).

„Atvaizdavimai“ – tradiciniai Java objektai konvertuojami į reliacines esybes duomenų bazėje.

„Programos kodas“ – klientinė PĮ.

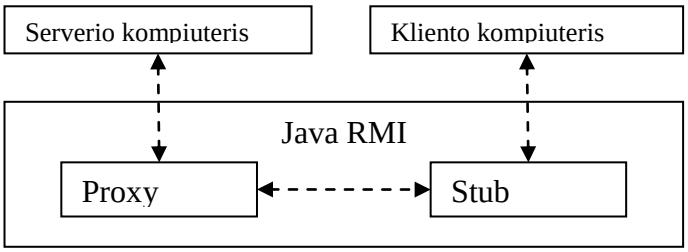
Tam, kad surišti modulį su Hibernate, naudojama Hibernate Session Factory (fabriko) Metodas[11] – tai yra programavimo šablonas objektų kūrimui (angl. creational pattern). Šis projektavimo šablonas yra sąsaja tam tikros klasės egzempliorių kūrimui. Kūrimo metu

palikuoniai gali nustatyti, kurią klasę inicijuoti. Kitais žodžiais tariant, fabrikas deleguoja objektų kūrimą tėvų klasės palikuoniams. Tai leidžia naudoti aplikacijos kode ne specifines klases, o manipuliuoti abstrakčiais objektais aukštesniam lygyje. Šio šablono naudojimo pliusas yra tame, kad objektų kūrimo kodas tampa universalesnis, nepririštas prie konkrečių klasių, o operuojantis tik konkrečia sąsaja.

Hibernate įrankių rinkinys platinamas pagal LGPL licenciją. JPA gali dirbti su skirtingomis duomenų bazėmis, tokiomis kaip: Oracle, DB2, Microsoft SQL Server, Sybase, MySQL, PostgreSQL, TimesTen, HypersonicSQL, SAP DB, InterSystems Cache, Informix, Interbase, Microsoft Access, Firebird, SQLite.

2.6.7. Bendravimas tarp modulių, Java RMI

Paskirstytų objektų architektūros realizavimui naudojama priemonė Java RMI (angl. Remote Method Invocation)[17]. Veikimo principas pavaizduotas 7 pav.



7 pav. RMI veikimo principas

Sugeneruojamos dvi papildomos klasės: proxy‘is ir stub‘as, tuomet klientas bendrauja su stub‘u, o serveris su proxy‘iu. Tai paslepia bendravimą tarp serverio ir kliento, tiek klientui, tiek serveriui atrodo, kad dirbama tame pačiame kompiuteryje.

Java RMI naudoja du protokolus[18]: Java objektų serialiazaciją (angl. Java Object Serialization) ir HTTP. Objektų serializacijos protokolas naudojamas kai reikia apdoroti prisijungimą ir gražinti duomenis. HTTP protokolas naudojamas prisijungimo inicijavimui ir gauti duomenų gražinimą, jei tai bus reikalaujama.

2.6.8. Log4j ir veiksmų sekimas.

Tam, kad visi veiksmai ir įvykiai butų susekti ir pažymėti atitinkame faile, naudojamas Log4j įrankis. Tai biblioteka, kuria sukūrė Apache bendruomenė, ji leidžia parašyti Java kode bibliotekos iškvietimus ir toliau išsamiai konfigūruoti sekimo lygį ir parametrus nekeičiant jau parašyto kodo.

Tam, kad nurodyti veikimo parametrus[19] naudojamas konfiguracinis (nustatymų) failas, kuriame nustatomi sekimo (angl. logging) parametrai.

2.6.9. Sistemų testavimo technologijos

Sistemų testavimas yra programinės įrangos (PĮ) kokybės užtikrinimo proceso dalis[20]. Pirminis testavimo tikslas yra identifikuoti PĮ klaidas taip, kad juos galima būti atrasti ir ištaisyti. Testavimas negali garantuoti, kad produktas funkcionuos visomis sąlygomis, bet tik gali įrodyti, kad tam tikromis sąlygomis jis veikia netinkamai[21].

Metodų teisingumas tikrinamas panaudojant jUnit[22] testus. jUnit – tai biblioteka, kuri palengvina ir automatizuoja testavimo procesą. Testui įvykdyti surašoma speciali testinė klasė. Metodas assertTrue tikrina, ar rezultatas yra teisingas. Taip pat galimi kiti rezultato tikrinimo metodai assertEquals, assertFalse, assertNull, assertNotNull, assertSame. Tam, kad sujungti visus testus galima pasinaudoti klase TestSuite su jos metodu addTest. Visų testų paleidimui reikia pasinaudoti TestRunner. Yra tekstinis junit.textui.TestRunner ir grafinės versijos – junit.swingui.TestRunner, junit.awtui.TestRunner. Naudosime integruota į Eclipse IDE sąsają įrankį.

Sudėtingesnėms klasėms testuoti gali būt panaudojami tokie metodai kaip setUp ir tearDown. setUp metodas gali inicijuoti vieną arba daugiau testuojamų klasių egzempliorių naudojimui keliuose TestCases. tearDown metodas išjungia paleistus per inicializaciją resursus.

Testavimo metodai sudarinėjami panaudojant baltos dėžės arba juodos dėžės strategijas.

2.7. SKYRIAUS IŠVADOS

1. Naujų modulių sukūrimas yra būtinas kadangi senų posistemių praplėtimas yra sudėtingas, veikimo kokybė, prižiūrimumo sąnaudos ir funkcionalumas jau netenkina vartotojų poreikius.
2. Kadangi testavimo sistema beveik nenusileidžia analogiškiems e-mokymo sistemoms, nuspręsta kurti naują versiją, bet neatsisakyti sistemų palaikymo ir naudojimo.
3. Architektūrinių šablonų panaudojimas leido struktūrizuoti ir supaprastinti kodo struktūrą, suteikė skaidrumą, sumažino kodo dubliavimą (užtikrinamas pakartotinis panaudojimas), tokiu būdu pagerinant suprantamumą, prižiūrimumą, patikimumą ir efektyvumą.
4. Hibernate karkasas panaudojimas įgalino lengvos migracijos galimybę tarp skirtingų duomenų bazių gamintojų, automatizuotą Java objektų atvaizdavimą duomenų bazėje, sutaupė programuotojų laiką realizuojant transakcijas ir sesijų palaikymą. Hibernate panaudojimas padeda pasiekti minėtą ankščiau portatyvumą.

5. Sistemos realizacijoje panaudoti veiklos sekimo įrankiai leido pilnai suprasti apie sukurtų sistemų vykdomus veiksmus kiekvienu laiko momentu. Tai pagerino PĮ testuojamumą, prižiūrimumą bei suprantamumo rodiklius.
6. Realizuojant naujus TestTool modulius buvo siekiama laikytis kelių sluoksnių architektūros panaudojant įvairias technologijas.

3. PASKIRSTYTŲ E-MOKYMO SISTEMOS MODULIŲ SPECIFIKACIJA

Tam, kad sukurti kokybišką sistemą yra labai svarbu korektiškai ir tiksliai apibrėžti visus sistemos reikalavimus.

Skyriuje bus pateikiama ši informacija: 1 šio skyriaus dalyje yra suformuluoti projekto apribojimai. 2 dalyje paminėtos kiti svarbus faktai – tokie, kaip pritaikoma licencija, numatomas komercinis sistemų panaudojimas. 3 dalis skirta apibrėžti funkcinius reikalavimus/ 4 šio skyriaus dalis skirta aprašyti nefunkcinius reikalavimus. 5 dalis aprašo šios specifikacijos žinomi netikslumai arba problemos. 9 poskyryje buvo apskaičiuotos kūrimo rizikos, tai labai svarbus žingsnis bendrame kūrimo procese, kadangi bet koks neplanuotas įvykis projekto metu gali labai rimtai įtakoti projekto kūrimą ir to pasekmėje jo kokybę. 6, 7 dalis nustato projekto kainos apskaičiavimo taisykles ir reikalavimus dokumentacijai. 8,9 dalis apibendrina apibrėžtas idėjas ir sprendimus. 10 dalis Nustato reikalavimus architektūrai. Išplėstai parodo modulių architektūrą: DAO šablonų realizaciją su realiom klasėm, OC ir RS programinių sluoksnių pasiskirstymą, duomenų bazės esybių diagramas, paketų struktūra (kadangi jie bus analizuojami metrikų analizėj). Detaliai architektūros nespecifikuosime, kadangi sistema jau sukurta, o magistriniame darbe pateiktam tyrimui detali architektūros specifikacija yra nereikalinga. 11 dalyje pateikiamos apibendrintos išvados.

3.1. PROJEKTO APRIBOJIMAI

3.1.1. Apribojimai serverio modulių veikimui

Programinė įranga veikia aplinkoje, kur yra įdiegta java vykdymo aplinka (JAVA 1.6). Programinės įrangos serverio posistemėms reikalinga paleistas rmiregistry aplikacija 1099 prievadu. Taip pat turi būti atidarytas atitinkamas prievadas, ant kurio paleista duota aplikacija (šį prievadą nustato duotos aplikacijos konfigūracijos failas).

3.1.2. Diegimo aplinka

Diegimo aplinkoje turėtų būti užtikrintos sąlygos, kurios aprašytos reikalavimų skyriuje „Apribojimai serverio modulių veikimui“. Kitų specialių apribojimų nėra.

3.1.3. Bendradarbiaujančios sistemos

8 pav. Pavaizduotos bendradarbiaujančios sistemos, žymėjimai:

- nauji moduliai
- esami moduliai, kuria kiti programuotojai
- bendravimo ryšis per programines sąsajas (vienpusis arba abipusis)

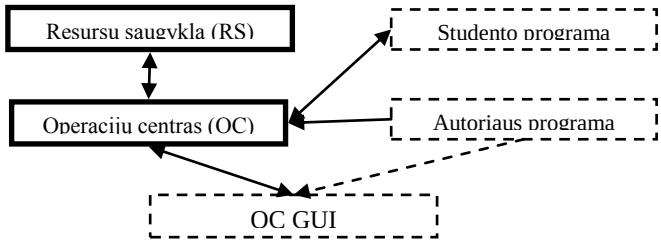
- - - - - - - - - - ► - bendravimo ryšis ne tiesiogiai. (pvz. autoriaus programoje sukurti klausimo variantai failo pavidalu, rankiniu būdu pakraunami į testų administravimo posistemę).

RS – resursų saugykla.

OC – operacijų centras. OC sudaro serverio posistemė (OC server), ir nuotolinė grafinė sąsaja serverio posistemei valdyti (**OC GUI**). OC serveris taip pat turi priemones RS saugyklos valdymui.

Studento programa –programa testams spręsti (šioje ataskaitoje studentas visada operuoja per šią programą).

Autoriaus programa – specialus įrankis klausimo variantams formuoti.



8 pav. Bendradarbiaujančios sistemos

Pateiktoje schemoje OC saugo ir apdoroja mokymosi resursus resursų saugykloje. OC valdomas per grafinę sąsają (schemoje parodyta kaip „OC GUI“). Studento programa užklausia prieinamus egzaminus, bei jų pateikimo būdus iš operacijų centro. Ateityje planuojama, kad instruktorius (dėstytojas, administratorius) per autoriaus programa kurs klausimą su variantais, kuri galės tiesiogiai per „Autorių“ nusiųsti į savo arba viešą resursų saugyklą. Šiuo metu jis tai gali padaryti saugant į failą ir rankiniu būdu per Administravimo programą (OC GUI) sudėti į pasirinktą resursų saugyklą.

3.1.4. Specializuoti programų paketai

Posistemėms kurti buvo naudojami laisvai platinami įrankiai ir paketai.

RS ir OC kūrimo procese ir galutinėje realizacijoje naudojamos šie programiniai produktai:
Java SDK 1.6, Hibernate, log4j, Apache ANT, jUnit, mysql connector, MySQL duomenų bazė.

Darbui organizuoti naudojama versijų kontrolės sistema Mercurial, bei projektų valdymo sistema Redmine.

3.2. KITOS SVARBIOS PRIELAIDOS

3.2.1. Licencija

Realizuojamos posistemės bus integruotis į sistemą, kuriuos licencija nėra tiksliai apibrėžta. Realizuojamas posistemes atskirai ar sudėtyje su kitais produktais naudoti komerciniams tikslams be šių komponentų autorių sutikimo draudžiama. Toks sutikimas gali būti suteiktas tik raštišku patvirtinimu tam tikram laikotarpiui su tiksliomis sąlygomis ir išlygomis, kurie išdėstyti raštiškai bei patvirtinti abiejų šalių. Išsamiau licencija yra pateikta 1 priede.

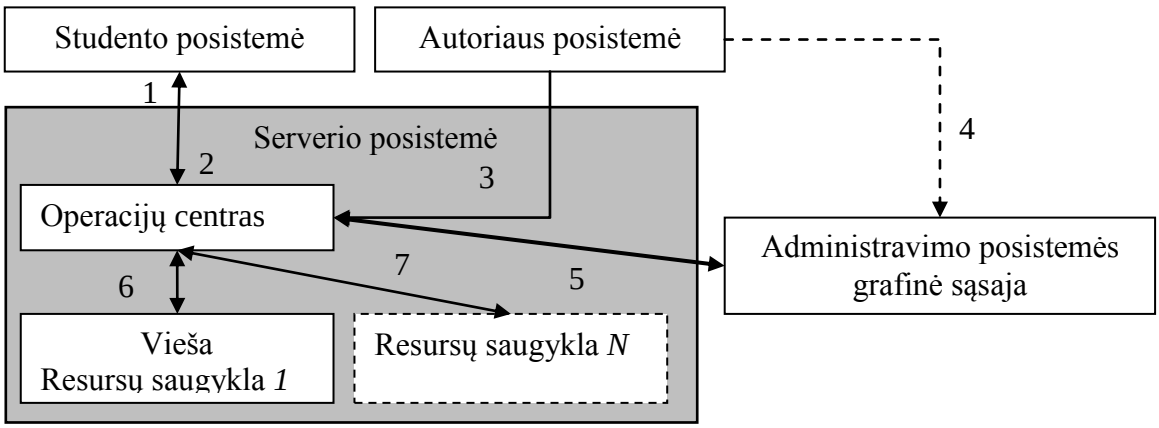
3.2.2. Komercinis panaudojimas

Sukurti paskirstyti moduliai bus integruoti į TestTool žinių testavimo sistemą. Tokiu būdu produktas bus platinamas kartu su šią sistemą. Potencialūs sistemos pirkėjai arba naudotojai galėtų būti organizacijos, kurie nori patikrinti savo darbuotojų žinias.

3.3. FUNKCINIAI REIKALAVIMAI

Funkciniai reikalavimai kuriamiems moduliams yra sudaryti pagal Volere šabloną, pateikti 6 priede.

3.3.1. Veiklos kontekstas (arba veiklos sudėtis)



9 pav. Konteksto tarp posistemių diagrama

Aukščiau pateiktoje schemoje ryšių tarp modulių paaiškinimas:

1 - Siunčiami išspręsti variantai, vartotojo informacija.

2 - Gaunami užduočių variantai, testai, egzaminai, kursai, įvertinimai.

3 - Siunčiami klausimai ir variantai.

4 - Kuria klausimus ir variantus.

5 - Organizuojamas ir kontroliuojamas mokymo procesą.

6, 7 - Administruojami klausimų resursai, saugomi/gaunami/vykdomi vertinimai.

Čia „Studento posistemė“ – studento programa, kurioje studentai sprendžia testus. „Autoriaus posistemė“ (toliau Autorius) – panaudodamas šią programą instruktorius kuria klausimus su variantais, klausimai siunčiami į viešą resursų saugyklą per operacijų centrą, arba saugomi atskirai ir per administravimo posistemę užkraunami į dėstytojo RS (čia reikėtų pabrėžti, kad visi resursai į RS siunčiami per Operacijų centrą) . Resursų saugykla tai sukurtos programinės įrangos dalis, nepriklausomas serverio komponentas atliekantis testavimo sistemoje testavimo ir vertinimo (įvertinimų) medžiagos saugojimą ir apdorojimą. Šių modulių kiekis gali būti nuo 1 iki N (N – natūrinis sveikas skaičius, kurio reikšmės prasideda nuo 1), kurie yra paleisti įvairiuose serveriuose. Vienu metu serveryje gali būti paleista tik viena RS serverio esybė.

RS resursai skirstomi į bendrus ir privačius, bendri resursai yra visi, kurie neturi rakto (RS posistemėje resurso duomenų laukas key) atributo. Atitinkamą raktą turi operacijų centro sistemos naudotojas, kuris įkrovė šį resursą į resursų saugyklą. Raktas yra unikalus ir nesikartoja. Klientinė programa, kuri jungiasi į resursų saugyklą turėtų užtikrinti, kad resursai, kurie turi netuščią atributą su raktu nebūtų dalinami naudotojams, kurių raktas skiriasi nuo rakto priskirto šiam resursui.

Bendri resursai yra tie, kurių unikalus raktas (resurso duomenų laukas key) yra tuščias. Prie šių resursų gali prieiti visi resursų saugyklos naudotojai.

Kiekvienam pridėtam OC administratoriui yra priskiriamas unikalus raktas – UUID. Šis raktas yra visiškai unikalus, nepriklauso nuo operacinės sistemos bei OC serverio esybės; yra priskiriamas prie resurso, kuris kraunamas į resursų saugyklą, jei norimą padaryti šį resursą privačiu. Naudotojų slaptažodžiai yra šifruojamas md5.

OC turi 3 tipų vartotojus - administratoriai, super-administratoriai dirba per administratorių RMI sąsają, studentai dirba per Studento RMI sąsają, panaudodami tam specialiai sukurta Studento programą, tiesiogiai prie posistemės jungtis negali. Administratoriai naudojasi visomis OC posistemės funkcijomis savo kurso viduje. Super-administratoriai yra globalūs administratoriai, kurie prieina prie visų OC posistemės funkcijų, įskaitant administratorių valdymą.

RS gali būti viešos arba priskirtos konkrečiam dėstytojui/instruktoriui. OC valdomas per „Administravimo posistemės grafinę sąsają“ (tekste minimas kaip OC GUI). RS posistemėje teisės nėra specialiai skirstomos. Posistemė neturi aiškiai apibrėžtų naudotojų. Bet kokios sistemos, kurios jungiasi į posistemę panaudojant RMI prisijungimo sąsają bei naudojasi jos posistemės paslaugomis galima skaityti šios posistemės naudotojais.

Su magistrinio projekto realizacija susijusią dokumentaciją galima surasti projekto informacinėje sistemoje[23]. Visas TestTool programas galima rasti oficialiame TestTool puslapyje[24].

3.3.2. Veiklos padalinimas

2 lentelėje pateikiamas veiklos įvykių sąrašas OC posistemei. Kadangi vartotojai gali jungtis į RS posistemes tik per OC posistemę, veiklos padalijimas atskirai neaptariamas, bet prie įeinančių srautų pažymima, kad šis informacijos srautas pasiekia RS posistemę (rs). Nors ir tie informacijos srautai praeina pro OC posistemę.

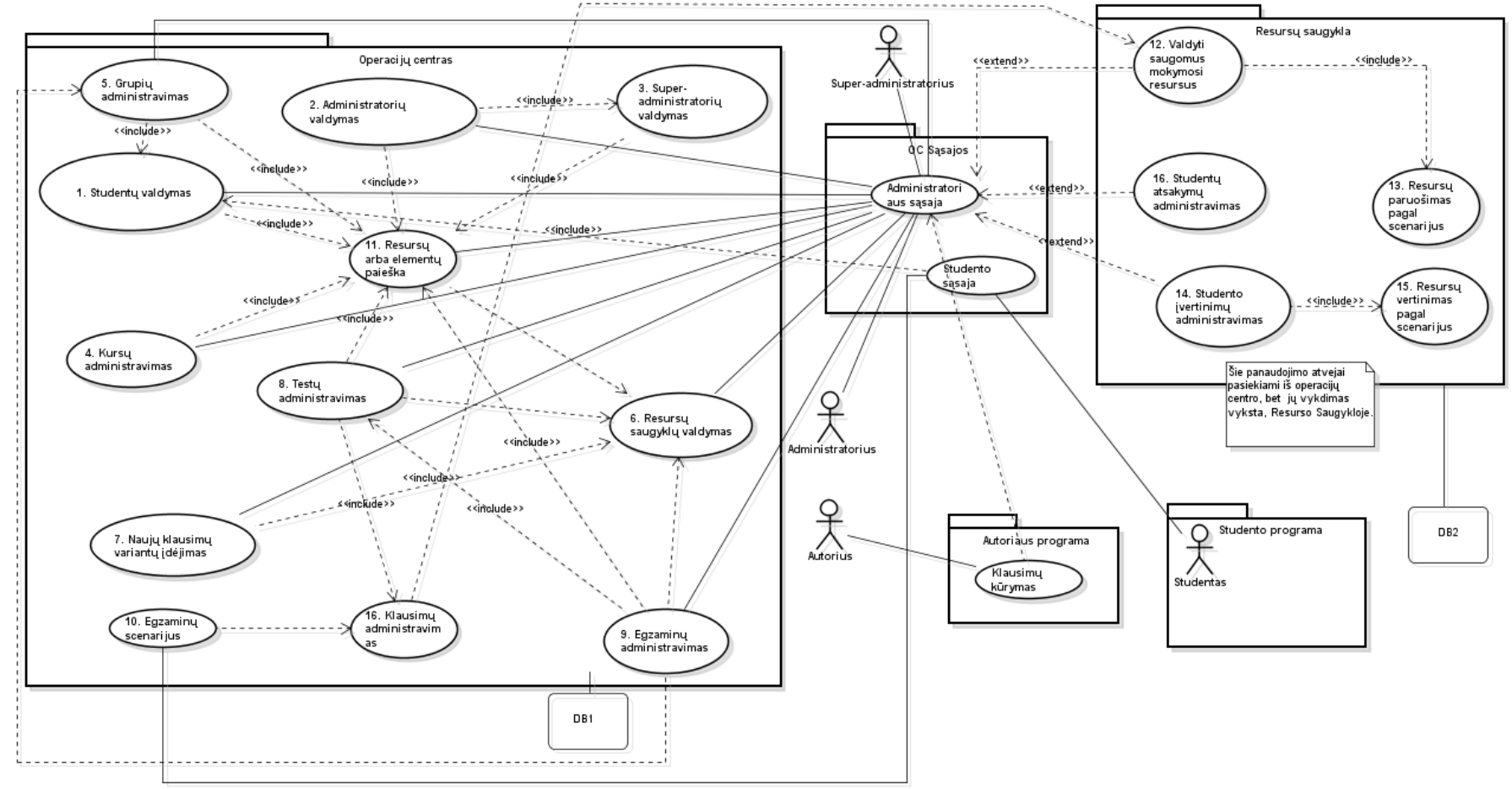
2 lentelė. Veiklos įvykių sąrašas OC posistemėje

Eil. Nr.	Įvykio pavadinimas	Įeinantys/Išeinantys informacijos srautai
1.	Super-administratorius sukuria administratorių ir kursą. Suriša sukurta administratorių su naujai sukurtu arba esamu kursu ir jei reikia priskiria jam nuosavą RS.	Vartotojų informacija (in) Kursų informacija (in) Egzistuojančių RS sąrašas (in)
2.	Administratorius kuria kursus (tik naujų sukūrimas arba jam priskirtų pakeitimas), studentus, grupes, testus, egzaminus, nustato egzamino organizavimo tvarką, vertinimą.	vartotojai (in/out) kursai (in/out) grupės (in/out) testai (in/out) egzaminai (in/out) klausimai (in/out) (rs) variantai (in/out) (rs) vertinimo rezultatai (out) (rs) vertinimo organizacija (in), (rs)
3.	Studento programa gauna prieinamų kursų sąrašą studentui, bei skirtus jam testus, egzaminus ir jų klausimus. Studento programa pasiekus tam tikras aplinkybes inicijuoja vertima pagal pateiktą tvarką, bei užklausia vertinimo rezultatus.	Kursai (out) Grupės (out) Testai (out) Egzaminai (out) Klausimo variantai (out) (rs) Atsakyti variantai (modifikuoti klausimo failai) (in) (rs) Vertinimo rezultatai (out) (rs)

3.3.3. Sistemos ribos

Sistemos ribas parodo panaudojimo atvejų diagrama, kuri pateikta 10 pav. Sistemos ribos

Atskirai panaudojimo atvejų nespecifikuosime, apibendrintai sukurtų modulių paskirtis yra apibrėžta sistemos aprašymo skyriuje.



10 pav. Sistemos ribos

3.3.4. Paskirstytų modulių aktoriai

OC modulyje administratoriai, super-administratoriai dirba per administratorių RMI sąsają.

Administratoriai naudojasi visomis OC posistemės funkcijomis savo kurso viduje. Super-administratoriai yra globalūs administratoriai, kurie prieina prie visų OC posistemės funkcijų, įskaitant administratorių valdymą.

Studentai dirba per Studento RMI sąsają, panaudodami tam specialiai sukurta Studento programą, tiesiogiai prie posistemės jungtis negali.

RS posistemei aktorių rolių neturi, tiesiogiai jungtis negali, prisijungimo teisę ir priėjimas suteikiamas išskirtinai per OC modulį.

3.4. NEFUNKCINIAI REIKALAVIMAI

3.4.1. Reikalavimai sistemos išvaizdai

Operacijų centro ir resursų saugyklų serverio posistemės grafinės sąsajos neturės.

3.4.2. Reikalavimai panaudojamumui

Administratoriai sistemoje yra kaip dėstytojai arba instruktoriai, jiems nereikia specialių žinių, bet pageidautina žinoti apie pagrindines sistemos taisykles ir principus, tokius kaip, kad klausimas susideda iš variantų, testas iš klausimų, egzaminas iš testų, klausimų variantai egzamine dalinami pagal egzamino scenarijų ir t. t. Su šiais principais vartotojas galės susipažinti vartotojo dokumentacijoje.

Super-administratorius panašiai kaip ir administratorius turi žinoti tam tikrus sistemos naudojimo ypatumus, kurie bus išdėstyti vartotojo vadove.

Serverio posistemių administratoriai turi būti susižinę su programų sandarą ir dalimis, palaikančios duomenų bazės, aplinkos kintamųjų nustatymai, reikiamų bibliotekų sėkmingam programų veikimui pridėjimas serveryje bei mokėjimas orientuotis programų log failuose.

3.4.3. Reikalavimai vykdymo charakteristikoms

Šiai sistemai netaikomi ypatingi greitaveikos reikalavimai.

3.4.4. Reikalavimai veikimo sąlygoms

Darbui su sistema reikalingi standartiniai kompiuterio valdymo prietaisai.

Darbui su klientinėmis posistemis – klaviatūra, pelė, vaizduoklis. Serverio posistemėms klaviatūra reikalinga tik paleidimo ir konfigūravimo metu.

3.4.5. Reikalavimai sistemos priežiūrai

Galimybės, kurios leidžia suprasti sistemos veikimą, serverio posistemės turi būti vedamas veiksmų log failas.

Kai administratorius prisijungia per grafinę sąsają (kuria kiti programuotojai) turi būti realizuotos galimybės ir atitinkamos funkcijos serverio pusėje einamų įvykių stebėjimui.

3.4.6. Reikalavimai saugumui

Administratorius naudoja slaptažodį prisijungimui prie administravimo sistemos. Slaptažodžiai duomenų bazėje nesaugomi atviru tekstu. Kitų specialių saugumo reikalavimų kuriamai sistemai nėra.

3.4.7. Teisiniai reikalavimai

Sistema kuriama laikantis LR įstatymų ir organizacijoje nustatytų vidinių taisyklių.

3.5. PROJEKTO IŠEIGA

3.5.1. Atviri klausimai (problemos)

Šie reikalavimai galutinai ir tiksliai neapibrėžia kokiu būdu bus organizuotas sukurtų per Autoriaus programa klausimo variantų perdavimas, priskyrimas ir atskiriamas priėjimas (teisės) administravimo posistemėse.

Reikalavimai taip pat tiksliai neapibrėžia kaip turėtų būti realizuojamas klausimo pateikimo scenarijus (kuris atsakys už klausimo variantų pateikimą). Reikalavimai tiksliai nenurodo koks tikslūs duomenų formatas turi būti naudojamas administravimo serverio posistemėje.

3.5.2. Galimos problemos diegimo aplinkai

Posistemių įdiegimo metu diegimo aplinkai neturėtų kilti jokiu problemų.

3.5.3. Įtaka jau instaliuotoms sistemoms

Įdiegus posistemes jie naudosis suteiktais aparatiniais ir programiniais resursais kartu su kitomis sistemomis. Normalus posistemių veikimas neturėtų įtakoti kitas suinstaliuotas sistemas, tiesiogiai nesusijusius su jais. Nekorektiškas sistemos naudojimas gali įtakoti kitas instaliuotas sistemas arba gretutines posistemes.

3.5.4. Neigiamas vartotojų nusiteikimas

Kai kurioms grupėms vartotojų gali būti nepriimtina grafinės sąsajos nebuvimas, vidinių komponentų struktūra arba modulių kalba. Sistemos aptarnavimo metu reikėtų atkreipti

dėmesį į šiuos vartotojus ir prieiti kompromisam taip pat svarbu pateikti aiškią dokumentaciją, kad vartotojams būtų paprasčiau adaptuotis pasikeitusioje sistemoje.

3.5.5. Kliudantys diegimo aplinkos apribojimai

Diegimo aplinkoje turi būti laisvi prievadai reikalingi diegiamai posistemei. Priešingu atveju diegiamas posistemes bus neįmanoma sėkmingai naudoti.

3.5.6. Galimos naujos sistemos sukeltos problemos

Jei sistema naudosis nekvalifikuotas personalas arba galimas tyčinis sistemos resursų išnaudojimas dideliais kiekiais, tai gali iškilt resursų nepakankamumas sistemoje, ko pasekmėje gali nekorektiškai veikti kitos nepriklausomos sistemos, kurios dalija resursus su kuriama sistema.

3.5.7. Reikalavimai esamų duomenų perkėlimui

Esami testų klausimai ir variantai sukurti per Autoriaus sistema (Autoriaus posistemės versija 5.2) yra suderinami su kuriama administratoriaus posistemę, todėl nereikalauja papildomų manipuliacijų, tam, kad paruošti juos naudojimui naujoje sistemoje. Jau sukurti egzaminai, kursai, testai yra nesuderinami su kuriama sistema. Vartotojų informacija gali būti perkelta į naujai kuriamą sistemą.

3.5.8. Reikalingas duomenų transformavimas perkeliant į naują sistemą

Keliant vartotojų informacija iš senesnių posistemių versijų galimas tik dalinis informacijos laukų perkėlimas, kadangi ne visi vartotojų laukai sutampa.

3.5.9. Galimos paskirstytų modulių kūrimo rizikos

3 lentelėje pateiktas tikimybinis įvertinimas parodo tikimybę rizikos atsiradimui, kur 1 – neišvengiamas įvykis, 0 – įvykis neįmanomas.

3 lentelė. Galimos sistemos kūrimo rizikos

Rizikos faktorius	Tikimybinis įvertinimas
Reikalavimų specifikacijos pasikeitimai realizavimo fazėje	0,9
Personalo ligos	0,5
Programinės architektūros nesuderinamumas su reikalavimų specifikacijoje išvardintais reikalavimais arba sprendimais	0,6
Programinių modulių arba komponentų nesuderinamumas su sistema.	0,4

Neaiški arba neegzistuojanti dokumentacija modifikuojamiems posistemėms.	0,9
Aparatinių priemonių gedimai ir saugomos informacijos praradimas.	0,4

3.5.10. Atsitiktinumų (rizikų) planas

Rizikos faktoriai ir numatomi planai problemoms spręsti

4 lentelė. Rizikų planas

Rizikos faktorius	Problemos sprendimas
Reikalavimų specifikacijos pasikeitimai realizavimo fazėje	Prioritetinių uždavinių realizacija, mažesnio prioriteto užduočių atmetimas.
Personalo ligos	Laiko rezervo išskyrimas nenumatytiems atvejams.
Programinės architektūros nesuderinamumas su reikalavimų specifikacijoje išvardintais reikalavimais arba sprendimais.	Jei neįmanoma greitai pakeisti programinio produkto architektūros (ne daugiau 15% pakeitimų), keisti arba atsisakyti specifikacijoje nurodytų reikalavimų. Sprendimus derinti su produkto kūrėjais ir užsakovu.
Programinių modulių arba komponentų nesuderinamumas su sistema.	Naudoti tik tuos modulius arba komponentus, kurių suderinamumą su kuriamos sistemos aplinką patvirtina duotų modulių gamintojai.
Neaiški arba neegzistuojanti dokumentacija modifikuojamiems posistemėms.	Iš anksto skirti daugiau laiko toms dalims, kur bus modifikuojamos egzistuojančios sistemos, jei šis programuotojas nepatyręs arba nėra modifikuojamos sistemos autorius..
Aparatinių priemonių gedimai ir saugomos informacijos praradimas.	Laiku daromos atsarginės kopijos.

3.6. KAINA

Projekto kainą neįmanoma tiksliai apskaičiuoti. Projekto vertę galima įvertinti laiku, kuris yra sugaištas projekto realizacijai.

3.7. VARTOTOJO DOKUMENTACIJA IR APMOKYMAS

Serverio posistemės turi trumpas paleidimo ir konfigūravimo instrukcijas. Posistemės, su kuriom bendrauja galutinis vartotojas turi būti pateikta kartu su vartotojo vadovu.

Planuojama parengti dokumentaciją šioms posistemėms: OC serverio administravimo posistemei, komponentams, kurie pakeisti arba naujai įdiegti į egzistuojančias posistemes.

3.8. PERSPEKTYVINIAI REIKALAVIMAI

- Galimybė naudoti keletą resursų saugyklų su vienu operacijų centru.

3.9. IDĖJOS IR SPRENDIMAI

- Vertinimo ir egzaminų resursų atskirimas ir pakartotino panaudojamumo užtikrinimas.
- Klausimų variantų pateikimo organizavimo scenarijai.
- Paruošimo scenarijai.
- Vertinimo scenarijai.
- Svoriai paremti vertinimo scenarijais.
- Architektūrinių programavimo šablonų panaudojimas, ko pasekmėje padidėja sistemos lankstumas ir pakartotinio panaudojimo galimybė.
- Skirtingų duomenų bazių palaikymas.

3.10. PASKIRSTYTŲ ADMINISTRAVIMO POSISTEMIŲ ARCHITEKTŪRA

3.10.1. Architektūros tikslai ir apribojimai

Sistemos architektūrą įtakoja keletas reikalavimų ir apribojimų, kurie buvo išvardinti projekto specifikacijos:

Apribojimai: Kadangi sukurtos posistemės integruojamos į esamą sistemą, tai jos turi būti realizuojama Java kalba - suderinamumui užtikrinti.

Tikslai:

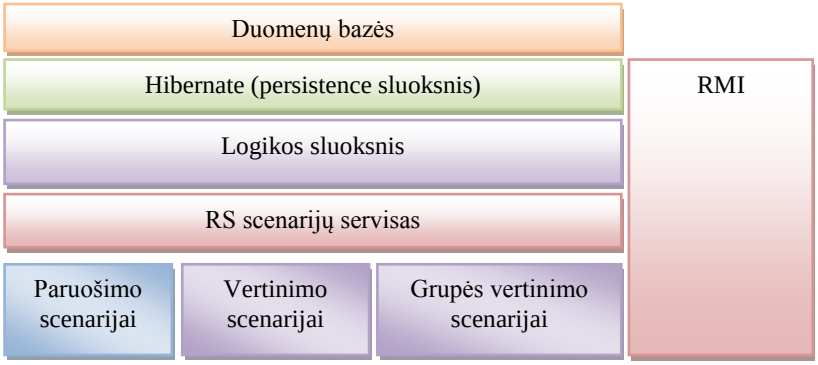
- Sukurti tokią sistemos architektūrą, kurie leistų lengvai plėsti sistemą.
- Sudaryti sąlygas ir/arba galimybes naudoti sistemą įvairiems žmonių grupėms (neapsiriboti studentais), lanksčiai reguliuoti sistemos vidinius procesus pagal pageidavimą (pvz. vertinimą, testų organizavimą).
- Užtikrinti sistemos būsenos stebėjimą, naudojant vykdomų veiksmų sekimą, rašant įvykius į log failą.
- Užtikrinti sistemos stabilų veikimą.

Sistemos architektūra pateikiama, panaudojant šias diagramas: panaudojimo atvejų vaizdas - panaudojimo atvejų diagrama, sistemos statinis vaizdas - sistemos modulių paketai ir apibendrinta komponentų diagramos, duomenų bazės diagramos.

Sukurtą sistemą sudaro šie moduliai: Operacijų centras, Resursų saugyklą, papildomi paketai kitų programų sudėtyje, kurie užtikrina šių modulių veikimą visoje sistemoje.

3.10.2. Modulių apžvalga

Resursų saugykla (RS) – užtikrina mokymosi resursų saugojimą ir grąžinimą pagal reikalavimą, mokymosi resursų vertinimą ir paruošimą panaudojant scenarijus.



11 pav. Resursų saugyklos architektūros programiniai sluoksniai

Hibernate sluoksnis užtikriną bendravimą tarp aplikacijos logikos ir duomenų bazės, realizuojamas panaudojant Hibernate biblioteką ir aplikacijos logiką.

Logikos sluoksnis realizuojamas grupuojant panaudojant DAO (angl. Data Access Objects) programinį šabloną. RS DAO objektų struktūra pavaizduota 3 pav.

RMI sluoksnis užtikrina bendravimą tarp modulių.

Scenarijų servisas užtikrina scenarijų egzistavimą ir jų panaudojimą aplikacijoje.

Paruošimo scenarijai (11 pav.) (tik RS posistemėje) užtikriną resursų paruošimą pagal tvarką, kurią aprašo scenarijus, scenarijaus rezultatas yra paruoštas scenarijus.

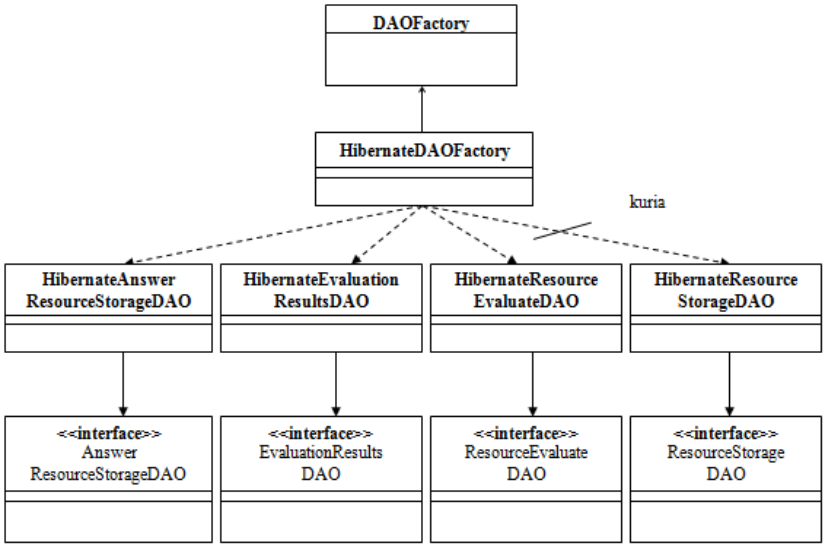
Vertinimo scenarijai (11 pav.) (tik RS posistemėje) užtikrina mokymosi resurso (bendru atvejų tai studentų atsakymas) įvertinimą.

Grupės vertinimo scenarijai (11 pav.) (tik RS posistemėje) – bendru atveju naudojami grupės studentų įvertinimui, bet priklausomai nuo konkretaus scenarijaus realizacijos gali būti panaudoti ir kitiems tikslams, pvz. elektroninio pašto siuntimui nurodytai asmenų grupei, panaudojant tam tikra scenarijuje aprašytą šabloną.

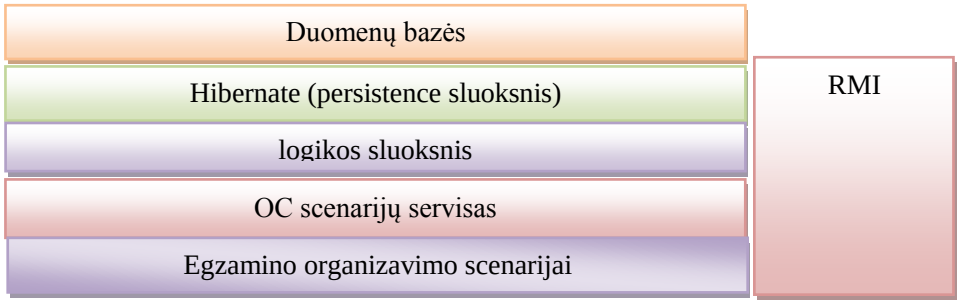
Klausimo variantų scenarijai (13 pav.) (tik OC posistemėje) – priklausomai nuo konkretaus scenarijaus realizacijos užduoda klausimų variantų pateikimą studento programai.

Operacijų centras (OC) – užtikrina žinių testavimo proceso organizavimą, valdo vartotojus, testavimo dalyvius, testus ir egzaminus, valdo resursų saugyklą (-as).

12 pav. pavaizduota RS posistemėje naudojama DAO objektų diagrama, panaudojant abstraktų fabriką.

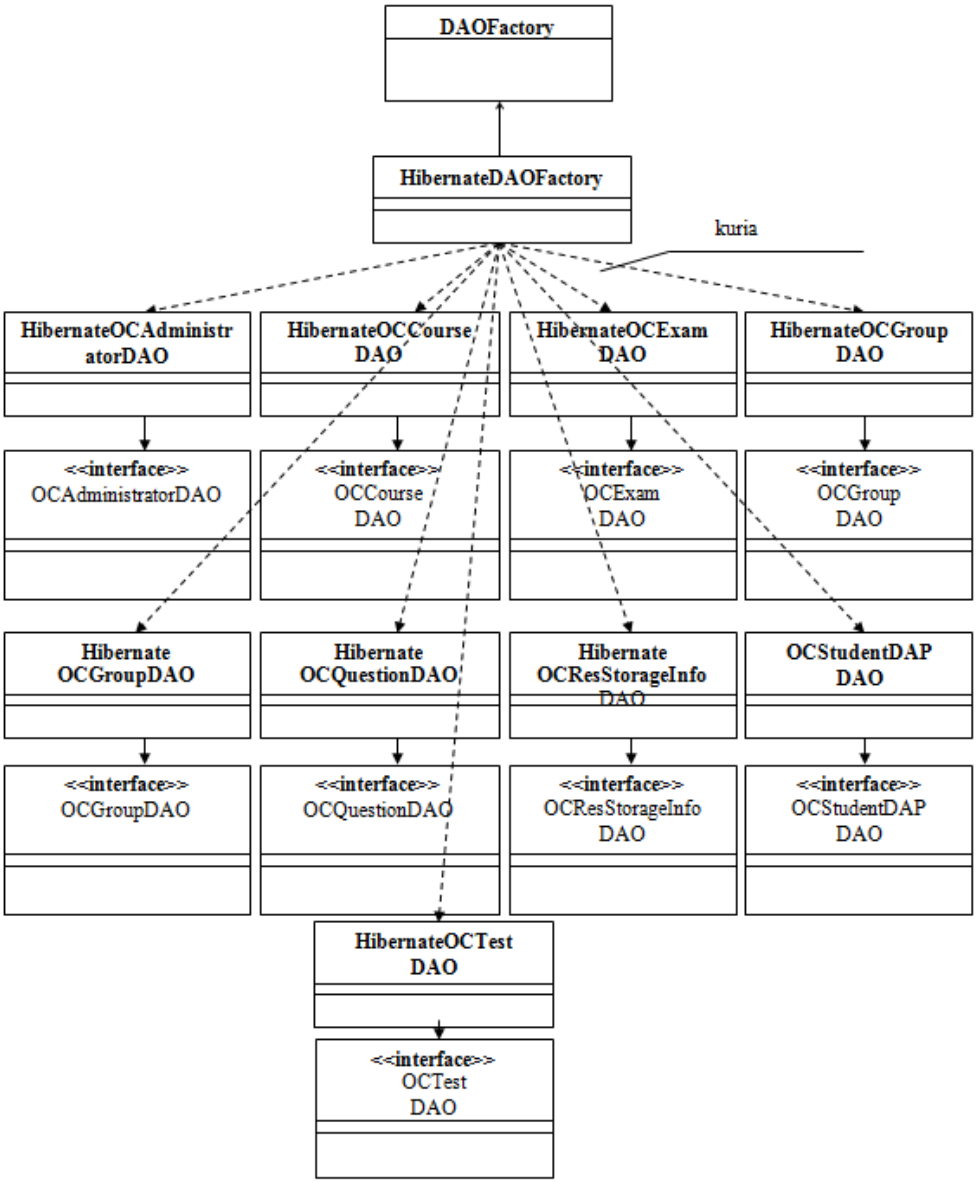


12 pav. RS DAO objektų schema, panaudojant abstraktų fabriką



13 pav. Operacijų centro architektūros programiniai sluoksniai

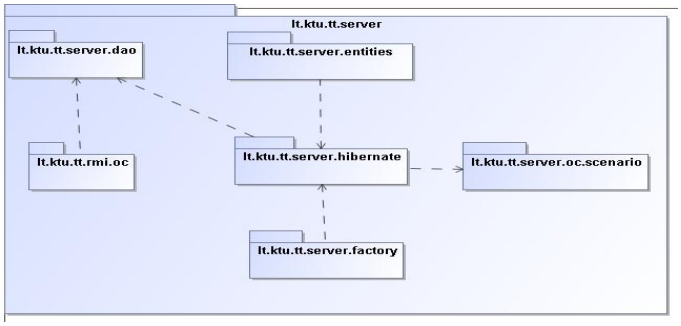
14 pav. pavaizduota OC posistemėje naudojama DAO objektų diagrama, panaudojant abstraktų fabriką.



14 pav. Pagrindinių OC DAO objektų schema, panaudojant abstraktų fabriką

RS ir OC architektūra realizuojama pagal vienodus principus ir struktūrą, bet atliks skirtingas funkcijas – abi aplikacijos turi sluoksnių architektūrą. Ši koncepcija parodyta aukščiau 11 ir 13 pav. Tokia architektūra turėtų užtikrinti lengvesnį sistemos plečiamumą..

Pagrindiniai OC paketai:



15 pav. Pagrindiniai OC modulio paketai

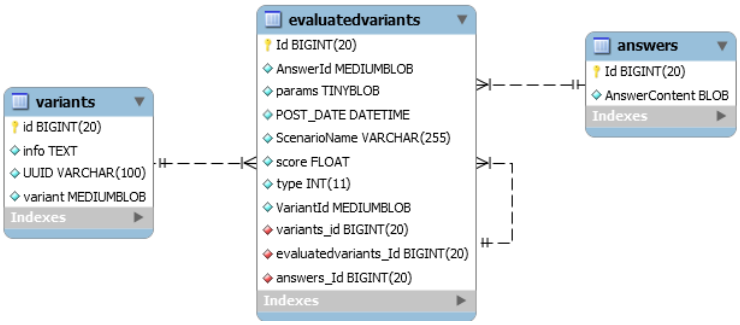
RS posistemėje naudojama panaši paketų struktūra, todėl atskirai jos nespecifikuosime.

Modulių klasių sandaros, funkcijų procesus bei sąveikas šiame dokumente nedetalizuosime, nes tai buvo pakankamai išsamiai detalizuota magistrinio projekto dokumentacijoje.

2 pav. jau buvo parodyta apibendrinta TestTool sistemos struktūra. Pilkai pažymėtų modulių architektūra šioje ataskaitoje nespecifikuojama, nes jų kūrimu užsiima kiti programuotojai.

3.10.3. Resursų saugyklos duomenų vaizdas

Žemiau yra pateikta resursų saugyklos duomenų bazės esybių diagrama.



16 pav. RS esybių diagrama

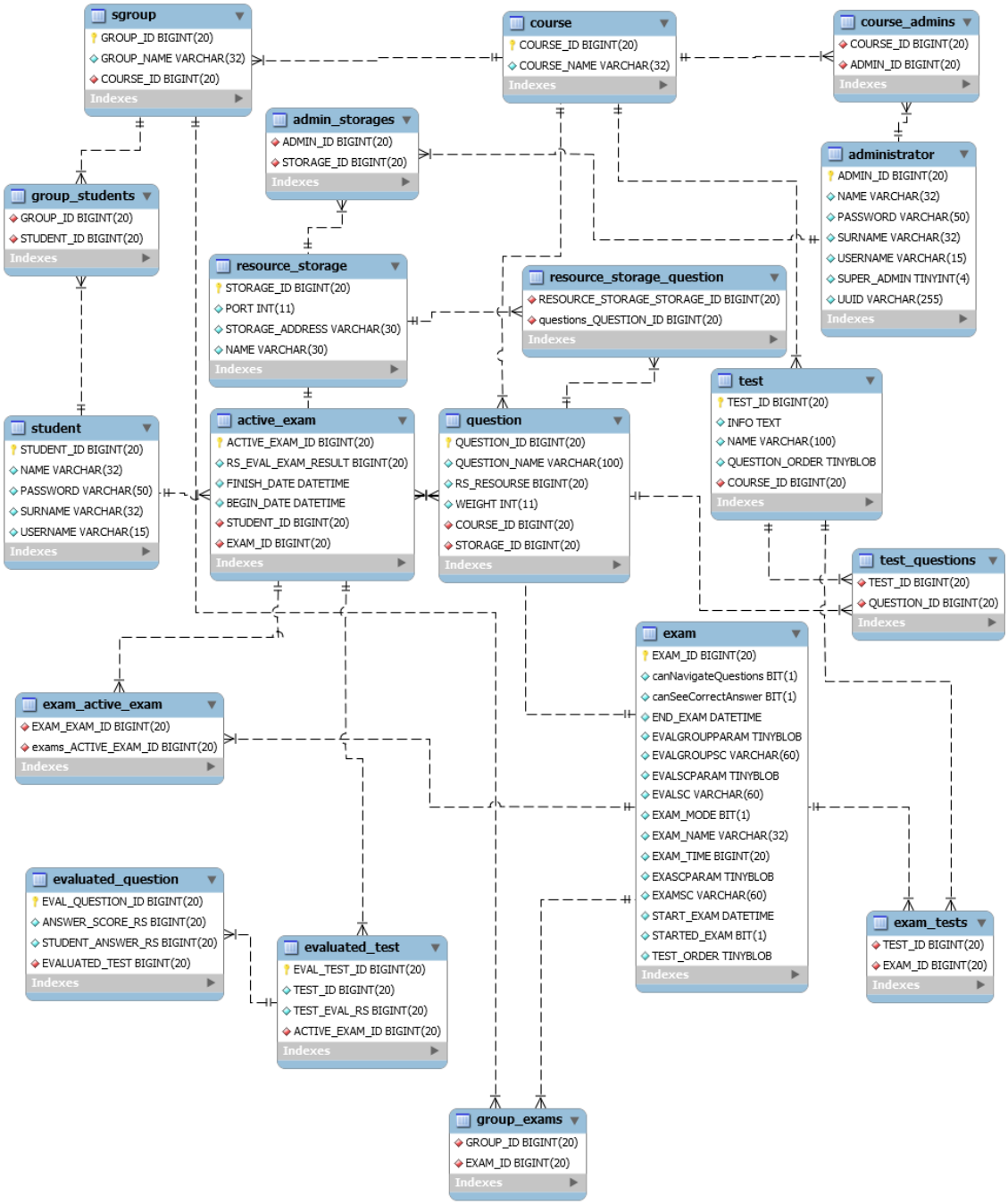
Aukščiau pateiktos diagramos duomenų bazės lentelių paskirtis yra paaiškinta 5 lentelėje žemiau.

5 lentelė. Resursų saugyklos duomenų bazės lentelių paskirtis

Lentelės pavadinimas	Paskirtis
variants	Mokymosi resursų saugojimui.
evaluatedvariants	Įvertintų variantų saugojimui.
answers	Studento įvertinimams saugojimui.

3.10.4. Operacijų centro duomenų vaizdas

17 pav. pateikta operacijų centro duomenų bazės esybių diagrama.



17 pav. OC esybių diagrama

Aukščiau pateiktų duomenų lentelių paskirtis yra paaiškinta 6 lentelėje.

6 lentelė. Operacijų centro duomenų bazės lentelių paskirtis

Lentelės pavadinimas	Paskirtis
sgroup	Saugo grupių informaciją.
course	Saugo kursų informaciją.
course_admins	Pagalbinė lentelė kursų ir jų administratorių surišimui.
administrator	Saugo administratorių informaciją.
admin_storages	Pagalbinė lentelė administratorių ir jiems priskirtų resursų

	saugykloms surišti.
group_students	Pagalbinė lentelė grupių ir jose esantiems studentams saugoti.
student	Saugoma informacija apie studentus.
resource_storage	Saugoma informacija apie resursų saugyklas
resource_storage_question	Pagalbinė lentelė, kurios paskirtis yra surišti konkrečius klausimus su atitinkamomis resursų saugyklomis.
active_exam	Aktyvių (vykdomų arba pradėtų vykdyti) informacijos saugojimas.
question	Sukurtų klausimų informacijos saugojimas.
test	Sukurtų testų informacijos saugojimas.
test_question	Pagalbinė lentelė testo klausimams saugoti.
exam	Egzamino informacijai saugoti.
exam_active_exam	Pagalbinė lentelė suristi aktyvų egzaminą su jam atitinkančiu realiu.
evaluated_question	Įvertintų klausimų informacija.
evaluated_test	Įvertintų testų informacija.
group_exams	Pagalbinė lentelė nurodanti kuriai grupei yra priskirtas realus egzaminas.
exam_tests	Pagalbinė lentelė, nurodanti, kokie testai yra priskirti kokiam egzaminui.

3.11. SKYRIAUS IŠVADOS

Sistema gali veikti skirtingoje platformose, kadangi ji realizuota Java kalba.

Sistemos naujai realizuotos posistemės turi pakankamai lanksčią pakeitimams architektūrą, tokiu būdu ji turėtu palengvinti busimus sistemos patobulinimus – naujų funkcijų realizaciją.

Veiksmų scenarijai suteikia lankstumo žinių testavimo procesui, kai galima realizuoti skirtingus vertinimo ir klausimų variantų pateikimo scenarijus, bei juos pritaikyti konkrečiam atvejui. Paruošimo scenarijai leis apjungti įvairių variantų arba kitų resursų paruošimą vienoje sistemoje, o nenaudoti papildomų įrankių, kurie dažnai būna prastos kokybės.

Sistemos posistemės gali veikti kaip viename kompiuteryje, taip ir nutolę vienas nuo kito. Posistemės gali naudoti ir skirtingas duomenų bazes, ir tą pačią. Kiekvienai aplikacijos esybei galima naudoti po vieną duomenų bazę. Kadangi naudojamas Hibernate sluoksnis, galime naudoti skirtingas duomenų bazes (pagal gamintoją): DB2, DB2 AS/400, DB2 OS390, PostgreSQL, MySQL, Oracle, Sybase, Microsoft SQL, SAP DB, Informix, HypersonicSQL,

Ingres, Progress, Mckoi SQL, Interbase, Pointbase, FrontBase, Firebird. Operacijų centras gali turėti vieną visiems jo kursams bendrą resursų saugyklą arba kiekvienam kursui atskirai, priklausomai nuo administravimo pageidavimų.

4. SUKURTŲ PASKIRSYTŲ MODULIŲ TESTAVIMAS

Testavimas yra programinės įrangos (PĮ) kokybės užtikrinimo proceso dalis. Pirminis testavimo tikslas yra identifikuoti PĮ klaidas taip, kad juos galima būti atrasti ir ištaisyti. Testavimas negali garantuoti, kad produktas funkcionuos visomis sąlygomis, bet tik gali įrodyti, kad tam tikromis sąlygomis jis veikia netinkamai [25].

4.1. TESTAVIMO TIKSLAI IR OBJEKTAI

Testavimas padeda užtikrinti minimalų klaidų ir netikslumų skaičių realizuojamame produkte. Testavimas leidžia užtikrinti atitikimą reikalavimams, kurie yra nurodyti reikalavimų specifikacijoje.

Testavimo planas nustato programinės įrangos komponentus, kurie bus testuojami, numato testavimo strategijas, resursus ir apribojimus, rezultatus, taip pat naudojamus įrankius ir aplinką.

Testavimo planas yra pateiktas 3 priede.

4.2. TESTAVIMO APIMTIS

Testavimo tipai, kurie bus panaudoti vykdant sukurtų OC ir RS modulių testavimą:

- Vienetų testavimas: Operacijų centro funkcijų testavimas, Resursų saugyklos funkcijų testavimas.
- Integracijos testavimas: Resursų saugyklos ir operacijų centro abipusio darbo testavimas, Studento programos ir operacijų centro, bei resursų saugyklos sąveika.
- Validavimo testavimas: klaidų kontrolė.
- Priėmimo testavimas: Operacijų centro operacijų vykdymas, studento egzaminų organizavimas ir kontrolę studento programoje per operacijų centrą, vartotojų administravimas, egzaminų, testų, klausimų kūrimas.
- Saugumo testavimas: vartotojų prisijungimo testavimas, vartotojų slaptažodžių saugumo analizė.
- Našumo testavimas: resursų saugyklos veikimo metu išnaudojamų resursų kiekis (operatyvioji atmintis, procesoriaus resursų išnaudojimas ir t. t.), Operacijų centro veikimo metu išnaudojamų resursų kiekis, įvairių Resursų saugyklos ir Operacijų centro operacijų vykdymo greitaveika.

Programinės įrangos savybių testavimo prioritetai:

- Funkcionalumas – atitikimas reikalavimų specifikacijoje išvardintomis funkcijoms.
- Panaudojamumas – ar realizuotos programinės įrangos API yra patogus programuotojui.
- Paprastumas – ar realizuota programinė įranga yra paprasta naudojime.
- Praplečiamumas (lankstumas) – ar realizuotą programinę įrangą galima lengvai praplėsti naujomis galimybėmis (ar pateiktoje programinėje įrangoje naudojami tam skirti programavimo šablonai, strategijos ir t. t.), ar sistemą galimą naudoti skirtingomis sąlygomis (pvz. su įvairiomis duomenų bazėmis).
- Vykdymo laikas, greitaveika – laikas, nėra kritinis parametras, bet jis turi būti visiems priimtinose ribose.
- Saugumas – nėra kritinis parametras, bet vartotojo slaptažodžiai turi būti šifruojami.

Pastaba: Kadangi sukurtų modulių apimtis yra didelė, o laikas skirtas testavimui yra ribotas, planuojama, kad bus įvykdytas tik dalinis testavimas Šiuo atveju testavimas bus užbaigtas tada, kai pasibaigs projekto plane numatytas laikas.

4.2.1. Pagrindiniai testuojamo objekto apribojimai

Organizaciniai, techniniai ir programiniai testavimo apribojimai. Apribojimai, dėl kurių gali nepavykti atlikti kai kuriuos testus, arba teisingai nustatyti testo rezultatą:

- Testuojama dalis priklauso nuo daugybės kitų komponentų (arba kitų funkcijų), kurie irgi gali turėti klaidų arba netikslumų.
- Testuojama dalis priklauso nuo posistemės (arba komponento), kurio realizacija nepriklauso šiam projektui.
- Operacijos vykdamos lygiagrečiai.
- Per trumpas laikas skirtas sistemos testavimui.

4.2.2. Testuojama programinė įranga

Testuojamos dalys: Resursų saugykla (RS), Operacijų centras (OC). Planuojama testuoti žemiau pateiktų komponentų funkcionalumą:

- Vartotojų administravimas (studentu, jų grupių ir administratorių).
- Klausimų, testų, egzaminų valdymas.
- Egzaminų organizavimo ir vertinimo parametrų pasirinkimas (scenarijai).

- Egzaminų vykdymas skirtingose režimuose (priklausomai nuo pasirinkto scenarijaus) studento programoje.

4.2.3. Vienetų testavimo strategija

Resursų saugyklos ir operacijų centro funkcijoms ištestuoti naudojamas vienetų testavimas. Bus naudojamas baltos dėžės testavimas, kur kiekviena modulio funkcija testuojama atsižvelgiant į jos struktūra. Testavimo metu bus paduodami tam tikri parametrai į įėjimus ir gautas rezultatas bus lyginamas su teisingu. Šis testavimas bus vykdomas panaudojant testavimo atvejus, kurie realizuojami jUnit pagalba.

4.2.4. Integravimo testavimo strategija

Panaudojant vienetų testavimą bus testuojami komponentų klasės ir moduliai. Po šių komponentų apjungimo bus naudojamas integravimo testavimas. Palaipsniui bus jungiamos komponentų dalys ir tikrinamas jų abipusis veikimas. Testuojant bus naudojamos atitinkamo veiksmo valdymo funkcijos.

4.2.5. Aukšto lygio testavimo strategija

Priėmimo testavimo metu, užsakovas išbando sukurtą programinę įrangą. Užsakovas nusprendžia kiek sukurta programinė įranga atitinka specifikaciją. Jei yra pastebėti nežymus neatitikimas su specifikacija, tokie netikslumai pašalinami, priešingu atveju gali prireikti pakeisti arba papildyti reikalavimų specifikaciją ir šiuos pakeitimus galima bus realizuoti tik kitoje programinės įrangos versijoje (revizijoje).

Saugumo testavimo metu, yra tikrinami ar vartotojo slaptažodis yra saugomas atvirtu tekstu ar koduotas. Taip pat ar įmanomas saugūs resursų saugojimas (atskirimas pagal vartotojus). Tikrinama ar reikalaujama vartotojų autorizacija, norint pasiekti programinės įrangos teikiamas paslaugas.

4.2.6. Testavimo resursai

Testavimui bus naudojama keletas kompiuterių, kuriuose bus įdiegtos atskiros „TestTool 5,3“ posistemės. Kompiuterio konfigūracija pateikta žemiau.

7 lentelė. Kompiuterio Nr. 1 konfigūracija

Procesorius:	Pentium 4, 3.40 Ghz (1 core)
Operatyvioji atmintis:	1 Gb
Vaizdo plokštė:	ATI Radeon X600, 128 Mb
Operacinė sistema	Windows XP Professional

8 lentelė. Kompiuterio Nr. 2 konfigūracija

Procesorius:	Core 2 Duo, 2.00 Ghz
Operatyvioji atmintis:	2 Gb
Vaizdo plokštė:	ATI Mobility Radeon X1600, 256 MB
Operacinė sistema	Windows 7 Ultimate

Resursų saugyklai ir *Operacijų centrui* reikės Java aplinkos ir duomenų bazės (testavimo eigoje bus naudojama MySQL duomenų bazė).

Vartotojo kompiuteriui, kuriame bus leidžiamos klientinės posistemės (*Autoriaus programa* klausimams sukurti, *Studento programa* egzaminams spręsti) reikalinga Java aplinka.

Visus testus kuria ir vykdo sistemos programuotojas.

4.2.7. Testavimo įrankiai ir aplinka

9 lentelė. Automatizuoto testavimo galimybės ir priemonės

Įrankis	Testai	Aprašymas
Eclipse	Vienetų testavimas	Java programavimo aplinka
jUnit	Vienetų testavimas	Atviro kodo įrankis, leidžiantys automatizuoti vienetų testavimą.

4.3. TESTAVIMO PROCEDŪRA

4.3.1. Vienetų testavimas

Vienetų testavimas negali būti vykdomas be išsamaus plano, kuris pateiktas 3 priede.

4.3.2. Priėmimo testavimas

Užsakovo firmoje ir jam dalyvaujant atliekamo testavimo veiksmai. Užsakovas pagal specifikaciją sutikrinus programinės įrangos funkcionalumą nuspręs ar duota programinė įranga atitinka specifikacijoje nurodytą funkcionalumą. Jei bus pateikti smulkūs pageidavimai kaip viena ar kita funkcija turėtų būti realizuojam, šie pakeitimai bus įvykdyti. Jei pakeitimai reikalaus daugiau laiko ir resursų, šie reikalavimų pakeitimai bus perkelti į kitą programinės įrangos versiją.

4.3.3. Aukšto lygio testavimas

Saugumo testavimo metu atliekami veiksmai parodyti lentelėje žemiau.

10 lentelė. Saugumo testavimas

Nr.	Veiksmų grupė	Prisijungimo informacija
	Testavimo atvejis	Laukiamas rezultatas
1.	Tikrinama ar vyksta vartotojo autorizacija operacijų centro posistemėje.	Vartotojui leidžiama atlikti veiksmus tik po autorizacijos.
2.	Tikrinimas ar privati vartotojo informacija tinkamai saugoma sukurtoje programinėje įrangoje.	Vartotojo slaptažodis duomenų bazėje turi būti saugomas užkoduotas md5 algoritmu (arba kitu vienkrypčiu algoritmu).

4.3.4. Testavimo resursų paskirstymas

Operacijų centras bus įdiegtas ir testuojamas kompiuteryje Nr. 1, kurio konfigūracija aprašyta 2.3 skyriuje.

Resursų saugykla bus įdiegta ir testuojam kompiuteryje Nr. 2, kurio konfigūracija aprašyta 2.3 skyriuje.

Vienetų testavimui Resursų saugyklai ištestuoti reikalingi šie įrankiai: paleistas ir tinkamais sukonfigūruotas MySQL serveris, Java 1.6 SDK, jUnit, Eclipse, Ant

Vienetų testavimui Operacijų centrui ištestuoti reikalingi šie įrankiai: paleista ir tinkamai sukonfigūruota Resursų saugykla, paleistas ir tinkamais sukonfigūruotas MySQL serveris, Java 1.6 SDK, jUnit, Eclipse, Ant.

Priėmimo, integravimo ir aukšto lygio testavimui ištestuoti reikalingi šie įrankiai: paleista ir tinkamai sukonfigūruota Resursų saugykla, paleistas ir tinkamai sukonfigūruotas Operacijų centras, paleistas ir tinkamai sukonfigūruotas MySQL serveris, Java 1.6 SDK.

- Studento ir Autoriaus programos.

Bet kokiam iš aukščiau nurodytų testavimų reikalingas kompiuteris su standartinėmis įvedimo ir išvedimo įrenginiais (pelė, klaviatūra, vaizduoklis). Matuojant greitaveiką ir našumą MySQL serveris turi būti paleistas atskirame (angl. dedicated) serveryje (kompiuteryje).

4.4. VIENETŲ TESTAVIMO REZULTATAI IR IŠVADOS

Vienetų testavimui buvo vykdomi automatizuoti testai panaudojant jUnit, kuris yra integruotas į kūrimo terpę Eclipse.

Testavimo metu buvo fiksuojamas kiekvieno testo vykdymo laikas, kadangi šio testo vykdymui reikalingų duomenų sukūrimas buvo vykdomas tame pačiame teste, šis laikas neparodo atskirai testuojamos operacijos trukmės.

Testuojami buvo resursų saugyklos ir operacijų centro žemiau išvardintos funkcijos.

Testuojamos šie vienetai:

- Mokymosi resursų (variantų) valdymas: mokymosi resurso saugojimas, mokymosi resurso gavimas, mokymosi resurso naikinimas;
- Prieinamų scenarijų gavimas (GetAvailableScenarios);
- DAO objektų sukūrimas;
- Factory sukūrimas;
- Funkcijų log metodų funkcionavimas;
- Variantų paieška: pagal raktą, pagal papildomą informaciją, pagal viską.

Testavimo atvejai buvo sudarinėjami baltos (angl. whitebox) ir juodos (angl. blackbox) dėžės metodais. Testavimo rezultatai pateikti lentelėse 5 priede „Testavimo rezultatai“.

Testuojant posistemių funkcijas buvo aptinkamos klaidos ir netikslumai, kurie buvo šalinami, kol testo rezultatas nebuvo sėkmingas. Testavimo rezultatai yra pateikti 4 ir 5 prieduose.

Testavimo metu nustatyta, kad operacijų vykdymo greitis yra priimtinas.

5. PASKIRSTYTŲ MODULIŲ KOKYBĖS ĮVERTINIMAS. PRIĖMIMO TESTAVIMAS

Šiame skyriuje pateikiamas sukurto projekto kokybės vertinimas, kuris apima kokybės įvertinimo rezultatus ir išvadas. Taip pat šis skyrius pateikia neišspręstus klausimus, svarbius pakeitimus, kurie buvo atlikti projekto sukūrimo metu, teikia informaciją kitiems susijusiems projektams, nustato būtinus pakeitimus prieš pateikiant projekto realizaciją galutiniam vartotojui. Remiantis šiuo skyriumi galima spręsti apie dokumentacijos ir funkcionalumo pilnumo įgyvendinimą. O pilnumas kaip jau buvo minėta ankščiau yra vienas ir kokybės faktorių.

5.1. REALIAI ATLIKTO DARBO KOKYBĖS ANALIZĖS TIKSLAI

Realiai atlikto darbo kokybės analizė apima:

- Klaidų aptikimas funkcionalume, logike, realizacijoje.
- Tikrinama ar programų sistema atitinka reikalavimų specifikacijoje.
- Jei sistema buvo kuriama pagal standartus, įsitikinti, kad šių standartų buvo laikymąsi.

5.2. APTIKTOS KLAIDAS MODULIŲ FUNKCIONAVIME, LOGIKOJE, REALIZACIJOJE

Klaidos programinės įrangos funkcionavime, logikoje, realizacijoje buvo aptiktos panaudojant metodologija, kuri aprašoma testavimo aukščiau skyriuje. Ar programa atitinka reikalavimų specifikacija buvo testuojama baltos dėžės vienetų testavimo metu, taip pat naudojant integravimo ir priėmimo testavimą.

5.3. KOKYBĖS VERTINIMO PROCESAS

5.3.1. Peržiūros

Projekto eigoje projekto kūrėjai atlikdavo susitikimus su užsakovu ir kitais suinteresuotais asmenimis, kurio metu buvo surenkama informacija apie tai ką jau pavykdo įvykdyti e-mokymo modulių kūrimo metu ir kas bus vykdoma toliau, kokie tobulinimai bus atliekami, arba kurių ankščiau pateiktų reikalavimų ar panaudojimo atvejų realizacijos reikėtų atsisakyti atsižvelgiant į pasikeitusias galimybes arba reikalavimus.

5.3.2. Interviu su užsakovu.

Projekto kūrimo metu vyko reguliarūs susitikimai su užsakovu.

Buvo numatytas galimų pakeitimų sąrašas:

- Konsolinis (arba su paprasta grafine sąsaja) OC posistemės klientas, pagrindinių funkcijų valdymui.
- Testinio kliento, kuris yra suprojektuotas projekto metu realizacija.

5.3.3. Projektavimo komandos narių peržiūrų aprašymas

Pagrinde kokybės vertinimu užsiima projektavimo komanda.

5.3.4. Rolės ir atsakomybė

Vykdant paskirstytų modulio kūrimą, projektavimo, programavimo ir testavimo darbais užsiima vienas asmuo.

5.3.5. Apklausų anketos

Kai kurios sukurtos sistemos charakteristikos buvo įvertinamos pildant 7 priede pateiktas anketas.

5.4. FORMALIOS TECHNINĖS PERŽIŪROS

Formalios techninės peržiūros yra skirtos nustatyti ar korektiškai buvo sukurta programinė įranga. Šia peržiūra užsiima projektavimo ir programavimo komandos nariai. Dėmėsis buvo skiriamas į išeities kodų struktūrą, komentarus.

5.5. VERTINIMO REZULTATAI

- Užsakovas buvo iš dalies patenkintas sukurta sistema, buvo atkreiptas dėmėsis į tai, kad turi būti realizuoti testiniai klientai, kurių pagalba galima būtų dalinai valdyti sukurtas posistemes, kuriose veiksmus galima būtų funkcijas kviesti norimą tvarką. Toks klientas buvo realizuotas, jo pagalba buvo atliktas 2 priėmimo testavimo etapas (5 priedas), kuris parodė, kad visos testuojamos funkcijos veikia korektiškai.
- Užsakovas buvo patenkintas scenarijų realizacija ir kitų paminėtų šiame projekte panaudojimo atvejų realizacija.
- Formalios techninės peržiūros metu buvo nustatyta, kad kai kurios funkcijos yra nepakankamai komentuotos.
- Vertinimo metu programa veikė stabiliai. Testavimo rezultatai parodė, kad operacijų vykdymo laikas yra priimtinas.
- Testiniai klientai pagal užsakovo reikalavimą buvo realizuoti, posistemė buvo sėkmingai ištestuota panaudojant šių klientų funkcionalumą.

6. IŠEITIES KODŲ METRIKŲ IR CHARAKTERISTIKŲ TYRIMAS

Dalis sukurtos programinės įrangos kokybės įvertinimo jau buvo atlikta vykdant įvairius testavimo metodus bei formalias technines peržiūras. Šių įvertinimų rezultatai pateikiami Rezultatų įvertinimo skyriuje.

Sukurtos programinės įrangos kokybės įvertinimui ir analizei galima naudoti statinę programinės įrangos kodo analizę – tai programinės įrangos metrikų išskaičiavimas ir interpretacija. Programų metrika – tai programos arba jos dalies tam tikros savybės (charakteristikos) kiekybinė (išmatuojama) išraiška. PĮ metrikos leidžia išmatuoti ne tik PĮ kokybę, bet ir jos sukūrimo procesą bei pateikti rekomendacijas, kurios gali padėti patobulinti sukurta PĮ. Pagal IEEE standartą „Standart of software Quality Metrics Methodology“, PĮ metrikos tai funkcija, kur įėjime programinės įrangos informacija, o išėjime reikšmė, kuri apsprendžia kaip duotas atributas įtakoja PĮ [26].

Tyrimai, kurie yra pagrįsti metrikomis prasidėjo 70ais. Kodo eilučių skaičių (angl. Line Of Codes, trump. LOC) pirmas pasiūlė Wolverton 1974 metais tam, kad išmatuoti programuotojų pajėgumo santykį. Jis pasiūlė naudoti kriterijų asmuo/mėnesiui kaip metrikos vienetus. Vėliau atsirado ir daugybė kitų PĮ metrikų. Tyrimo tikslas yra analizuojant sukurto modulio PĮ charakteristikas ir metrikas įvertinti atitikimą ankščiau išvardintiems kokybės faktoriams - sudėtingumą ir priežiūros kiekį, atlikti palyginimą tarp sukurto modulio ir ankstesnės TestTool serverio administravimo posistemės versijos. Kadangi 5.2 versijos tyrimo tikslas tik palyginimas su sukurta, labai plačiai jos netirsime, palyginsime tik pagrindines kodo charakteristikas, leidžiančias nustatyti skirtumus tarp esamos ir buvusios versijos. Priešingai, kai kurias 5.3 versijos charakteristikas pažiūrėsime šiek tiek giliau, kad galima būtų pateikti rekomendacijas patobulinimams. Prieš pradedant nagrinėti tyrimo rezultatus būtina susipažinti su žemiau pateiktomis metrikų ir charakteristikų apibrėžimais.

- Klasių kiekį (angl. number of classes) – bendras klasių kiekis pasirinktoje srityje.
- Vaikų kiekis (angl. number of childrens, trump. NOC) – kiekis tiesioginių klasės sudėtinių klasių. Klasė, kuri realizuoja sąsają yra skaičiuojama, kaip tiesioginis šios sąsajos vaikas.
- Sąsajų kiekis (angl. number of interfaces, trump. NOI) – sąsajų kiekis pasirinktoje srityje.
- Paveldėjimo medžio gylis (angl. depth of inheritance tree, trump. DIT) – atstumas nuo klasės objekto paveldėjimo hierarchijoje.
- Perrašytų metodų kiekis (angl. number of overridden methods, trump. NORM) – Bendras kiekis metodų pasirinktoje srityje, kurie perrašyti nuo protėvio klasės.

- Metodų skaičius (angl. number of methods, trump. NOM) – bendras metodų skaičius pasirinktoje srityje.
- Laukų kiekis (angl. number of fields) – bendras laukų kiekis pasirinktoje srityje.
- Kodo eilučių kiekis (angl. lines of code), ši charakteristika padalinta į dvi:
  1..1. Bendras eilučių kiekis (angl. total lines of code, trump. TLOC), kuris parodo netuščias ir ne komentarų eilutes.
  1..2. Metodo eilučių kiekis (angl. method lines, trump. MLOC), bus paskaičiuoti ir susumuoti netuščios ir ne komentarų eilutės metodo viduje.
- McCabe ciklomatinis sudėtingumas (angl. mcCabe Cyclomatic Complexity) - nepriklausomų kelių programos kodo grafe skaičius.
- Pasvertų metodų skaičius klasėje (angl. Weighted Methods per Class, trump. WMC) – suma visų McCabe ciklomatinių metodų sudėtingumų klasėje.
- Metodų rišlumo stoka (angl. Lack of Cohesion of Methods, trump. LCOM) -
- Aferentinis (įcentrinis) susietumas (angl. Afferent Coupling, trump. Ca) – kiekis klasių paketo išorėje, kurie priklauso nuo klasių paketo viduje.
- Eferentinis (išcentrinis) susietumas (angl. Efferent Coupling, trump. Ce) – kiekis klasių paketo viduje, kurie priklauso nuo klausių paketo išorėje.
- Nepastovumas (angl. Instability, trump. I) – Ce / (Ca + Ce) – ši metrika priklauso nuo aferentinio ir eferentinio susietumo, kinta [0;1]. Jei I = 0, tai parodo maksimaliai stabilų paketą. Jei I=1, tai maksimaliai nestabilus paketas.
- Abstraktumas (angl. Abstractness, trump. A) – Abstrakčių klausių kiekis (ir sąsajų) padalintas iš visų tipų kiekio pakete.
- Normalizuotas atstumas nuo pagrindinės sekos (angl. Normalized Distance from Main Sequence) - |A + I – 1|, šis skaičius turėtų būti mažas, jei paketų architektūra yra gera artimas nuliui.
- Priklausomybių tarp paketų grafas – parodo labiausiai priklausomus paketus ir kreipinių kiekį iš vieno paketo į kitą.

6.1. TYRIMUI ATLIKTI NAUDOJAMŲ ĮRANKIŲ IR PRIEMONIŲ APRAŠYMAS

Programinės įrangos metrikų skaičiavimui panaudoti šie įrankiai:

Metrics 1.3.6 - tai specialus Eclipse programavimo aplinkos įskiepis. Jis realizuoja metrikų skaičiavimus, kurie aprašyti Brian Henders-Sellers [27], o ataskaitą gali pateikti XML arba CSV formatu. Programos sąsajos kalba – anglų. Jis yra atviro kodo ir nemokamas. Panaudojant šį įrankį nustatysime daugumą metrikų ir charakteristikų reikšmių.

Kitas panašus įrankis JHawk, priešingai nuo Metrics yra mokamas, turi skirtingas pirkimo licencijas, tyrime jo nenaudosime, nors ir prieinama šio įrankio demo versiją Eclipse įskiepio pavidalu. Pagrindinės demo versijos trūkumas yra tas, kad vienu metu gali analizuoti tik pora Java failų arba vieną paketą, dėl to negalėsime įvertinti viso projekto. Šis įrankis skaičiuoja dauguma panašių metrikų kaip ir Metrics įrankis, tuo pačiu ir Halstead'o sudėtingumo metrikas, ko Metrics neskaičiuoja.

Priklausomybėms tarp paketų tirti ir nagrinėti gerai tinka įrankiai:

- Understand - tai tarpplatforminis, į PĮ palaikymą orientuota kūrimo aplinka, skirta padėti prižiūrinčiam personalui geriau vidinę PĮ sudėtį ir priklausomybes. Šis paketas yra mokamas, bet prieinama 15 dienų bandomoji versija su neribojamu funkcionalumu. Šis įrankis turi savo grafinę aplinką, leidžia piešti įvairius priklausomybės grafikus, UML diagramas, vykdyti metrikų analizes ir formuoti ataskaitą CSV arba HTML formatu. Šia priemone nubraižysime priklausomų paketų grafą.
- CAP (angl. Code Analysis Plugin) įrankis. Šio įrankio autoriai yra Johannes Schneider ir Matthias Mergenthaler, jis platinamas nemokamai pagal GPL licenciją. Pateikiamas kaip Eclipse įskiepys, deja naudotis juo nėra ypač patogu, nes jis paketus nagrinėja visumoje, įskaitant tuos, kurių nėra tiesiogiai išeities koduose, bet jie yra aprašyti arba paskelbti. Kadangi nagrinėjamas objektas yra paskirstyta sistema, natūralu, kad daugelyje klasių egzistuoja kitų posistemių aprašai arba kreipiniai į juos. Dėl to šio įrankio metrikų paskaičiavimai yra nekorektiški. Iš kitos pusės šio įrankio pateikiamos priklausomybių diagramos yra labai informatyvios.

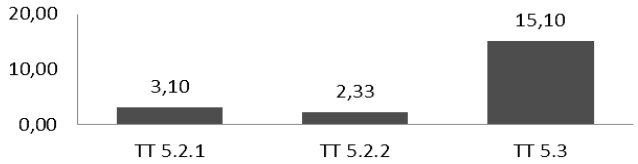
Metrikų reikšmes 2D stulpelinių diagramų ir lentelių pavidalų braižysime Microsoft Office Excel paketu.

6.2. OPERACIJŲ CENTRO PROGRAMINĖS ĮRANGOS KOKYBĖS TYRIMAS, REMIANTIS IŠEITIES KODŲ METRIKOMIS.

Ankstesnių TestTool versijų (5.2.1 ir 5.2.2) architektūra ir kodo struktūra smarkiai skiriasi nuo naujai sukurtos 5.3 – naudojami skirtingi programavimo šablonai, paketų struktūra, programavimo įrankiai ir technologijos. Nepaisant to, mes vis tiek galime palyginti šių posistemių išeities kodo metrikas, panaudojant tuos pačius įrankius, kurios aprašėme aukščiau.

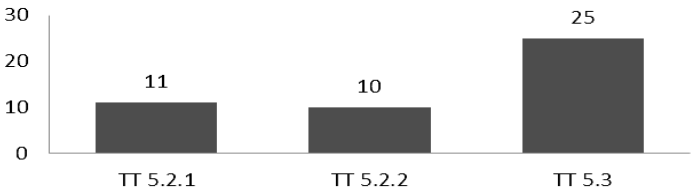
Pradžioje palyginsime bendras dydžio ir apimties metrikas.

Naudojamų bibliotekų svoris pateiktas diagramoje žemiau.



18 pav. Bibliotekų užimama vieta kietajame diske(MB)

Naudojamų papildomų bibliotekų kiekis pateiktas diagramoje žemiau.

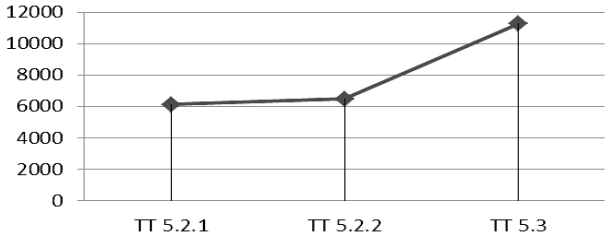


19 pav. Bibliotekų skaičius, vnt.

Kadangi TestTool 5.3 naudojama daug papildomų įrankių ir programinių karkasų, tai įtakoja priklausomų bibliotekų kiekiui, kurios yra reikalingos programų darbui. Galiausiai dėl to išauga programų užimama vieta. Nors atsižvelgiant į dabartinių kompiuterių kietųjų diskų apimtis, šis skirtumas neturi ypatingos įtakos.

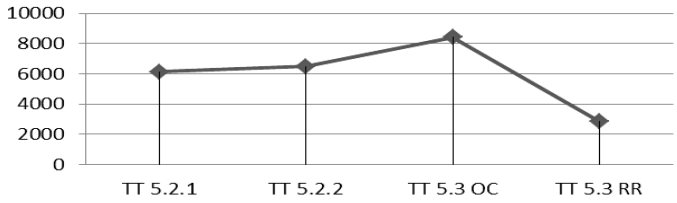
- Serverių dalių apimtis, viso išeities kodų eilučių kiekis (angl. trump. LOC).

Atvaizduosime šiuos duomenis diagramoje žemiau.



20 pav. Kodo eilučių skaičius skirtinguose TestTool serverio realizacijose

Matome, kad posistemių apimtis tobulinant aplikaciją nuo 5.2.1 iki 5.2.2 versijos didėjo nežymiai. Bet pasiūlius naujos architektūros paskirstyto modelio posistemę kodo apimtys išaugo žymiai. Šis pasikeitimas įvyko dėl to, kad žymiai padaugėjo posistemės funkcijų. TT 5.3 versijoje į bendra eilučių kiekį įskaičiuotas operacijų centro ir resurso saugyklos (RS) apimtis, jei RS iškelsime į atskirą posistemę tai gausime pakankamai nežymius apimties pasikeitimo duomenis. Diagrama pavaizduota žemiau.



21 pav. Kodo eilučių skaičius skirtinguose TestTool serverio modulių realizacijose

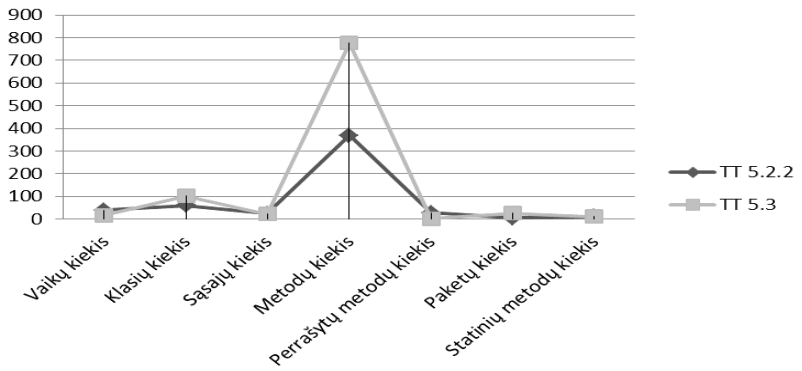
Padaugėjusią kodo apimtį TT 5.3 versijoje taip pat gali paaiškinti naudojami programų veikimo stebėjimo įrankiai, kurie žymiai prideda perteklinio kodo. LOC įvertinimas be šio funkcionalumo yra neįmanomas, bet jis galėjo parodyti daug mažesnius kodo apimtis. Iš kitos pusės natūralu, kad augant funkcionalumui auga ir programos apimtys.

5.2.1 ir 5.2.2 versijos skiriasi nežymiai, dėl to toliau lyginsime tik dvi versijas: 5.2.2 ir 5.3. Ligindami kiekio metrikas, skaitysime OC ir RS modulius kaip vientisą sistemą. Jei liginsime tik OC posistemę ir 5.2.2 versiją, tai skirtumai bus nežymus, tik metodų ir paketų skaičius naujoje versijoje dėl padaugėjusio funkcionalumo bus didesnis. Todėl ***11*** lentelėje 5.3 versijos kiekybines metrikas pamatuosime bendrai RS ir OC moduliui. Tolimesnėje analizėje bus lyginama tik 5.2.2 (trump. 5.2) versija su Operacijų centrų (toliau bus vadinama kaip 5.3 versija).

11 lentelė. Kiekybinės metrikos

	TT 5.2.2	TT 5.3
Vaikų kiekis	40	17
Klasių kiekis	61	103
Sąsajų kiekis	25	22
Metodų kiekis	369	777
Perrašytų metodų kiekis	28	2
Paketų kiekis	6	26
Statinių metodų kiekis	11	11

Pavaizduosime šias metrikas diagramoje, kad galima būtų vaizdžiau pamatysi skirtumus.



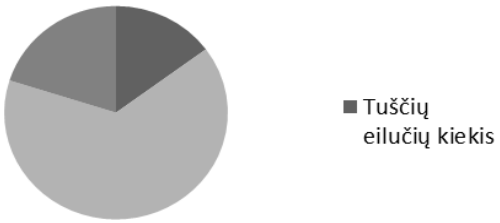
22 pav. Kiekybinių metrikų palyginimas

Matome, kad labiausiai padaugėjo metodų ir klasių kiekis, kadangi naujos posistemės turi didesnį funkcionalumą. Bendros kiekio metrikos pateiktos 12 lentelėje.

12 lentelė. Įvairios kiekybinės metrikos

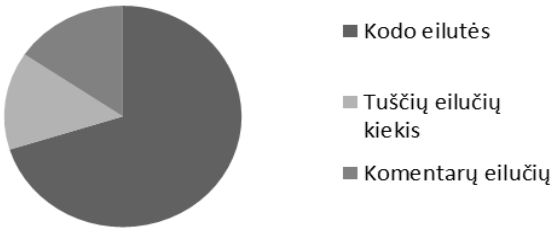
Metrika	5.3 versija	5.2 versija
Vaikų kiekis	5	40
Klasių kiekis	60	61
Sąsajų kiekis	17	25
Metodų kiekis	565	369
Perrašytų metodų kiekis	0	28
Paketų kiekis	14	6
Statinių metodų kiekis	5	11
Kodo eilutės	8307	6.131
Tuščių eilučių kiekis	1900	1.250
Komentarų eilučių	2568	1.358
Komentarų ir kodo santykis	30.91%	22,15%

OC kodo eilučių pasiskirstymas pateiktas *23* pav., 5.2.2 - *24* pav.



23 pav. Kodo pasiskirstymas 5.3 OC versijoje

Kaip matome komentarai sudaro šiek tiek daugiau nei 30% nuo viso modulio kodo.



24 pav. Kodo pasiskirstymas 5.2.2 versijoje

5.2.2 Komentarų kiekis užima dar mažesnę kodo dalį – tik apie 22%.

- Atributų kiekis (angl. Number ofAttributes)

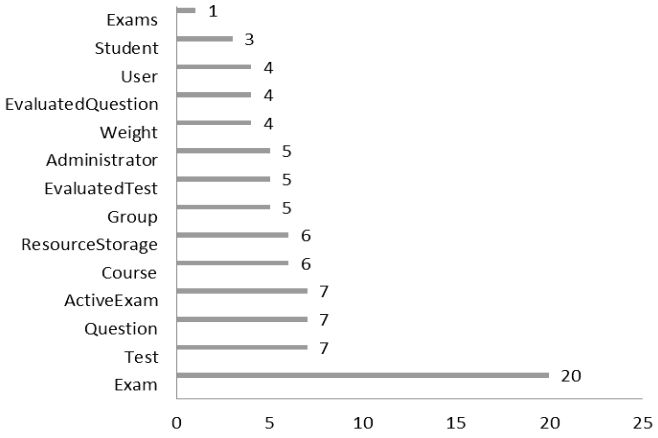
Pateiksime didžiausia kiekį atributų turinčias klases lentelėje *13*.

13 lentelė. Klasių atributai

Klasė	Reikšmė
Exam	20
TOExam	18

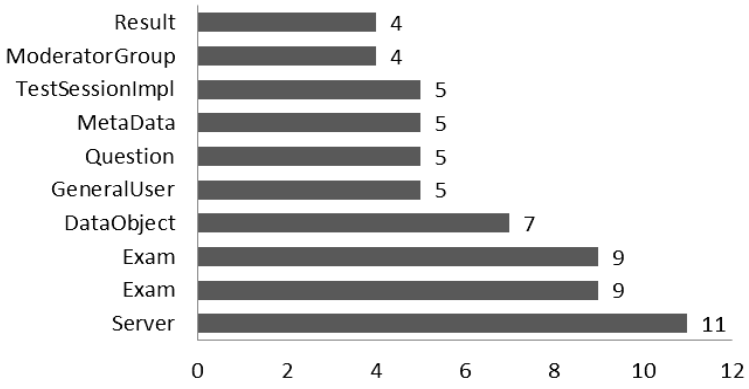
OCStudentImpl	16
OCServerImpl	15
EvaluatedVariant	8
ActiveExam	7
Question	7
Test	7
Course	6
ResourceStorage	6

Didžiausią atributų kiekį tiriamajame objekte turi duomenų klasė, palyginsime su kitais duomenų objektais, pateikdami duomenų objektus *25* paveiksle.



25 pav. Duomenų klasių atributų kiekis

Akivaizdu, kad viena iš duomenų klasių turi pernelyg daug atributų palyginus su kitomis, tai gali sąlygoti apie klaidingai suformuota struktūrą šioje klasėje, rekomenduojama ją išnagrinėti ir pagal galimybes išskaidyti. Iš kitos pusės aiškiai apibrėžtų ribų ši metrika neturi. 5.2.2 versijoje situacija parodyta *26* pav., panašu, kad atributai yra pasiskirstė tolygiai klasėse.



26 pav. 5.2.2 versijos posistemės klasių atributų kiekis

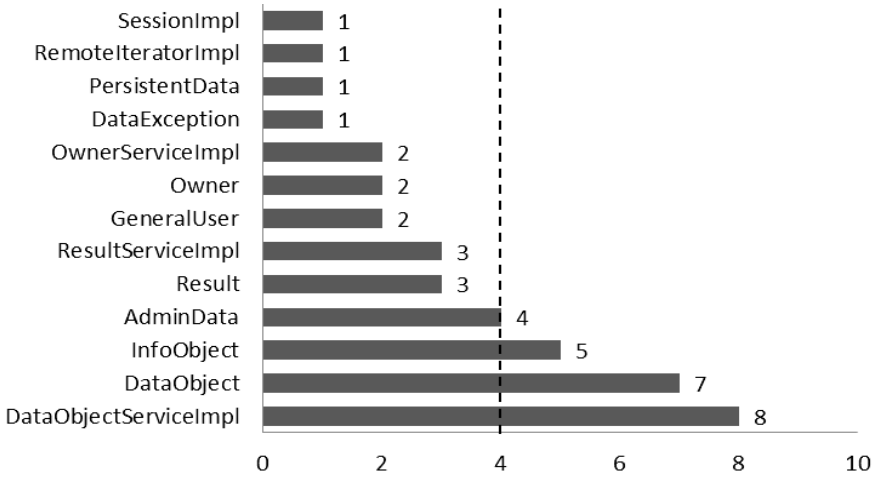
- Vaikų kiekis (angl. Number of Children, trump. NOC)

Aukšta NOC reikšmė rodo mažesnę klaidų tikimybę, tačiau aukštos NOC ir WMC (Metodų skaičius klasėje) reikšmės rodo didelį bazinės klasės sudėtingumą. Tokiu atveju, bazinę klasę reikia restruktūrizuoti. Paprastai rekomenduojami limitai yra nuo 1 iki 4.

Nagrinėjame projekte paveldėjimą naudoja tik dvi klasės, User ir TOUser klasės, šios klasės turi po 2 vaikines klases.

Viena iš teorijų kodėl naudinga naudot paveldėjimą siejasi su tuo, kad paveldėjimas skatina pernaudoti jau realizuotus objektus. Mūsų sistemoje paveldėjimas taikomas pagal poreikius ir galimybes. Mažesnis paveldėjimų kiekis leidžia nevykdyti sudėtingo testavimo.

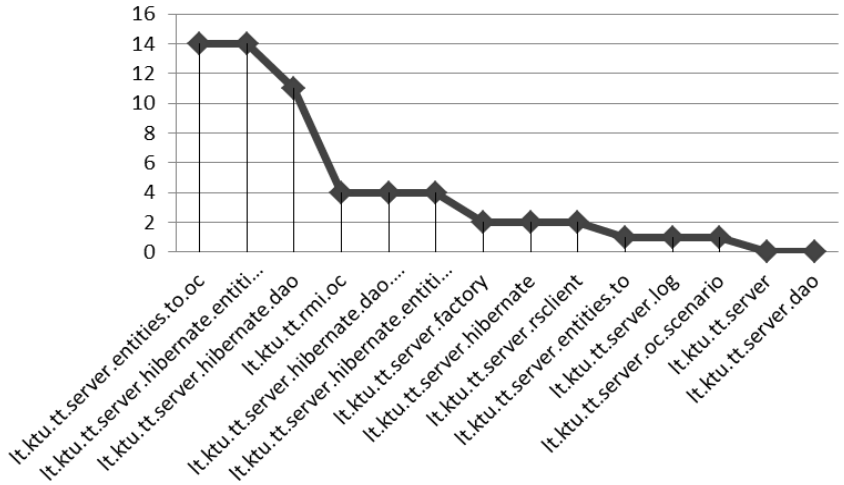
5.2.2 versijoje keliose klasėse vaikų kiekis viršija rekomenduojamas normas, kas didina sudėtingumą ir kodo suprantamumą, o tuo pačių ir testavimą. Kuriant arba tobulinant priklausomas nuo žemiau pateiktų klasių funkcijas klaidos tikimybė smarkiai padidėja.



27 pav. 5.2.2 versijos posistemės klasių vaikų kiekiai

- Klasių kiekis (angl. Number of Classes, trump. NOC)

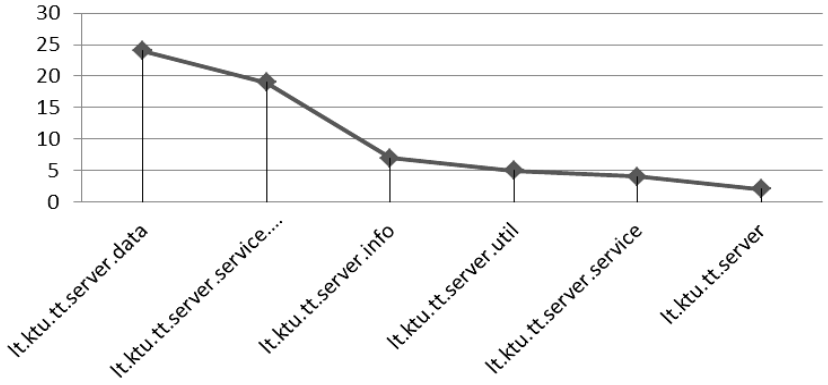
Klasių kiekis pagal paketus parodytas *28* pav.



28 pav. Klasių kiekis pagal paketus

Kai kurie paketai turi tik po 1 klasę, tai gali rodyti perteklinį paketų skaičių, bet kadangi sukurta PĮ yra orientuota į ateities tobulinimą, numatoma, kad šie paketai ateityje turės daugiau klasių. Paketai atsakingi už sąsajų saugojimą nesaugo klasių, dėl to diagramoje

lt.ktu.tt.server.dao ir lt.ktu.tt.server neturi klasių. Klasių pasiskirstymas 5.2.2 versijoje parodytas *29* pav.

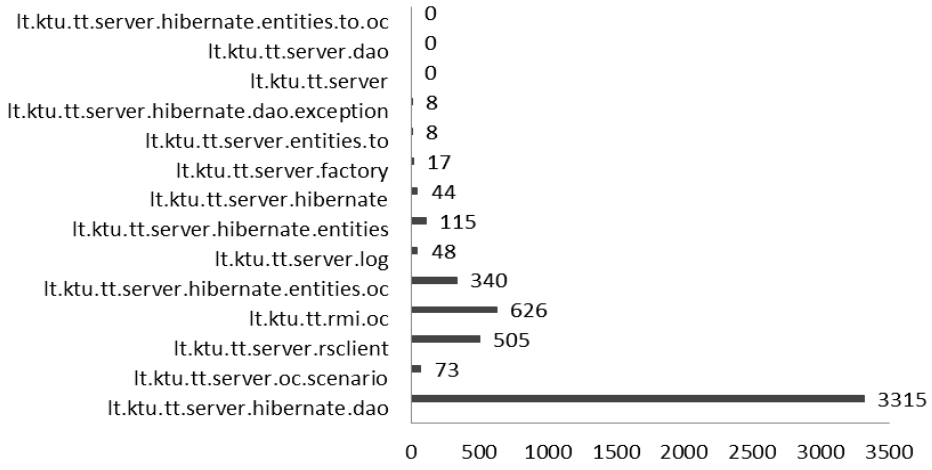


29 pav. 5.2.2 versijos posistemės klasės pagal paketus

Pasiskirstymo kreivė yra panaši į 5.3 versiją, tik paketų atitinkamai dvigubai mažiau.

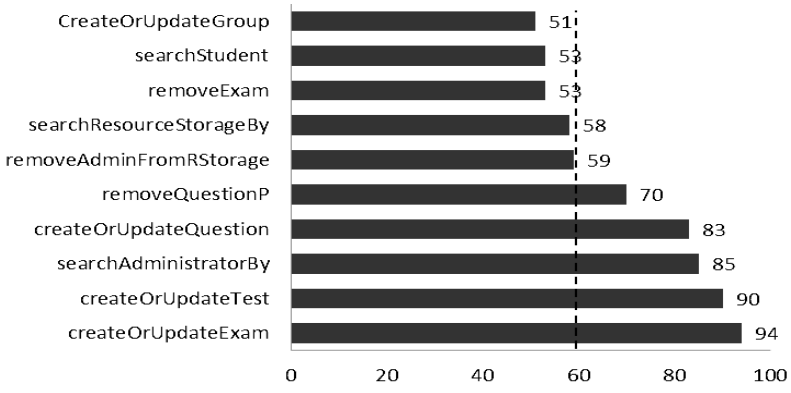
- Metodų eilučių kiekis (angl. Method Lines of code, trump. MLOC)

Kodo eilučių kiekis metoduose pagal paketus parodytas *30* pav.



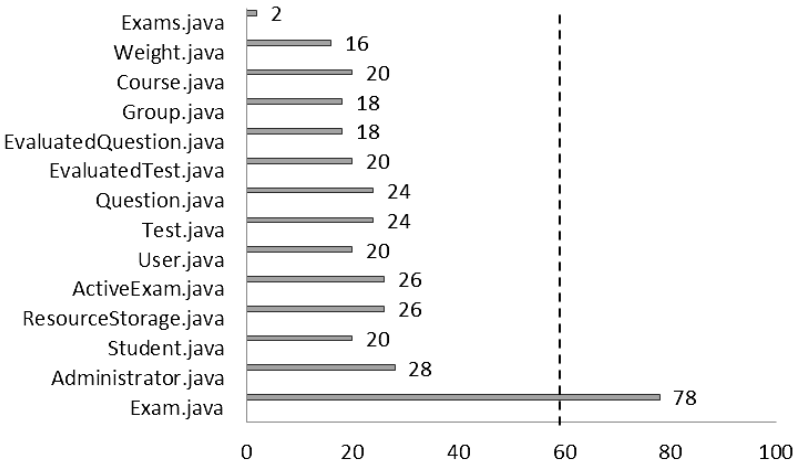
30 pav. Kodo eilučių kiekis pagal paketus

Metodų eilučių kiekis pagrindinėse veiksmų ir duomenų objektų klasėse parodytas atitinkamai ***32*** ir ***33*** paveiksluose. Atsižvelgiant į veiksmų klasių MLOC reikšmes anomalijų nesimato. ***31*** pav. parodyti didžiausi metodai.



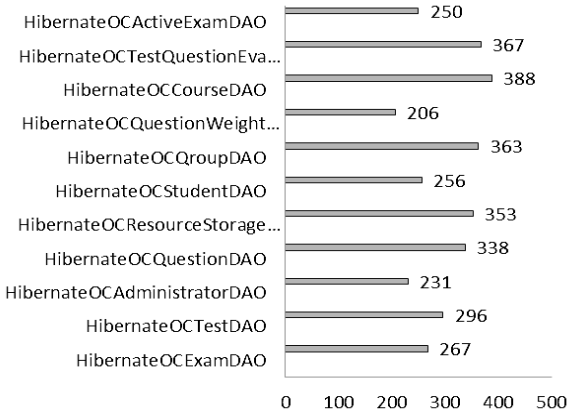
31 pav. Didžiausi metodai

Paprastai yra svarbu kiek eilučių užima vienas metodas, įprasta kad vienas metodas neturėtų viršyti 50-80 eilučių. Mūsų atveju, 5.3 versijoje kiekviename metode įskaitant ir pagrindinę veiklą, dažnai kviečiami sistemos elgsenos stebėjimo mechanizmo funkcijos (log funkcijos), o tai savo ruožtu veda prie daugiariopo metodo eilučių kiekio padydėjimo, dėl to ši metrika daugelyje metodų smarkiai viršija rekomenduotinas normas.



32 pav.Kodo eilučių kiekis duomenų klasėse

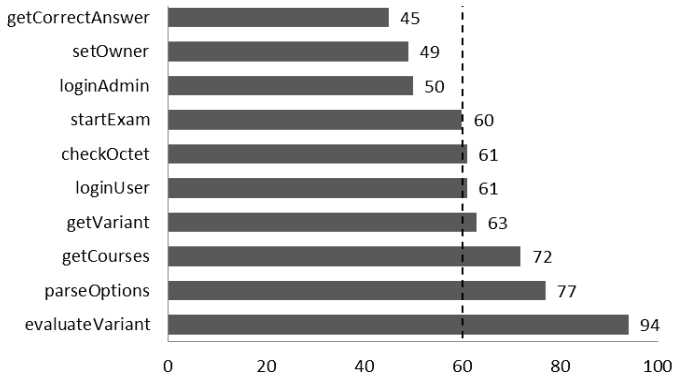
Jei analizuojant išeities kodus galima būtų atmesti šiuos iškvietimus, vidutiniškai metodų apimtis neviršytų 50-70 eilučių. Exams klasė turi 2 kodo eilutes, tai tuščia ir nenaudojama niekur klasė, turėtų būti išimta iš projekto. Klasė Exam turi dižiausią kodo eilučių skaičių, kadangi joje yra daug atributų.



33 pav. Kodo eilučių kiekiai pagrindinėse veiksmų klasėse

Atsižvelgiant į veiksmų klasių MLOC reikšmes anomalijų nesimato.

5.2.2 versijoje naudojamas kitoks programos darbo stebėjimo mechanizmas ir požiūris į stebėjimą, jis ne taip išsamiai stebi programos darbą, todėl jo egzistavimas kode beveik neįtakoja šios metrikos reikšmės. Didžiausi metodai parodyti *34* pav.

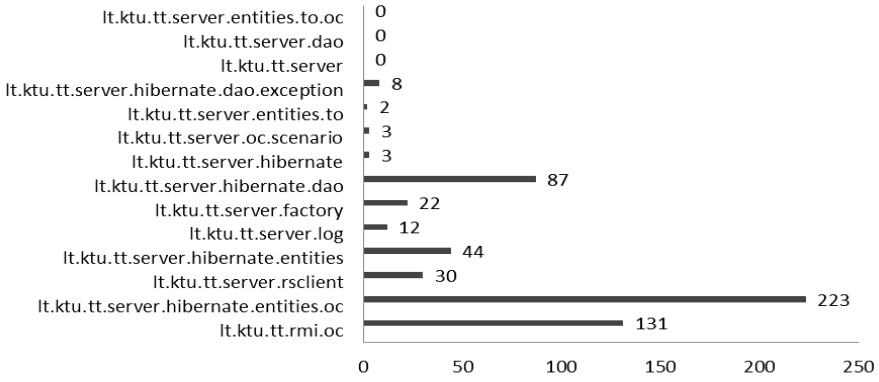


34 pav. 5.2.2 versijos posistemės didžiausi metodai

Matome, kad metodų viršijančių rekomenduojamas normas kiekis yra labai panašus į sukurtos sistemos, bet kadangi čia negalime taikyti prielaidos apie sekimo metodų kodą, tai nurodytų metodų priežiūra yra labai sudėtinga, o klaidos tikimybė tokiame kode ypač aukšta.

- Metodų kiekis (angl. Number of Methods, trump. NOM)

Metodų kiekiui paketuose specialių apribojimų nėra. Metodų kiekis pagal paketus parodytas *35* pav.

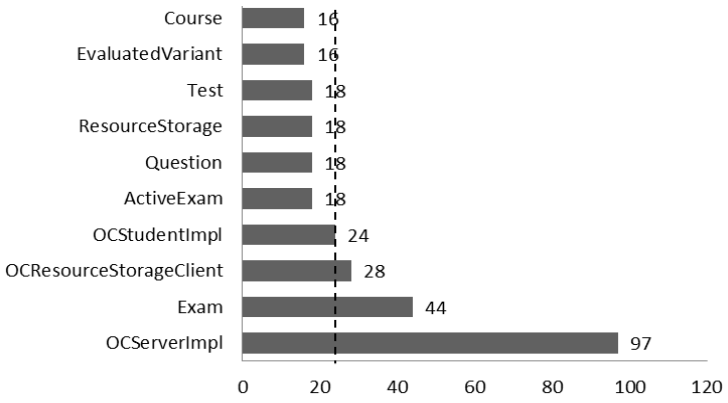


35 pav. Bendras metodų kiekis klasių paketuose

Matome, kad trijuose paketuose yra klasės be metodų arba sąsajos.

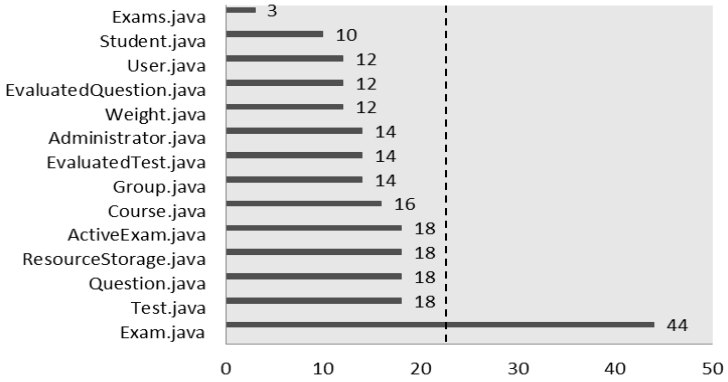
Kuo NOM reikšmė yra didesnė, tuo klasėje gali būti daugiau klaidų, tuo daugiau reikia pastangų klasės sukūrimui ir priežiūrai. Klasės su aukšta NOM reikšme turi būti išskaidytos į kelias mažesnes klases. Rekomenduojama, kad NOM neviršytų 20, o sistemoje būtų ne daugiau kaip 10% klasių, kurių NOM reikšmė yra didesnė nei 24.

36 paveiksle pateikta 10 klasių, su didžiausia NOM metrikos reikšme.



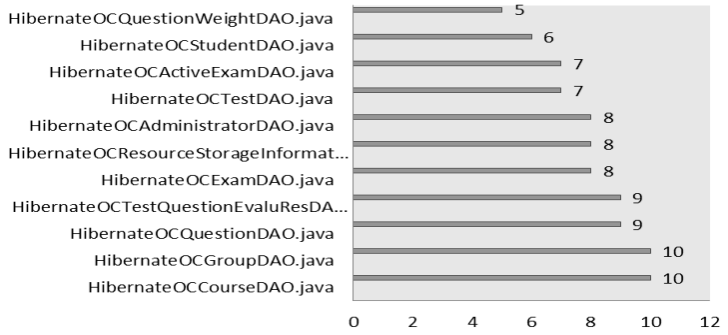
36 pav. Metodų kiekis klasėse

Klasėje OCServerIml yra viršytas rekomenduojamas metodų kiekis klasėje. Metodų kiekis pagrindinių veiksmų ir duomenų klasėse atitinkamai parodyta *37* ir *38* paveiksle.



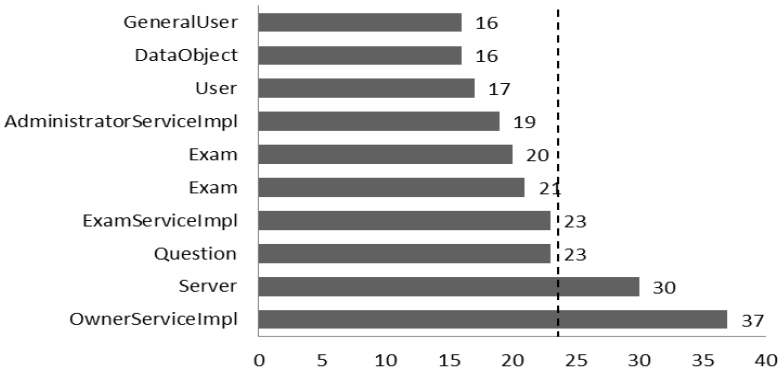
37 pav. Metodų kiekis lt.ktu.tt.server.hibernate.entities.oc pakete

Matome, kad klasėje Exam yra viršytas NOM parametras, kadangi šioje klasėje yra didelis skaičių atributų, o duomenų klasėse naudojami priėjimo metodai (angl. accessors), kurie užtikrina inkapsuliaciją, kuri savo ruožtu suteikia tokias galimybes kaip papildomas funkcionalumo praplėtimas pagal poreikį, paslepia vidinį apibrėžimą, užtikrina paprastesnį klaidos paiešką. Bet dėl šių metodų duomenų failuose kiekvienam privačiam kintamajam prisideda po 2 papildomus metodus, dėl kurių ir matome žymiai didesnį šios metrikos reikšmę.



38 pav. Metodų kiekis pakete lt.ktu.tt.server.hibernate.dao

Pagrindinėse veiksmų klasėse ši metrika neviršija rekomenduojamų ribų. Ši metrika yra viršyta ne daugiau 10% klasių nuo viso nagrinėjamo objekto. Metodų kiekis 5.2.2 versijoje parodytas *39* pav.



39 pav. 5.2.2 versijos metodų kiekiai klasėse

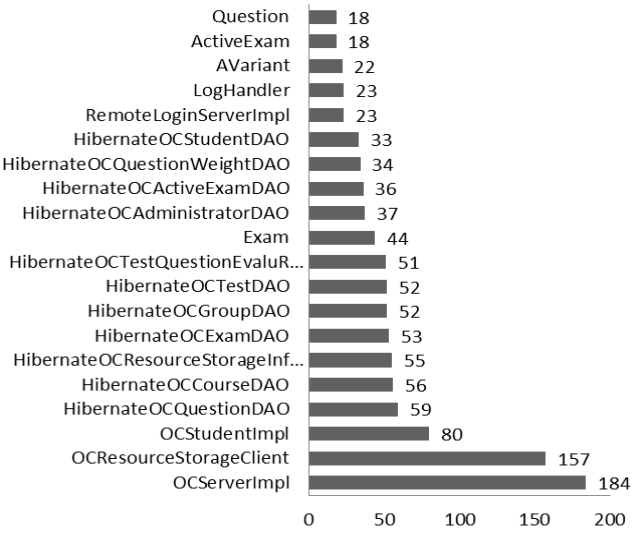
Kaip ir 5.3 versijoje rekomenduojamas metodų kiekis klasės yra viršytas dviem klasėms.

- Pasvertų metodų kiekis klasėje (angl. Weighted methods per Class, trump. WMC)

Šis parametras išskaičiuojamas kaip klasės metodų sudėtingumų suma. Parodo klasės palaikymo ir pakeitimų sudėtingumą. Metodų sudėtingumas skaičiuojamas sumuojant visų klasės metodų McCabe sudėtingumo metrikų reikšmes.

Kuo WMC didesnis, tuo klasėje gali būti daugiau klaidų, tuo daugiau reikia pastangų klasės sukūrimui ir priežiūrai. Klasės su aukšta WMC reikšme turi būti išskaidytos į kelias mažesnes klases. Klasės, kurios turi didesnį kiekį metodų, yra labiau orientuotos į specifinę užduotį ir yra sunkiau panaudojamos pakartotinai.

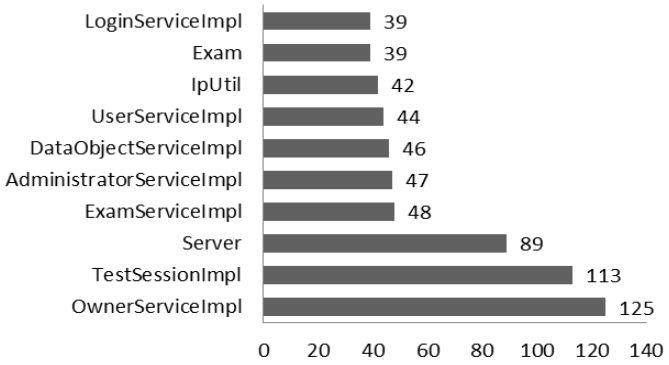
20 klasių su didžiausiu sudėtingumu pateikta ***40*** paveiksle.



40 pav. didžiausios klasių WMC reikšmės

Sudėtingiausia priežiūra ir pakeitimai gali būti OCServerIml ir OCResourseStorageClient klasėse, kitos klasės yra pakankamai lengvos.

5.2.2 versijos WMC reikšmės parodytos *41* pav.



41 pav. 5.2.2 versijos didžiausios klasių WMC reikšmės

- Lizdinių blokų gylis (angl. Nested Block Depth, trump. NBD)

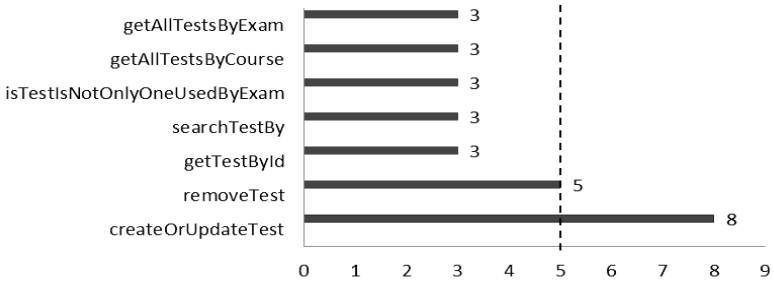
Metrics programos kūrėjai rekomenduoja, kad šis parametras neturėtų viršyti 5. Aukštesnė reikšmė reiškia sudėtingesnę struktūrą, kurioje padidėja klaidos tikimybė bei apsunkina galimą pakeitimą.

10 klasių su aukščiausiu lizdinių bloku gyliu parodytas *14* lentelėje.

14 lentelė. Lizdinių blokų gylių reikšmės skirtingiems klasėms

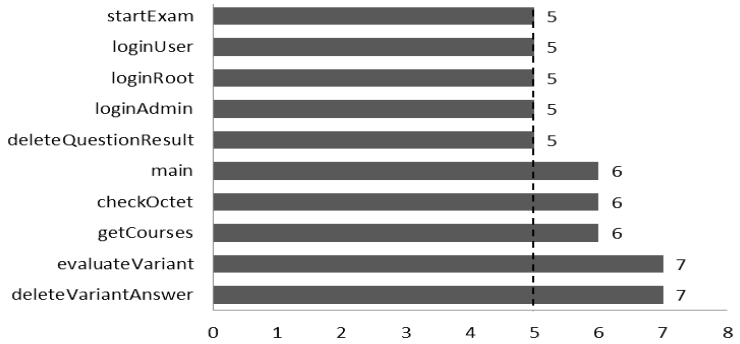
Klasė	NBD reikšmė
createOrUpdateTest	**8**
RemoveAllEvaluationsFromActiveExam	5
getExams	5
searchAdministratorBy	5
createOrUpdateExam	5
getExamListByCourse	5
createOrUpdateQuestion	5
removeQuestionP	5
removeAdminFromRStorage	5
removeTest	5

NBD reikšmė yra viršyta rekomenduojama norma metodui, kuris atsakingas už testo sukūrimą arba atnaujinimą, patariama restruktūrizuoti šią funkciją. Kituose tos pačios klasės metoduose ši metrika neviršija rekomenduojamų (***42*** pav).



42 pav. Lizdinių blokų gylis HibernateOCTestDAO klasėje

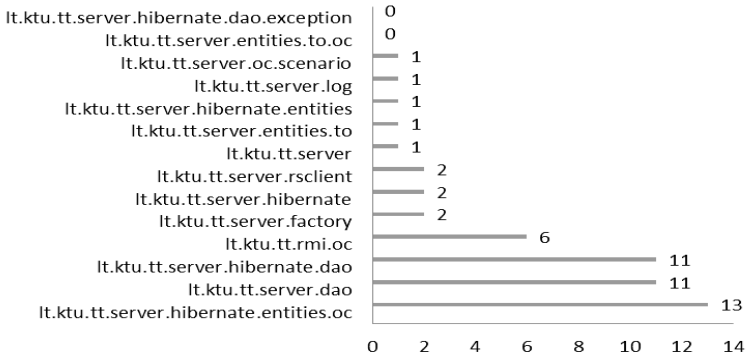
5.2.2 versijos posistemės informacija parodyta ***43*** pav. Ši metrika yra viršyta daugiau klasių nei 5.3 versijoje. Tai reiškia, kad naujai sukurtoje versijoje pavykdo pasiekti lengvesnių metodų. . Kuo daugiau yra lizdinių if/else konstrukcijų to kodas yra sudėtingesnis, tai apsunkina jo skaitomumą, suprantamumą ir galimus pakeitimus.



43 pav. Didžiausios lizdinių blokų gilių reikšmės 5.2.2 versijoje

- Eferentinis (išcentrinis) susietumas (angl. Efferent Coupling, trump. Ec)

Charakterizuoja objektų tipų kiekį, nuo kurių yra priklausomas konkretus tipas, iš esmės mūsų atveju parodo kiekį klasių paketo viduje, kurie priklauso nuo klasių paketo išorėje. Paketų išcentrinis susietumas parodytas 44 pav.

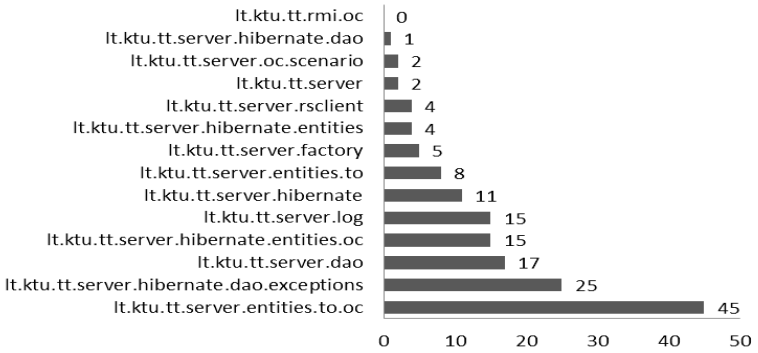


44 pav. Paketų eferentinis susietumas

Matome, kad paketai, kuriose laikomos išimčių klasės ir persiuntimo objektai (angl. transfer object), neturi priklausomų klasių paketo išorėje, tai yra normalu, nes šios klasės yra iškviečiamos iš kitų klasių, bet pačios neturi būti priklausomos nuo jų. Visi kiti paketai turi ryšį su klasėmis iš kitų paketų.

- Aferentinis (įcentrinis) susietumas (angl. Afferent Coupling, trump. Ac)

Įcentrinis susietumas reiškia klasių kiekį paketo išorėje, kurie priklauso nuo klasių paketo viduje. Kai Ac = 0, tai gali reikšti, kad konkretus kodo vienetas nėra naudojamas išorėje. Tiriamos posistemės paketų įcentrinis sujungiamumas parodytas 45 pav.



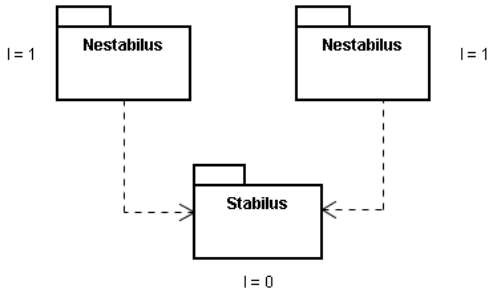
45 pav. Paketų aferentinis susietumas

Aukščiau pateiktoje diagramoje matome, kad tt.rmi.oc paketo Ac = 0, tai reiškia, kad klasės iš šio paketo nėra kviečiamos iš kitų paketų. Tai yra įprasta, kadangi pakete yra metodai per kurios bendraujama su kitomis posistemėmis ir jos yra susietos su klasėmis iš kitų paketų, t. y. Ec≠0.

- Nepastovumas (angl. Instability, trump. I)

Nepastovumas arba nestabilumas (angl. Instability, I) – Ce / (Ca + Ce) – ši metrika priklauso nuo aferentinio (Ca) ir eferentinio (Ce) susietumo, kinta [0;1]. Jei I = 0, tai parodo maksimaliai stabilų paketą. Jei I=1, tai maksimaliai nestabilus paketas.

Galima pravesti priklausomybę tarp abstraktumo ir nepastovumo, jei A=0 ir I=0, tai bus labai stabilus ir konkretus paketas. Toks paketas iš esmės yra nepageidautinas, kadangi jis yra tvirtas ir ne abstraktus, jis negali būti praplėstas.. Taip pat jį yra labai sudėtinga pakeisti dėl jo pastovumo (stabilumo). Paprastai sistemoje gali būti stabilių ir nestabilių paketų. Jei visi paketai būtų stabilūs, tai sistemos būtų neįmanoma pakeisti. Idealius atvejis pavaizduotas 46 pav.[28].

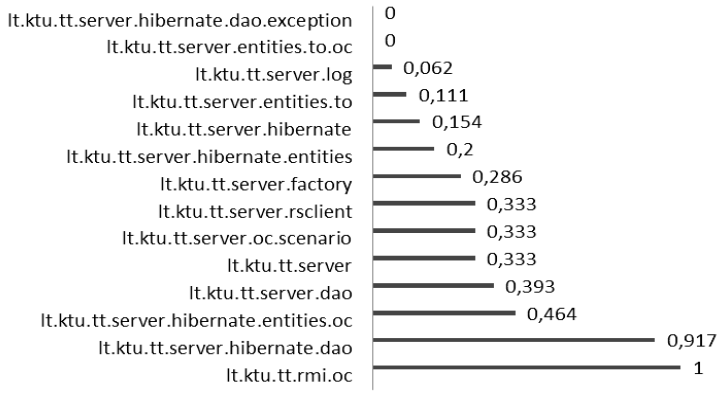


46 pav. Ideali paketų konfigūracija

Jei A=1 ir I=1. Tai bus taip pat nepageidautinas variantas (jei iš vis įmanomas), kadangi maksimalus abstraktumas ir jokio sąryšio. Toks paketas yra taip pat labai tvirtas, nes jo neįmanoma praplėsti. Jei yra paketas su A=0.5 ir I=0.5, tai paketas yra dalinai praplečiamas, kadangi yra pusiau abstraktus ir, kartu, pusiau stabilus taip, kad praplėtimai nebus maksimalaus nestabilumo priežastimi. Toks paketas yra subalansuotas – stabilumas subalansuotas su abstraktumu. Šios charakteristikos yra idealios ir praktikoje beveik

neįmanoma jų pasiekti, paprastai kaip ir mūsų projekte toli gražu ne visi paketai atitinka norimą priklausomybę, tie paketai, kurie yra arčiau pagrindinės sekos turi geriausias charakteristikas[21].

Tiriamo objekto paketų nepastovumo metrikos reikšmės parodytos *47* paveiksle.



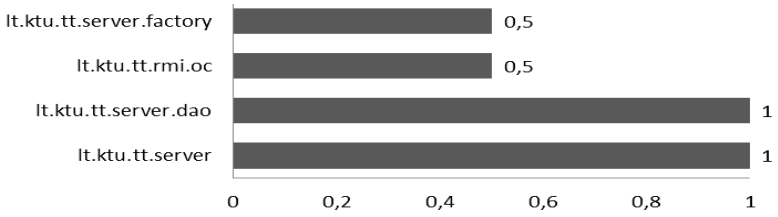
47 pav. Nepastovumo metrikos reikšmės pagal paketus

Maksimaliai nestabilus paketas yra rmi paketas. Iš esmės ši metrika negali turėti kažkokių ypatingų rekomendacijų, dėl geresnių rodyklių, kadangi viskas labai priklauso nuo paketo paskirties, vien tik klasių ir sąsajų buvimas kartu ar nebuvimas iš karto smarkiai įtakoja abstraktumo ir stabilumo metrikas. Kadangi OC modulio paketuose sąsajos ir klasės beveik visada laikomos skirtingose paketuose, tai ir abstraktumo metrikos reikšmės negali būti laikomos pakankama priežastimi projekto pertvarkymui (angl. refactoring). Yra visiškai normalu, kad kai kurie paketai yra daugiau arba mažiau stabilus. Jei kūrėjas nori, kad paketas būtų stabilesnis, tai jis turėtų mažiau priklausyti nuo kitų paketų, kurie priklauso nuo jo. Paketas būna nestabilus jei jo priklausomybė nuo kitų paketų yra didesnė už tą, su kurią tie paketai priklauso nuo nagrinėjamo. Todėl bet kokios priklausomybės pakeitimas gali smarkiai įtakoti visas reikšmes.

- Abstraktumas (angl. Abstractness, trump, A)

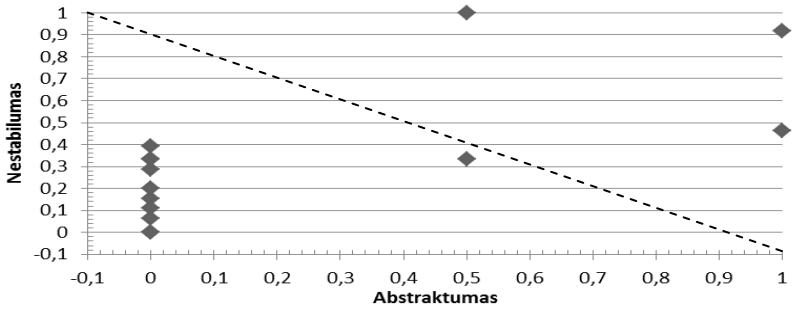
Ši metrika neturi rekomenduojamų ribų, kadangi labai priklauso nuo to, kam paketas yra skirtas. Kaip jau buvo minėta ankščiau labiau abstraktūs paketai yra labiau pageidautini, kadangi turi didesnį lankstumą.

48 diagramoje parodyti paketai kurių abstraktumas yra nenulis.



48 pav. Paketų abstraktumas

Reikšmės keičiasi nuo 0 iki 1. Dauguma tiriamų paketų abstraktumas yra nulinis tai iš esmės reiškia, kad paketus yra sunkiau pakeiti, negriaunant paketų struktūros. Taip pat tai įrodo stabilumo sąryšį tarp paketų, kuris yra daugumoj žemesnis nei vidutinis, o tai reiškia, kad paketai eina link didesnio stabilumo. Sąryšis tarp šių dvejų metrikų parodytas 49 pav.



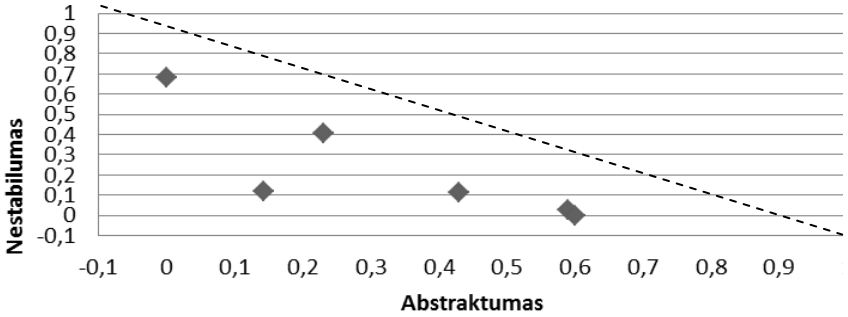
49 pav. Ryšis tarp nestabilumo ir abstraktumo

Ryšis tarp paketų yra pakankamai stiprus. Paketai nėra subalansuoti, nes taškai yra toli nuo pagrindinės sekos – punktyrinės linijos nubrėžtos diagramoje.

Nestabilumo ir abstraktumo reikšmės 5.2.2 versiaji yra pateiktos 15 lentelėje, o ryšis tarp šių metrikų pateiktas paveiksle 50.

15 lentelė. TestTool 5.2 Abstraktumo ir nestabilumo metrikų reikšmės

Paketas	Abstraktumas	Nestabilumas
lt.ktu.tt.server	0	0,679
lt.ktu.tt.server.data	0,231	0,405
lt.ktu.tt.server.info	0,143	0,119
lt.ktu.tt.server.service	0,429	0,111
lt.ktu.tt.server.service.admin	0,59	0,029
lt.ktu.tt.server.util	0,6	0



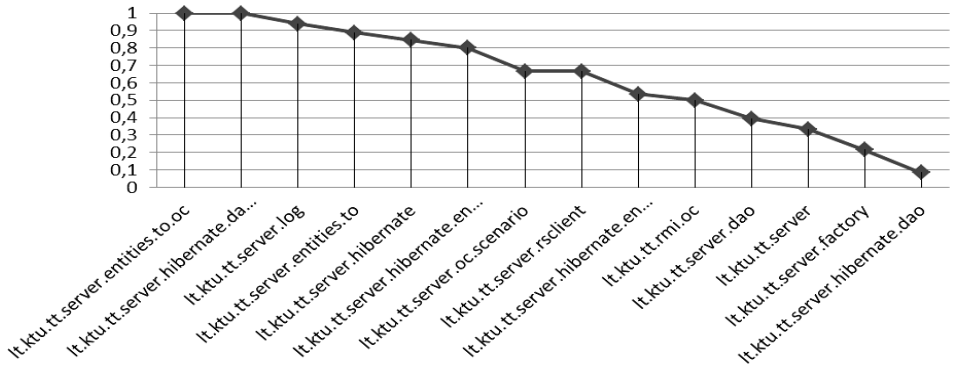
50 pav. Priklausomybė TestTool 5.2.2 versijoje tarp abstraktumo ir nestabilumo

Matome, kad dauguma paketų eina link balanso tarp stabilumo ir abstraktumo, kas yra normalu. Ryšis tarp paketų yra silpnesnis nei 5.3 versijoje. Abstraktumas yra geresnis nei 5.3 versijoje, kas sąlygoja apie didesnes paketų pakartotino pqanaudojimo galimybes. 5.3 versijoje mažiau subalansuotų paketų, priklausomybė tarp jų yra didesnė. Paketų struktūra abstraktumo ir nestabilumo metrikų atžvilgiu yra geresne senesnėje TestTool versijoje.

- Normalizuota distancija (angl. Normalized Distance)

Distancija iki pagrindinės sekos arba normalizuotos distancijos metrika parodo kaip toli paketas yra nuo idealumo, reiškia statmenišką distanciją nuo paketo iki pagrindinės sekos. Ji vadinama distancija (D) ir matuojama pagal supaprastintą formulę |(A+I-1)| (kartais yra taikoma ir pilna formulė |(A+I-1) ÷ 2|, kuri kinta nuo [0, ~0,707]), ši metrika kinta tarp [0,1]. Bet koks paketas, kurio D yra nenulis, gali būti restruktūrizuotas arba peržiūrėtas. Tokia metrika gali padėti išskirti paketus, kurie yra labiausiai pakartotinai panaudojami (angl. reuse) ir yra mažiau jautresni pakeitimams.

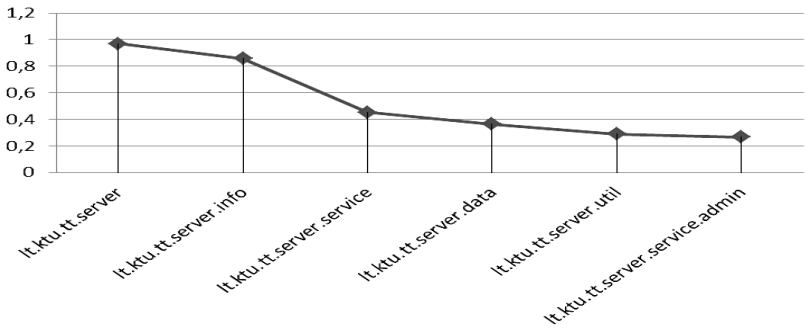
Tiriamo objekto paketų distancija yra parodyta *51* pav.



51 pav. Normalizuota distancija tarp paketų

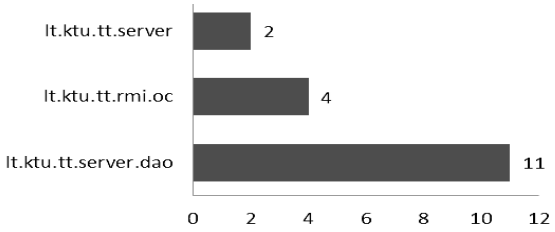
Distancija taip pat parodo duoto paketo inkapsuliaciją, kol distancija yra 0, tai reiškią idealią inkapsuliaciją, tuo tarpu 1 reiškia, kad inkapsuliacijos nėra.

5.2.2 versijoje situacija pavaizduota *52* pav.



52 pav. Normalizuota distancija tarp paketų 5.2.2 versijoje

- Sąsajų kiekis (angl. Number of Interfaces, trump. NOI)



53 pav. Sąsajų kiekis paketuose.

53 pav. pavaizduoti paketai turintys bent vieną sąsają, kiti paketai sąsajų neturi. Šiame projekte visos sąsajos sudėtos į atskirus paketus.

- McCabe ciklomatinis sudėtingumas (angl. McCabe Cyclomatic Complexity)

Parodo nepriklausomų kelių programos kodo grafe skaičių.

Ciklomatinio sudėtingumo reikšmės vertinimas parodytas ***16*** lentelėje.

16 lentelė. Ciklomatinio sudėtingumo reikšmės vertinimas

CC reikšmė	Programos įvertinimas
1 - 10	paprasta, klaidų tikimybė maža
11 - 20	vidurinio sudėtingumo, klaidos tikimybė vidutinė
21 - 50	sudėtinga, didelė klaidos tikimybė
> 50	netestuojama

Metrics įrankis rekomenduoja sudėtingumą laikyti nedidesnį už 10.

10 metodų su didžiausiu sudėtingumu parodyti *17* lentelėje.

17 lentelė. Ciklomatiniai metodų sudėtingumai

Metodas	Reikšmė
createOrUpdateExam	20
createOrUpdateTest	17
createOrUpdateQuestion	15
searchAdministratorBy	13
removeQuestionP	12
getRSRemoteObject	12
removeQuestionWeight	11
removeAdminFromRStorage	10
searchResourceStorageBy	9
getVariant	9

Egzamino sukūrimo metodas yra pakankamai sudėtingas, į tai gali įtakoti ankščiau ištirtas egzamino duomenų klasė, kurioje buvo rasta netikslumų.

Vidutinio sudėtingumo yra naujo duomenų objekto įkėlimo į duomenų bazę bei paieškos klasės metodai. Keičiant šiuos metodus reikėtų atkreipti daugiau dėmesio, nes klaidos tikimybė yra vidutinė. Išnagrinėjus visus metodus netestuojamo kodo (pagal 16 lentele) nėra.

5.2.2 versijos ciklomatinis sudėtingumas parodytas *18* lentelėje.

18 lentelė. Ciklomatiniai 5.2.2 versijos metodų sudėtingumai

Metodas	Reikšmė
checkOctet	20

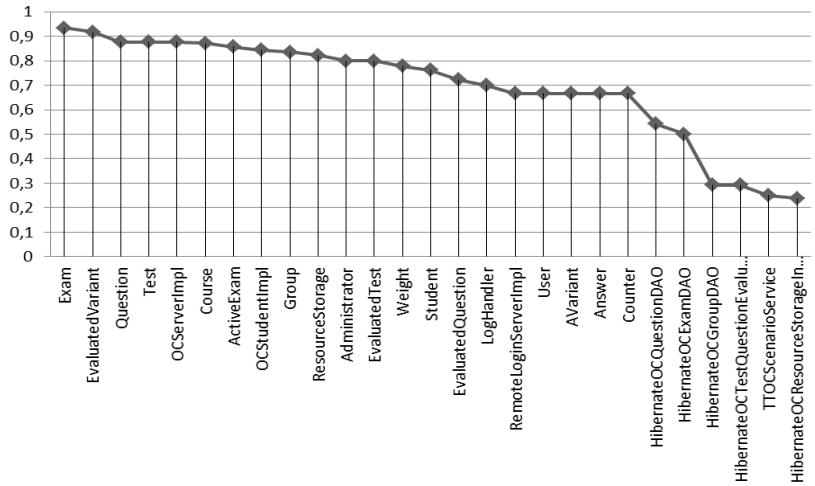
checkExamAccess	15
evaluateVariant	15
getCourses	15
parseOptions	14
setOwner	11
loginUser	11
getCorrectAnswer	11
getVariant	11
startExam	11

Matome, kad ciklomatinio sudėtingumo maksimalios reikšmės abejose versijose yra vienodos.

- Metodų rišlumo stoka (angl. Lack of Cohesion, LCOM)

Metrika, kuri parodo rišlumą klasėje. Skaičiuojama pagal Henderson-Seller metodą[27]. Jei m(A) yra metodų skaičius kurie naudoja atributą A, skaičiuojamas vidurkis m(A) visiems atributams, atimamas skaičius metodų m ir gautas rezultatas padalinamas iš (1-m). Maža reikšmė parodo rišlią klasę, o reikšmė artimesnė 1 parodo rišlumo stoką, tokiu atveju yra rekomenduojama dalinti klasę į sudėtines klases.

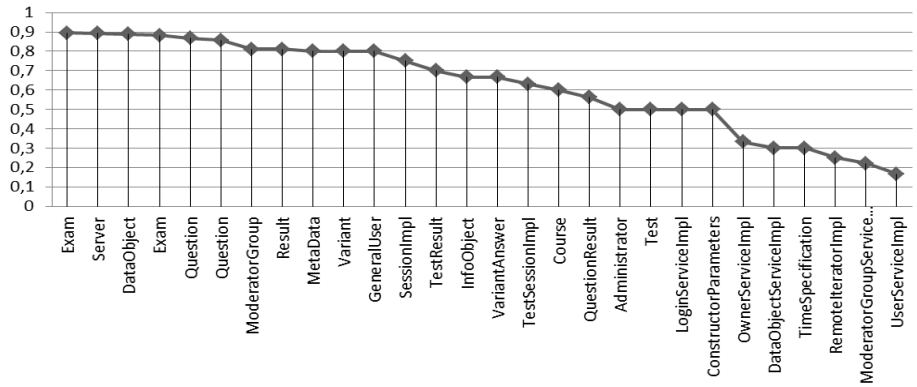
54 paveiksle parodytos klasės, kurios turi rišlumo stoką, kitu klasių LCOM yra nulis, kas reiškia, kad jie neturi rišlumo stokos.



54 pav. Klasės, kurios turi didžiausia rišlumo stoką

Žemas rišlumas sako apie didėjanti sudėtingumą, per kurį kūrimo metu gali būti padaryta nemažai klaidų.

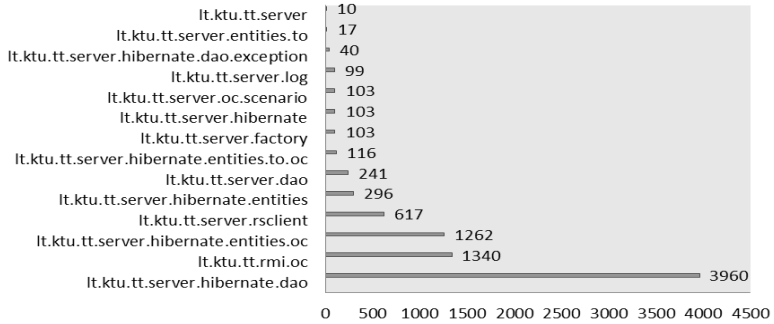
5.2.2 versijos šios charakteristikos reikšmės parodytos *55* pav.



55 pav. 5.2.2 versijos klasės, kurios turi didžiausią rišlumo stoką

- Bendras kodo eilučių kiekis (angl. Total Lines of Code, trump. TLOC)

Bendras kodo eilučių kiekis pagal paketus parodytas *56* pav.



56 pav. Bendras kodo eilučių kiekis pagal paketus

Ši metrika neturi kritinių ribų, bet didesnė jos reikšmė kalba apie modulio apimtį.

- Parametrų kiekis (angl. Number of Parameters)

Didesnis parametrų kiekis didina sudėtingumą, Metrics įskiepyje yra rekomenduojama, kad metodo parametrų kiekis neviršytų 5. Metodai su didžiausiu parametrų kiekiu pateikti *19* lentelėje.

19 lentelė. Metodai su didžiausiu kiekiu parametrų

Metodas	Parametrų kiekis
storeEvaluationResult	*7*
searchAdministratorBy	6
createOrUpdateQuestion	5
createOrUpdateQuestion	5
rollbackTransaction	5
evaluateAndSave	5
evaluateWithoutSave	5
evaluate	5
prepareByItem	4
searchAdministrator	4

Metoduose, kuriuose rekomenduojama norma yra viršyta patartiną parametrų kiekį, rekomenduojama modifikuoti taip, kad būtų paduodamas objektas, vietoje atskirų atributų.

5.2.2 versijos metodai su didžiausiu parametrų kiekiu yra parodyti *20* lentelėje

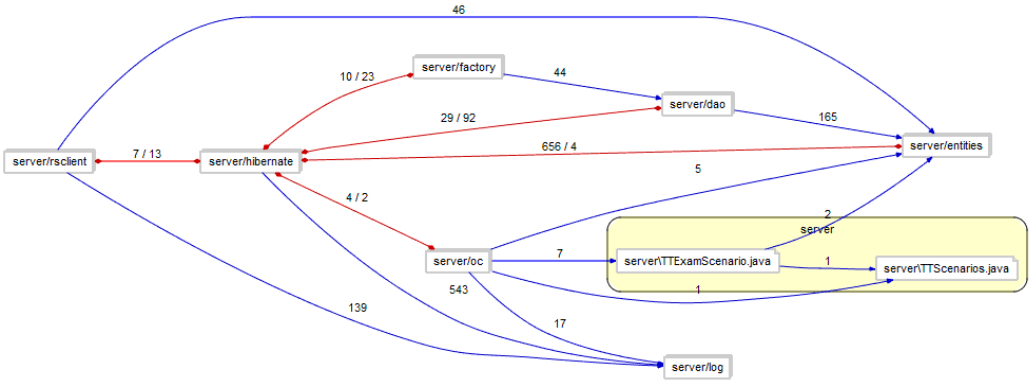
20 lentelė. TT5.2 versijos metodai su didžiausiu parametrų kiekiu

Metodas	Parametrų kiekis
Exam	*11*
User	*7*
Exam	*6*
Moderator	*6*
Variant	*6*
Administrator	5
Question	5
Test	5
MetaData	5
createModerator	5

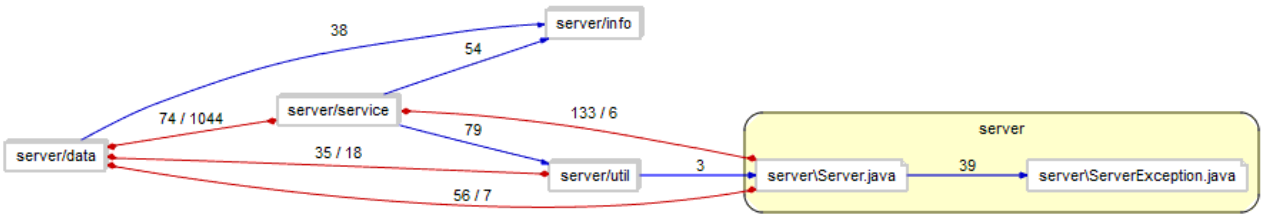
Rekomenduojamas kiekis parametrų yra viršytas didesniam kiekiui metodų nei 5.3 versijoje. Neskaitant to, matome, metodai turi vienodus vardus, bet randasi skirtingose klasėse ir paketuose, (du Exam metodai tuo pačiu pavadinimu), tai gali papildomai suklaidint kūrėją, bei padidinti klaidos tikimybę, testavimas irgi sudėtingėja.

- Vidinių priklausomybių grafas paketams

Priklausomybes tarp paketų taip pat galima parodyti brėžiant vidinių priklausomybių grafą. Virš ryšių nurodytas krepinių (angl. reference) kiekis į priklausomą paketą. Toks grafas parodytas 57 ir 5.2.2 versijai - 58 pav.



57 pav. Vidinių priklausomybių tarp paketų grafas



58 pav. 5.2 versijos vidinių priklausomybių grafas

Matome, kad didesnė priklausomybė tarp paketų yra 5.3 versijoje (57 pav.), nei senesnėje versijoje (58 pav). Tai patvirtina ir ankščiau pateiktų metrikų parodymai.

- Statinių metodų kiekis (angl. Number of Static Methods)

lt.ktu.tt.rmi.oc, lt.ktu.tt.server.factory, lt.ktu.tt.server.hibernate, lt.ktu.tt.server.log, lt.ktu.tt.server.oc.scenario turi po vieną statinį metodą (viso 5), kiti paketai statinių metodų neturi, pagal atitinkamas klases paskirstymas parodytas *21* lentelėje.

21 lentelė. Statinių metodų kiekis klasėse

Klasė	Metodų kiekis
RemoteLoginServerImpl	1
DAOFactory	1
HibernateUtil	1
LogHandler	1
TTOCScenarioService	1

5.2.2 versija turi didesnį kiekį statinių metodų - *22* lentelė**.** Didesnis skaičius statinių metodų rodo prastesnę architektūrą.

22 lentelė. 5.2 versijos statinių metodų kiekis klasėse

Klasė	Metodų kiekis
IpUtil	5
RemoteIteratorAdvancedImpl	2
DataUtil	2
DataFields	1
Server	1

- Statinių atributų kiekis (angl. Number of Static Attributes)

Statinių atributų kiekis parodytas 23 lentelėje. Rekomenduojama, kad klasėje būtų nedaugiau 5.

23 lentelė. Statinių atributų kiekis klasėse

Klasė	Atributų kiekis
LogHandler	4
RemoteLoginServerImpl	3
OCServerImpl	2
TTOCScenarioService	2
OCResourceStorageClient	2
OCStudentImpl	1
DAOFactory	1
HibernateOCActiveExamDAO	1
HibernateOCAdministratorDAO	1
HibernateOCCourseDAO	1
HibernateOCExamDAO	1
HibernateOCGroupDAO	1
HibernateOCQuestionDAO	1

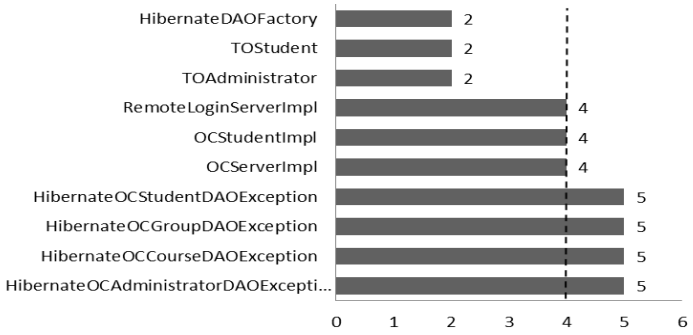
HibernateOCQuestionWeightDAO	1
HibernateOCResourceStorageInformationDAO	1
HibernateOCStudentDAO	1
HibernateOCTestDAO	1
HibernateOCTestQuestionEvaluResDAO	1
HibernateUtil	1
Viso:	27

Ši metrika mūsų projekte yra neviršyta. Didesnis skaičius atributų rodytų prastą klasių architektūrą, ypač jei tai rištųsi su aukštu metodų kiekiu klasės. Klasės, kurios neturi statinių atributų yra specialieji atvejai, kurios nebūtinai yra anomališkos. 5.2.2 versijoje yra net 44 statiniai atributai, ir net 8 statiniai atributai Server klasėje, tai gali sąlygoti apie blogesnę senesnės versijos architektūrą.

- Paveldėjimo medžio gylis (angl. Depth of Inheritance Tree, trump. DIT)

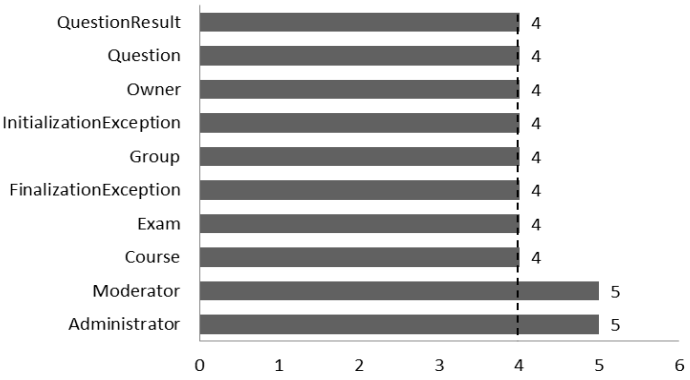
Kuom gilesnė klasė hierarchijoj, tuo didesnis skaičius metodų yra paveldėtas, tuo darosi sudėtingiau atspėti jų elgseną. Labiau gilesni medžiai seka prie didesnio projekto sudėtingumo, kadangi dalyvauja daugiau klasių ir metodų. Iš kitos pusės kuo giliau tam tikra klasė yra hierarchijoje, tuo didesnis potencialus tų pačių metodų pakartotinas panaudojimas. Rekomenduojama reikšmė yra nuo 0 iki 4.

Klasės su didžiausia šios metrikos reikšme yra parodytos 59 pav.



59 pav. Klasės su didžiausiu paveldėjimo medžio gyliu

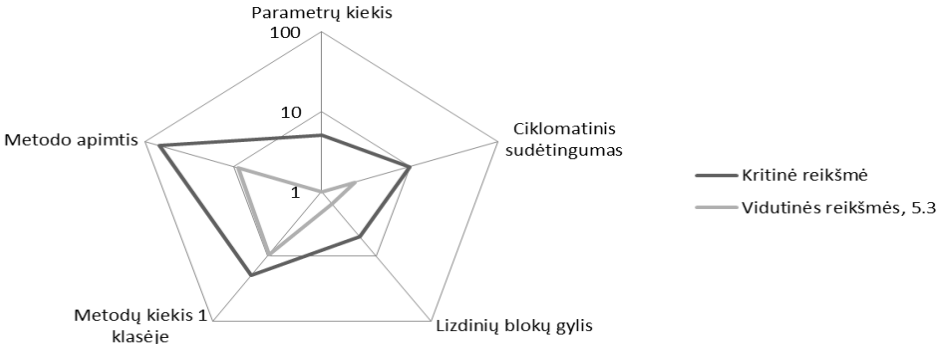
Kaip matome keliose klasėse rekomenduojama reikšmė yra nežymiai viršyta, tai gali vesti prie didesnio sudėtingumo ir sukompromituoti inkapsuliaciją, pagal metodų pavadinimus matome, kad tai išimčių klasės, priešingai nei 5.2.2 versijoje. Paveldėjimo medžio gylio metrikos reikšmės 5.2.2 versijai parodyti *60* pav.



60 pav. 5.2.2 versijos klasės su didžiausiu paveldėjimo medžio gyliu

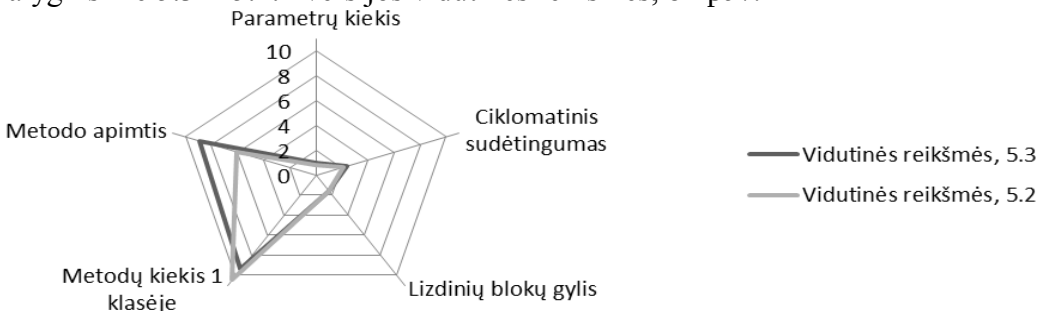
- Vidutinių kiekio metrikų reikšmių palyginimas su ribinėmis.

61 paveiksle yra pateiktos projekto tyrimo metu išmatuotų metrikų reikšmių „Radar“ diagrama log pagrindu 10 skalėje. Kaip matome, visumoje kritinės reikšmės yra neviršytos. Iš čia galima padaryt išvadą, jog bendras šio produkto prižiūrimumo lygis yra vidutinis arba net gi geras, kadangi padidėjęs prižiūrimumas yra ne daugiau nei 5% objektų visame modulyje. Tačiau rekomenduojama atsižvelgti į tyrimo metu rastus netikslumus, plačiau apie juos šios ataskaitos 6.3 skyriuje.



61 pav. Vidutinių reikšmių metrikų palyginimas su kritinėmis

Palyginsime 5.3 ir 5.2.2 versijos vidutines reikšmes, *62* pav.



62 pav. Vidutinių 5.3 ir 5.2.2 versijų reikšmių palyginimas

Matome, kad abi versijos, liginant jų vidutines reikšmes, pakankamai panašios. Maksimalios kiekybinių metrikų palyginimas pateiktas 22 pav. aukščiau.

6.3. SKYRIAUS IŠVADOS

Atlikus tyrimą buvo nustatyta eilė netikslumų ir galimų patobulinimų. Žemiau pateikti tyrimo rezultatai ir rekomendacijos.

- Išeities kodas yra nepakankamai komentuotas, gerai sukomentuotos pagrindinės sąsajos ir kai kurie pagrindiniai metodai. Rekomenduojama sukomentuoti visas sąsajas bei metodus.
- Lizdinių blokų gylio metrika yra viršyta (reikšmė yra 8) createOrUpdate metode. McCabe ciklomatinis sudėtingumas taip pat viršytas createOrUpdateExam (reikšmė yra 20), createOrUpdateTest (reikšmė yra 17), createOrUpdateQuestion (reikšmė yra 15), removeQuestionP (reikšmė yra 12), searchAdministratorBy (reikšmė yra 13), removeQuestionWeight (reikšmė yra 11), getRSRemoteObject (reikšmė yra 12) klasėse patartina naudoti išskleidimo funkciją tam, kad padalinti šiuos metodus į dalis.
- Parametrų kiekis metoduose yra viršytas storeEvaluationResult (reikšmė yra 7), rekomenduojama paduoti duomenų objektą kaip parametrą.
- Kai kurios klasės turi žemą rišlumo stoką, rekomenduojama jas pertvarkyti, tam, kad sumažinti sudėtingumą.
- Norint sumažinti klaidos tikimybę iki mažos rekomenduojama restruktūrizuoti šias klases: createOrUpdateExam, createOrUpdateTest, createOrUpdateQuestion, searchAdministratorBy, removeQuestionP, searchAdministratorBy, createOrUpdateQuestion, createOrUpdateTest.
- Rekomenduojama didinti balansą tarp abstraktumo ir nestabilumo paketuose.
- Metodų kiekio metrika yra viršyta ne daugiau 10% sistemos klasėms, kas yra normos ribose.
- Exams klasė neturi ryšių ir yra tuščią, rekomenduojama ją panaikinti iš sistemos.
- Rekomenduojama peržvelgti Exam klasės struktūrą, gali būti, kad ją reikią išskaidyti į kelias sudėtines klases.
- Klasių paketai yra padalinti taip, kad gali būti lengvai papildyti, jei sistemos funkcionalumas augtų. Iš kitos pusės dauguma paketų yra glaudžiai surišti, dėl to keičiant šių paketų pavadinimus arba struktūrą, galimų pakeitimų įtaka turėtų būti įvertinta, kad pakeitimai nesutrikdytų korektiško darbo.
- Bendras produkto prižiūrimumo lygis yra vidutinis arba net gi geras atsižvelgiant į vidutines ir kritines reikšmes. Tačiau rekomenduojama atsižvelgti į tyrimo metu rastus pavienius netikslumus.

- Nepaisant smulkių trūkumų ir netikslumų nustatytų išeities kodo metrikų analizės metu realizuota posistemė yra tinkama integravimui į galutinę TestTool sistemą.

Palyginimo išvados:

- TestTool 5.2.2 kritinės ribos yra nežymiai viršytos: lizdinių blokų gylio, McCabe ciklomatinio sudėtingumo ir parametrų kiekio metrikose. Šių metrikų reikšmės yra viršytos ir tiriamoje paskirstytoje sistemoje.
- Kadangi 5.3 versijoje funkcionalumas yra padidėjęs beveik dvigubai, palyginus su ankstesnėmis versijomis, sistema tapo pasiskirstyta, o sudėtingumas yra panašus kaip senesnės versijos, iš to galima padaryti išvadas, kad 5.3 sistemos architektūra yra pakankamai gera, kadangi pavykdo išvengti augančio sistemos sudėtingėjimo, didinant funkcionalumą.
- Prižiūrimumas yra žemesnis nei senesnėje versijoje, pakeitimai 5.3 versijoje yra lengvesni klasėse, bet ne pakėtuose, paketų struktūra yra pakankamai griežta.
- Laikantis naudojamų architektūrinių šablonų logikos. klasių ir metodų praplėtimas nesukels ypatingų sunkumų programuotojams.
- 5.3 posistemės, priešingai nei ankstesnės versijos, turi naudotojo ir programuotojos dokumentaciją.

7. IŠVADOS IR REKOMENDACIJOS

1. Kokybišką sistemą galima sukurti tik tada kai taikomas sklandus ir suderintas kūrimo procesas.
2. Naujai realizuotos posistemės turi pakankamai lanksčią pakeitimams architektūrą, tokiu būdu ji turėtu palengvinti busimus sistemos patobulinimus – naujų funkcijų realizaciją, užtikrintas portatyvumas. Posistemės gali naudoti skirtingų gamintojų duomenų bazes.
3. Žinių testavimo scenarijai suteikia lankstumo testavimo procesui, kadangi galima realizuoti skirtingus vertinimo ir klausimų variantų pateikimo scenarijus, bei juos pritaikyti konkrečiam atvejui. Mokymosi resursų paruošimo scenarijai leidžia apjungti įvairius variantų arba kitų resursų paruošimą vienoje sistemoje, o nenaudoti papildomų įrankių, kurie dažnai būna prastos kokybės.
4. Testavimo metu visos aptiktos surastos klaidos ir netikslumai yra pašalintos. Operacijų vykdymo greitis yra priimtinas. Realizuota PĮ veikia stabiliai ir nedaro neigiamos įtakos kitai programinei įrangai, esančiai toje pačioje aplinkoje, kokybės požiūriu sistema yra efektyvi. Testuojamos funkcijos veikia korektiškai. Testuojamumas yra geras, sistema yra pakankamai patikima. Nagrinėjama PĮ atitinka vartotojų keltus reikalavimus ir funkcionalumą, dauguma numatytų panaudojimo atvejų yra realizuota.
5. Kokybės tyrimo metu, remiantis išeities kodų metrikomis, buvo nustatyti vienetiniai atvejai kai buvo viršytas lizdinių blokų gylis, klasės metodų parametrų kiekis, atributų klasėje kiekio metrikos, tai veda prie didesnio posistemių sudėtingumo. Bendrai vidutinės metrikų reikšmės neviršijo kritinių rodyklių. Formalios techninės peržiūros metu buvo nustatyta, kad kai kurios funkcijos kode yra nepakankamai dokumentuotos.
6. 5.3 versija palyginus su ankstesnę 5.2 yra sudėtingesnė bei reikalauja daugiau priežiūros pradiniame naudojime, iš kitos pusės atsižvelgiant į senesnės versijos vartotojų nusiskundimus ir sukurtos PĮ testavimo rezultatus, šioje versijoje pavyko pasiekti teigiamų rezultatų veikimo stabilume, funkcionalume, naudojimo patogume. Žymiai pasikeitė posistemių apimtis, bet sudėtingumas augo nežymiai, tai reiškia, kad paskirstytos sistemos su žinių testavimo scenarijais kokybė yra pakankamai aukštame lygyje.
7. Posistemės yra tinkamos integravimui į galutinę TestTool sistemą.
8. Rekomenduojama po sistemų integracijos atlikti visos sistemos testavimą.

8. TERMINŲ IR SUTRUMPINIMŲ ŽODYNAS

A trump. nuo angl. Abstractness, žr. **Abstraktumas**

Abstrakcija galimybė programuoti nežinant konkrečių detalių apie informaciją.

Abstraktumas metrika, kuri išsiskaičiuoja iš Abstrakčių klausių kiekių (ir sąsajų) padalintų iš visų tipų kiekio pakete.

Aferentinis susietumas metrika kitaip dar vadinimas įcentriniu susietumu, nurodanti klasių kiekį paketo išorėje, kurie priklauso nuo klasių paketo viduje.

Aktoriai sistemos naudotojai.

Balta dėžė Testavimo strategija (angl. white box, trump. wb), kurios metu kuriant testavimo atvejus yra nagrinėjamas programos kodas.

Bendras eilučių kiekis – metrika, kuri reiškia netuščių ir ne komentarų eilučių kiekį.

Ca trump. nuo angl. Afferent Coupling, žr. Aferentinis susietumas

Ce trump. nuo angl. Efferent Coupling, žr. Eferentinis susietumas

CSV - trump. Duomenų saugojimo formatas.

DAO trump. nuo angl. Data Access Objects, duomenų šaltinio abstrakčios sąsajos.

Demo paprastai reiškia bandomąją programinės įrangos versiją, gali būti apribotas funkcionalumas arba veikimo laikas.

DIT trump. nuo angl. depth of inheritance tree, žr. Paveldėjimo medžio gylis

Eferentinis susietumas metrika kitaip dar vadinamas išcentriniu susietumu, nurodanti kiekį klasių paketo viduje, kurie priklauso nuo klausių paketo išorėje.

Eclipse atviro kodo integruota kūrimo aplinka

Egzaminas Terminas naudojamas TestTool sistemoje, žymį egzaminą, egzaminui priskiriamas testas. Egzaminas negali egzistuoti be testo (žr. testo aprašymą). Egzaminas gali būti priskirtas vienai arba kelioms grupėm (žr. grupės aprašymą).

Grupė Terminas naudojamas TestTool sistemoje, žymi studentų grupę, grupei gali būti skirtas egzaminas, tokiu būdu kiekvienas duotos grupės studentas galės vykdyti priskirta duotai grupei egzaminą.

GUI trump. nuo Grafinė vartotojo sąsaja

Hibernate biblioteka, kuri leidžia atvaizduoti objektines klases reliacinėse duomenų bazėse.

I trump. nuo angl. Instability, žr. **Nestabilumas**

ID trump. nuo Unikalus Identifikatorius, pagal kuri atskiriamos esybės duomenų bazėje.

IDE trump. nuo angl. Integrated Development Environment, integruota kūrimo aplinka, pvz. Eclipse IDE

Java objektinė programavimo kalba**.**

JDK trump. nuo angl. Java Development Kit, Java programavimo aplinka.

Juoda dėžė Testavimo strategija (angl. black box, trump. bb), naudojant vartotojo reikalavimus arba specifikaciją.

JVM trump. Java Virtuali Mašina

Klasių kiekis angl. number of classes, bendras klasių kiekis pasirinktoje srityje.

Klausimas Terminas naudojamas TestTool sistemoje, žymi klausimą, kuriam dažniausiai priskiriamas pavadinimas ir variantų failas.

Kursas Terminas naudojamas TestTool sistemoje, kuris žymį mokymosi dalyką, savo ruožtu apjungia grupes, studentus, egzaminus.

Laukų kiekis metrika parodanti bendrų laukų kiekį pasirinktoje srityje

LOC trump., žr. bendras eilučių kiekis

McCabe ciklomatinis sudėtingumas angl. McCabe Cyclomatic Complexity, metrika nurodanti nepriklausomų kelių programos kodo grafe skaičių.

Metodo eilučių kiekis metrika, kuri reiškia eilučių kiekį, kurias užima vienas metodas klasėje

Metodų skaičius metrika nurodanti – bendras metodų skaičių pasirinktoje srityje.

MLOC trump. nuo angl. method lines, žr. Metodo eilučių kiekis

Mokymosi resursai suprantami kaip TestTool variantai, klausimai, atsakymai, įvairios mokymosi bylos, mokymosi medžiaga ir t. t.

Nestabilumas Kitaip dar kaip nepastovumas, ši metrika priklauso nuo aferentinio ir eferentinio susietumo.

NOC trump. nuo angl. number of childrens, žr. Vaikų kiekis

NOF trump. nuo angl. number of fields, žr. Laukų kiekis

NOI trump. nuo angl. number of interfaces, žr. Sąsajų kiekis

NOM trump. nuo angl. number of methods, žr. Metodų skaičius

NORM trump. nuo angl. number of overridden methods, žr. Perrašytų metodų kiekis

Objektas Duomenų ir susijusio funkcionalumo talpinimas į vientisus vienetus; objektai padeda pasiekti moduliškumą ir nusako objektinės programos struktūrą

OC trump. Operacijų centras, sukurtos programinės įrangos dalis, nepriklausoma testavimo sistemos serverio komponentas atliekantis testavimo sistemos administravimo posistemės funkcijas.

OS trump. Operacinė sistema

Pasvertų metodų skaičius klasėje metrika nurodanti sumą visų McCabe ciklomatinių metodų sudėtingumų klasėje.

Paveldėjimas Objektų organizavimas specializuojant egzistuojančius bendresnius tipus, papildant ar iš dalies pakeičiant funkcionalumą

Paveldėjimo medžio gylis metrika nurodanti atstumą nuo klasės objekto paveldėjimo hierarchijoje.

Perrašytų metodų kiekis Bendras kiekis metodų pasirinktoje srityje, kurie yra perrašomi nuo protėvio klasės.

Produktas suprantamas kaip programinė įranga sukurta projekto metu. Į produktą įeina OC ir RS posistemės bei visi projekto metu sukurti programiniai komponentai. Į produktą taip pat įeina visa projekto metu su juos susijusi dokumentacija.

Profaileris Programa, kuri gali stebėti kitos programos darbo charakteristikas, tokias kaip atskirų fragmentų vykdymo laikas (dažniausiai paprogramių), operatyvios atminties išnaudojimas ir pan.

RMI trump. nuo angl. Remote Method Invocation, standartų sistema koordinuotam Java kalba parašytų programų darbui internete.

RS trump. nuo Resursų Saugykla, TestTool 5.3 serverio posistemė, angl. Resource Repository, trump. RR. Resursų saugykla, sukurtos programinės įrangos dalis, nepriklausomas serverio komponentas atliekantis testavimo sistemoje testavimo ir vertinimo (įvertinimų) medžiagos saugojimą ir apdorojimą.

RVS Rezultatų ir vertinumo saugykla, posistemės „Resurso saugykla“ dalis.

Sąsajų kiekis metrika nurodanti sąsajų kiekį pasirinktoje srityje.

SQL trump. nuo Struktūrizuota užklausų kalba

SSL trump. nuo angl. Secure Sockets Layer, saugaus duomenų perdavimo protokolas

Studentas Terminas naudojamas TestTool sistemoje, žymį asmenį, kuris atliks testus (egzaminus), spręs pateiktus klausimus.

Testas Terminas naudojamas TestTool sistemoje, žymi klausimų grupę, gali būti priskirtas prie egzamino ir tada egzamino sudėtyje įvykdytas studentu.

TestTool testavimo sistema. trump. TT.

TLOC trump. nuo angl. Total Lines Of Code, žr. Bendras eilučių kiekis

TT trump. nuo TestTool, pilnavertė žinių testavimo sistema.

TT trump. nuo TestTool.

Vaikų kiekis metrika parodanti kiekį tiesioginių klasės sub-klasių. Klasė, kuri realizuoja sąsają yra skaičiuojama, kaip tiesioginis šios sąsajos vaikas.

Variantas arba variantų failas Terminas naudojamas TestTool sistemoje, dažniausiai žymi variantų failą, kuriami saugomi vieno klausimo variantai. 5,3 versijoje saugomas Resursų saugyklos serverio posistemėje.

Vienetas Terminas naudojamas vienetų testavime, reiškia testuojamą vienetą paprastai reiškią klasę, metodą, procedūrą arba funkciją.

WMC trump. nuo angl. Weighted Methods per Class, žr. Pasvertų metodų skaičius klasėje

XML trump. Išplečiama žymų kalba (angl. Extensible markup language)

9. LITERATŪROS SĄRAŠAS

[1] A. Yas Alsultanny, M. Ahmed Wohaish, Requirements of Software Quality Assurance Model, Second International Conference on Environmental and Computer Science, 2009, [žiūrėta 2011.05.06]

[2] R. Biswas, E. Ort, "Java Persistence API - A Simpler Programming Model for Entity Persistence," http://java.sun.com/developer/technicalArticles/J2EE/jpa/, 2009 , [žiūrėta 2011.05.06]

[3] P. Singh Sandhu,., PPal Singh; , A. Kumar Verma, Evaluating Quality of Software Systems by Design Patterns Detection, Advanced Computer Theory and Engineering, 2008. ICACTE '08. International Conference, 20-22 Dec. 2008 , [žiūrėta 2011.05.06]

[4] F. Khomh. Y.-G Gueheneuc , Univ. of Montreal, Montreal „Do Design Patterns Impact Software Quality Positively?“, Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference, 2008.04.1-4, [žiūrėta 2011.05.06]

[5] Pagrindinių dalykų kūrimas ir mokymas Webct Vista 4. KTU IF ITDC 2009 http://webct.liedm.lt/index_files/webct_blackboard_vista_vadovas.pdf [žiūrėta 2009-11-30]

[6] About Moodle. http://docs.moodle.org/en/About_Moodle [žiūrėta 2009-11-30]

[7] User Managemenet. Other. http://docs.moodle.org/en/Administrator_documentation#User_Management [žiūrėta 2009-11-30]

[8] Testuok.lt sistema. http://testuok.lt/index.htm [žiūrėta 2009-11-30]

[9] S. Liu, P. Chen, „Developing Java EE Applications Based on Utilizing Design Patterns, 2009 WASE International Conference on Information Engineering, 398-401, [žiūrėta 2009-10-10]

[10] G. Booch, A. M. Rober ir W. E. Michael, et al. Object-oriented analysis and design with applications, Peking, Posts & Telecom Press, 3rd ed, 2008.4, 1-20. [žiūrėta 2009-11-11]

[11] E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design patterns Elements of reusable object-oriented software, Peking: China Machine Press, 2004.9, 1-29. . [žiūrėta 2009-10-10]

[12] D. Alur, J. Crupi, D. Malks, Core J2EE Patterns: Best Practices and Design Strategies SECOND EDITION, Prentice Hall / Sun Microsystems Press, ISBN 0-13-142246-4, 2003, 464-497 [žiūrėta 2009-10-10]

[13] D. Alur, J. Crupi, D. Malks, Core J2EE Patterns: Best Practices and Design Strategies, SBN:0130648841; 1st edition (June 26, 2001), http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html [žiūrėta 2009-10-10]

[14] C. Bauer, G. King, Java Persistence with Hibernate, Manning Publications Co, ISBN 1-932394-88-5, USA, 2007, 4-36, [žiūrėta 2009-10-10]

[15] M. Xue, C. Zhu, Design and Implementation of the Hibernate Persistence Layer Data Report System Based on J2EE, Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference on, 2009.05.16-17, 232-235 [žiūrėta 2009-10-10]

[16] D. Mitter, J. Linwood, Beginning Hibernate: from novice to Professional, Apress Inc., ISBN-13 (pbk): 978-1-59059-693-7, ISBN-10 (pbk): 1-59059-693-5, 2-16, [žiūrėta 2009-10-10]

[17] Java Remote Method Invocation - distributed computing for Java, Technical report, http://java.sun.com/products/jdk/rmi/reference/whitepapers/javarmi.html, [žiūrėta 2009-10-10]

[18] RMI Wire Protocol, http://java.sun.com/javase/6/docs/platform/rmi/spec/rmi-protocol.html [žiūrėta 2009.02.16]

[19] Log4j dokumentacija - http://logging.apache.org/log4j/1.2/faq.html

[20] J. Tian „Software Quality Engineering Testing Quality Assurance and Quantifiable Improvment“, 2005, USA, ISBN 0-471-71345-7 [žiūrėta 2011.04.26]

[21] K. Cem; F. Jack and Nguyen, H/ Quoc . Testing Computer Software, 2nd Ed.. New York, et al: John Wiley and Sons, Inc.. pp. 480 psl., 1999. ISBN 0-471-35846-0. [žiūrėta 2011.04.26]

[22] J. B. Rainsberger, JUnit recipes: practical methods for programmer testing, Manning Publications, 2004.04.15, ISBN-10: 1932394230, [žiūrėta 2009-10-10] [17] SQL Dialects http://docs.jboss.org/hibernate/core/3.3/reference/en/html/session-configuration.html *[žiūrėta 2010.04.15]*

[23] Magistrinio projekto informacinė sistema - http://magistras.ruslanas.eu/

[24] TestTool sistemos adresas internete – http://testtool.ktu.lt

[25] R. Martin, „OO Design Metrics“, 1994-10-28 [žiūrėta 2011.03.23]

[26] T. Honglei, S. Wei, Z. Yanan, The Research on Software Metrics and Software Complexity Metrics, International Forum on Computer Science-Technology and Applications, College of Information Technology, Beijing Normal University, Zhuhai 519086, 2009[žiūrėta 2011.03.23]

[27] Brian-Henderson-Sellers „Object-Oriented Metrics, measures of Complexity“,Prentice Hall, 1996 [žiūrėta 2011.04.25]

[28] Martin, Robert C. “Agile Software Development Principles, Patterns, and Practices”,2003.Pearson Education Inc., USA, ISBN 0-13-597444-5 [žiūrėta 2011.04.25]

1 PRIEDAS. SUKURTŲ POSISTEMIŲ LICENCIJA

Žemiau pateikta licencija yra taikytina visom projekto metu sukurtom posistemėms bei su jomis susijusiai dokumentacijai.

LICENCIJA

Įsigyjant arba kitaip gaunant produktą naudotojas negauna visų teisių į programą bei netampa jos savininku, jam suteikiamos tik tos teisės, kurios yra aprašytos žemiau.

1. Leidimai

Jei produktas naudojamas nekomerciniams tikslams naudotojas gali laikyti ir naudotis juo, įskaitant produkto teikiamomis paslaugomis savo serveryje ar kompiuteryje. Naudotojas gali turėti vieną arba daugiau produkto kopijų.

Leidžiamas realizuoto produkto integravimas į kitas posistemes ir sistemas, jei produkto naudojimas tokiu būdu neprieštaraus licencijoje aprašytų sąlygų.

Jei produktas naudojamas komerciniams tikslams, naudotojas turi laikytis sąlygų, kurios aprašytos pirkimo-pardavimo sutartyje bei jos prieduose.

Jei produktas ar jo dalis yra integruotas į sistemą, projektą ar produktą, kuris naudojamas komerciniams tikslams, tai skaitoma, kad ir šioje sutartyje aptariamas produktas ar jo dalis naudojamas komerciniams tikslams, tokiu būdu tarp produkto kūrėjo ir naudotojo turi būti nustatyti papildomos sąlygos bei pasirašyta pirkimo-pardavimo sutartis, kuri neturi prieštarauti šioje licencijoje aprašytiems sąlygoms.

2. Licencijos galiojimas

Ši licencija yra neterminuota. Licencija įsigalioja kai tik naudotojas pradeda naudoti produktą ar su juo susijusią medžiagą, ar kitus produktus, kurių sudėtyje naudojamas produktas ar jo dalis. Jei naudotojas nesutinka su vienu ar daugiau šioje licencijoje aprašytų punktų, jis neturi teisės naudotis produkto, arba jo dalių. Tokiu atveju naudotojas privalo panaikinti visas produkto kopijas ir dalis (įskaitant rezervines) bei nustoti minėto projekto ar jo dalių naudojimą.

3. Produkto kaina ir įsigijimas

Produkto dalies ar viso naudojimas nekomerciniams tikslams yra nemokamas.

Jei produktas arba jo dalis bus naudojama komerciniams tikslams, produkto kaina priklausys nuo naudotojų kiekio, naudojimo sąlygų. Kiekviename atvejyje kaina turi būti derinama tarp produkto kūrėjo ir pirkėjo, patvirtinant bet kokius komercinius santykius turi būti pasirašoma pirkimo-pardavimo sutartis, kurioje aprašyti naudojimo sąlygos ir nuostatos.

4. Atsakomybės apribojimas

Produkto kūrėjai negarantuoja, kad suteiktas produktas ir su juo susiję paslaugos neturi klaidų, kad rastos klaidos bus ištaisytos ar, kad paslaugas bus apsaugota nuo virusų ar kitų galimai kenksmingų komponentų. Šios sąlygos taikomos ir visai kartu su produktu pateikiamai spausdintinei ar elektroniniu formatu pateiktai medžiagai.

Dokumentuose gali būti skelbiamos nuorodos į interneto tinklapius, už kurių tikslumą bei nurodomų tinklalapių turinį produkto kūrėjai neatsako.

Sistemos produkto naudotojas supranta ir sutinka, kad produktas pateikiamas kaip yra, t. y. bet kokia medžiaga, kuria naudotojas skaito, atsisiunčia, naudoja ar kitaip gauna panaudodamas suteiktą produktą ar produktu suteikta paslaugą yra išimtinai naudotojo nuožiūra ir rizika, ir tik jis atsako už visą žalą, kuri gali būti padarytą jo sveikatai, kompiuterinei įrangai arba kitiems objektams, naudotojas taip pat prisiima visas išlaidas būtinoms aptarnavimo paslaugoms, remontui, pataisoms ar žalai atlyginti.

Produkto autoriai (kūrėjai) neprisiima jokios atsakomybės ir neprisiima atlyginti turtinius ar kitokius nuostolius.

5. Naudotojo (vartotojo) įsipareigojimai

Naudotojas įsipareigoja pateikti kartu su produktu ar tą dalimi kurią jis naudojo ar integravo į kitą projektą ar produktą šią licenciją.

Naudotojas neturi teisės keisti bet kokius šios licencijos punktus be produkto kūrėjo žinios ir raštiškai pasirašytu patvirtinimu.

Naudotojas privalo paisyti pasirašytos sutarties sąlygų, jei tokia buvo pasirašoma arba kitaip patvirtinama, jam gaunant produktą.

6. Palaikymas arba garantinis aptarnavimas

Jei reikalingas palaikymas arba garantinis produkto aptarnavimas, pasirašoma atitinkama sutartis tarp kūrėjo ir tų paslaugų gavėjo. Paslaugos teikiamos pagal sutartyje nustatytas sąlygas.

2 PRIEDAS. „INFORMATION TECHNOLOGIES 2011“ KONFERENCIJOS STRAIPSNIS. (ANGLŲ K.)

DISTRIBUTED ARCHITECTURE FOR CONTEXT MODELING BASED E-LEARNING SYSTEM

Kazys Baniulis, Giedrius Paulikas, Jūratė Pauliutė, Ruslanas Sobolevas, Gytis Vilutis

Kaunas University of Technology, Department of Computer Networks, Studentų 50, Kaunas, Lithuania
kazys.baniulis@ktu.lt, giedrius.paulikas@ktu.lt, jurate.pauliute@ktu.lt, ruslanass@gmail.com, gytis.vilutis@ktu.lt

Abstract. Simulation-based e-learning allows students to use models imitating the real processes and thus accumulate knowledge and experience acquired in the process of active learning. This paper presents the advanced architecture of virtual interactive e-learning system TestTool that is under development in Kaunas University of Technology. TestTool system is implemented as an assessment and graphic modeling engine and is based on architecture of distributed services that is expected to transform to service oriented architecture in the future. The system consists of operation center and resource repository services that are accessible through admin, author and student client programs. In particular, context models are used for construction and assessment of graphic models that are available during learning process. Context models are created using additional design module that can be used as TestTool subsystem. These models describe the subject domain by using contextual graphs and feature diagrams. The employment of context model helps experts to simplify and speed-up the creation of effective graphic models for any given subject and to ensure proper and just-in-time up-/reskilling that can be used in continuous process of education and training.

Keywords: e-learning technology, active learning object, assessment, context modeling, contextual graph, distributed architecture.

1 INTRODUCTION

The quality of e-learning technologies of higher and secondary education, corporate learning and vocational training must be the main factor that governs their expansion [10]. E-learning should be developed for encouragement of widespread learning, cheaper learning and acquisition of the latest and most innovative material. More flexible coherence of the study curricula with student needs and individuality, creation of opportunities to avoid emotional tension and ensuring more objective assessment of all learners is also very important [10]. Thus, good results of studies depend on many factors of learning motivation, where the first premise is the change and dynamism of life. Learning technologies should adequately reflect the content of changes that come from the experience of dynamic life.

In advanced learning technologies the instructional material (content) is composed of Learning Objects (LO). Learning Object can to be seen as a logical container that represents an atomic web-deliverable resource such as Lesson (HTML page), Simulation (Java applet) or Test (HTML page with evaluation form). A high quality instructional material must contain the expository materials, active learning and (self-) assessment parts that constitute the full cycle of experiential learning [13]. The implementation of active learning is the most sophisticated part of e-learning technology design.

This paper discusses how simulation, experimentation and practice as well as assessment and self-control features can be implemented in active learning objects (ALOs). Simulation, practice and experimentation tools are usually created with Java, Flash or similar technologies that have high costs of development. We will show that very effective and fast applications for (self-) assessment in simulation-based active learning can be created for any particular domain by making the information model of the domain that later is used to design and implement the situations of interest.

2 USING CONTEXT MODELING FOR ALO DESIGN

This chapter discusses ALO peculiarities and its implementation that uses context modeling [7][9][17]. The important notions of context and contextual element will be presented. The new ALO information model will also be discussed as it helps to design better ALOs and forms the means for generation of explanations.

2.1 ACTIVE LEARNING OBJECT

Learning and education applications that rely on modern information and communication technologies are oriented to active studies that are implemented by virtual learning environments through various methods of active learning: tests of knowledge assessment, modeling practice, practical tasks, experimental tasks, interactive presentations, etc. The applications of active e-learning are based on experiential learning model of Kolb, where it is stated that "Learning is the process whereby knowledge is created through the transformation of experience" [13]. This model emphasizes the appreciation and usage of already accumulated experience to solve new problems and lends this ability to active learning which allows its user to master new and increasingly complex situations through references to already familiar ones [13]. The educational content is decomposed to separate problems that are solved by learners relying on their experience as well as new learning material. Active e-learning isn't linear or consecutive, i.e. the students are not instructed to absorb the content or solve problems step by step. They rather make decisions that are based on content, because, as noted by Baniulis and Tamulynas expository content presentation and assessment tests are not sufficient for comprehensive evaluation of acquired skills [4][19]. Many researchers (Bauer [2], Baniulis and others [4][19]) found that learning through activity is far more efficient, as modeling practice stimulates the active behavior and constructive learning process. E-learning environments tend to use more modeling practice that focuses students on problem at hand instead of separate fragments of domain knowledge. But lack of feedback in this learning method inhibits motivation and active behavior [15]. To solve this shortcoming Bauer suggests using the feedback link that consists of knowledge evaluation which is included in the modeling itself [2].

Among the most important features for effective learning materials is the level and type of interactivity, understood as the dialogue between students and presented materials or systems. Various levels of interactivity and difficulty are needed in active LOs, that offers users the opportunity to adapt the presentation of the content to their individual skills and needs, thus securing fast, situated, just-in-time up-/reskilling that is used in continuous education and training processes. It's important to ensure the flexible design of ALOs and comprehensive generation of user explanations. In order to achieve these goals the paper proposes a new information model of ALO that is based on context modeling. It relies on Brezillon research of Contextual Graphs (CG) and ALO information model research of Slotkienė [17]. The following is the discussion of conception of context that lies at the base of creation of context model.

2.2 CONTEXT

Traditionally and according to LOM specification, context is the environment in which the learning model (or learning object) is functioning. It is described by several parameters of metadata that characterize the environment. But in reality context is very complex. Context modeling in particular [9][17] has the notion of context that is much wider, it contains both the model (ALO) context and its environment context. Context has data and information [9] that describes elements of the domain, their relations and regularities of their changes. This way context can be viewed as a systematic description of the domain. In case of this conception of context we can implement the systematic approach that is opposed to the partial and one spot approach to the phenomena, e.g. common in traditional tests. In traditional tests, questions are often designed to contain one particular piece of knowledge; the answer to each question is seen as an independent data point [2]. In contrast, the individual actions within sequence of interactions of a simulation are often highly dependent on one another. Creation of graphic construct model by means of context modeling implements the methodical approach to phenomena that takes into account interaction among the elements of the whole system which constitutes an integral entity [3].

Context, as description of the part of the domain, is complex and dynamic, it covers various situations. The foundation of context information model (IM) is made of units that represent solutions of elementary tasks (solution steps), they that are called contextual elements.

2.3 CONTEXTUAL ELEMENT

Contextual element (CE) can be broken to focus, contextual data, rule and information [7][9]. Let's examine a simple example.

Let's say the student already knows the structure of the process for cyclic calculations and he/she has to understand the algorithm for cyclic sum of array elements and find out the differences from product algorithm. Let's take a simple problem of *initial value* of the accumulating variable. When the task means that *algorithm = sum*, the student must assign the accumulation variable *initial value* of 0, but when *algorithm = product*, the assigned value should be 1. Moreover, correct/incorrect are not sufficient as feedback of learning process, exhaustive explanations should be provided why the chosen solution was right or wrong and where the mistake was made.

Figure 1 shows a model of one CE. The sample ALO for algorithm of cyclic calculations is on the left, it contains the task and the solution with two empty spaces. The student has to fill in two sentences of this algorithm. The information model of filling the first sentence is presented on the right of the block diagram: i.e.

focus, the possible data (*Data 1)* and result (*Data 2*), information and rule. A sequence for applying the rule to the particular task and getting the result is also given. The sequence of actions for filling in the first sentence of the algorithm is shown in Table 1.

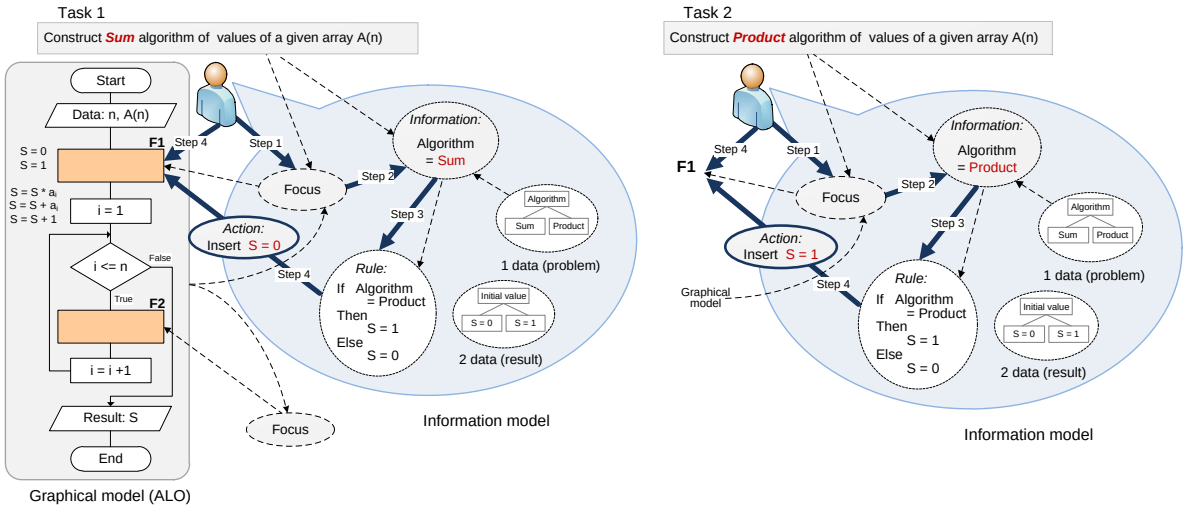


Figure 1. CE model

Table 1. The sequence of CE actions

Action	**Name**	**Content**	**Result**
1	Focus	Context analysis (Task 1 and graphic model)	Focus F1
2	Information	Analysis of Task 1 and Data 1	Infomation *algorithm = sum*
3	Rule	Rule input *algorithm = sum*	Initial value *S = 0*
4	Action	Deduction: *S = 0* must be placed to focus F1; Activity: place sentence *S = 0* to focus F1	Sentence *S = 0* in focus F1
Alternative: The same CE model can have different task (Task 2) where focus F1 in the algorithm of array element product must be filled with sentence *S = 1*.			

Information model that is shown in Figure 2 has two parts: the upper FD and lower CG. CG enumerates the tasks and their solutions that are available in the chosen topic. CG has two types of nodes: context elements and actions. Context element describes a step in the solution of the task, while the whole process of solution is presented by the graph path. The example of information model for cyclic calculations is given. The variability of topic data is reflected in FD: cyclic calculations can have three types of algorithms for sum, product or count. The initial value of accumulation variable can be 0 or 1 and it can be increased by the supplied value or 1. The data is related, e.g. accumulation variable is increased by 1 in the algorithm of count calculation.

In Figure 2 CG illustrates the example given in Figure 1, therefore it describes only a part of FD. CG describes the three possible tasks of the graphic model: the given block diagram can be filled in to create algorithm of sum, product or count. The first step of solution, filling in the initial value of accumulation variable, is described by CE C_1. If task is to form the sum algorithm, then after checking the condition of context element C_1 *Is algorithm = product?*, we get a negative answer, and, after choosing side branch C_1_N, perform action A1 (*S = 0*). The next branches are chosen by answering other questions of the CG path and lead to the actions of the problem solution. Construction of graphic model from contextual graph allows to verify the correctness of solution actions, specify errors and form explanations for the entire solution, where explanations gives details about the correctness of the solution and reveal the places of errors (if any). IM of CG can be used to form different ALOs that encompass one or more CEs, but ALO (graphic model) must include all paths that start at its CE. If ALO is made of a small number of CEs, there will be little paths that lead from them and, consequently, little variants (ALO instances). Such ALO will be easy (low level of difficulty). In order the number of CEs increases, the ALOs will vary in difficulty.

2.4 CONTEXT MODEL BASED IMPLEMENTATION OF ALO

Implementations of modeling practice employ many means ranging from specialized software to explicitly authored scenarios. The graphic system of distance assessment TestTool is under development in Kaunas University of Technology for almost a decade and is successfully used for modeling practice. The main feature that separates TestTool system from other similar tools is the ability for learners to perform practical actions that are required to answer graphic questions. This assessment system allows constructing various

problematic situations that are later presented for solution. This way student constructs the solution rather than selecting it.

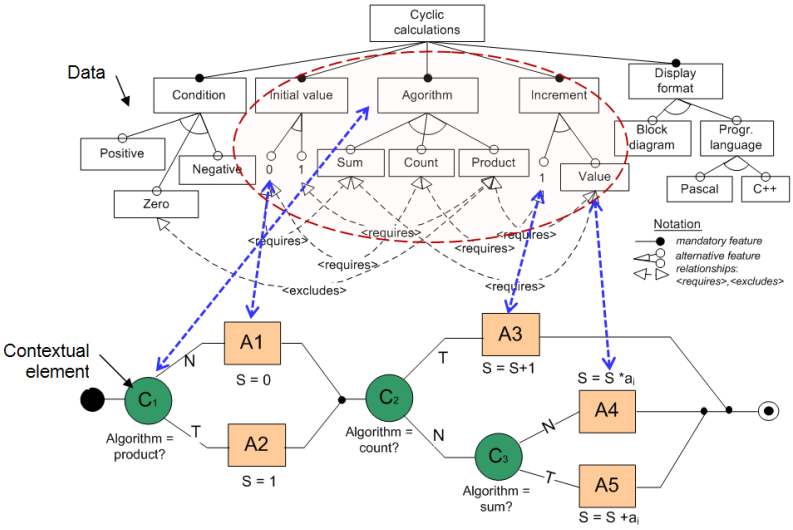


Figure 2. Information model of active learning model

Let's discuss the learning environment of some small or medium enterprise. Enterprise N collects from clients their faulty computers, an expert inspects them by running various tests, observes and analyzes test results, changes faulty components and returns computer to their owners. Construction of learning environment from the processes of computer repair that are used in enterprise N starts with creation of IM (made of FD and CG). TestTool uses IM to form the graphic model of computer repair (ALO). The base of this graphic model is the graphic pattern of testing scenarios that is prepared in accordance to CG. The creation of graphic model was supported by clips of real documents, screenshots of testing programs and other practical material. Four different versions of graphic model were created and each version was used to construct easy, moderate and difficult ALOs accordingly to the requirements of CG and FD.

Figure 3 shows a version of graphic model, where the presented context is made of the data of computer inspection, the graphic pattern of testing scenarios and the lists of possible conditions (C) and actions (A). For the given task student can use fault description *No power (no signs of life*) to construct the computer testing scenario, while modeling system will evaluate correctness of the solution and describe the errors. In the graphic pattern of testing scenarios student has to select from the list only those Ci and Ai that are required to repair the given computer. Computer faults in other ALO tasks will have different fault descriptions or testing screenshots, they will require different actions for inspection and component testing/replacement, i.e. we will have various scenarios of computer testing and repair.

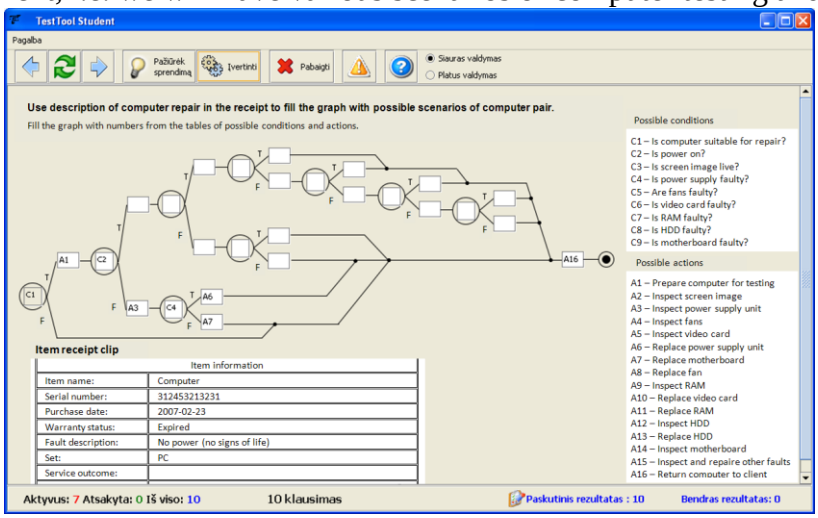


Figure 3. ALO example

The earlier described graphic model is created by means of graphic testing. In order to formalize the structure of design process, fully implement features of modeling-simulation and achieve higher degree of

system flexibility, a new version of the system is developed. The new version of TestTool 5.3 implements modeling-simulation features that use the new context modeling based information model of ALO (see chapter 2.3). The system was also redesigned to support improved flexibility and upcoming transition to service-oriented architecture. The next chapter discusses the details of the initial phase of this implementation – the architecture of TestTool 5.3.

3 ARCHITECTURE OF TESTTOOL 5.3.

The TestTool 5.3 system is implemented as an assessment and graphic modeling engine. It's a modular client-server system with several classes of users. The current system works as a network of distributed services and is planned to refactor to service-oriented architecture. Version 5.3 already supports the flexible incorporation of new scenarios and overall system changes.

3.1 STRUCTURE AND FUNCTIONS OF THE SYSTEM

The system consists of resource repository (RR) and operation center (OC) services that are consumed by administration, student and authoring client software (Figure 4). TestTool system can be used by users of three different levels – super-administrators, administrators and learners. Learning resources are made of courses, exams, tests, questions, variants and question answers. The educational process consists of the preparation of learning material (initial preparation stage) and its application for training of student groups (learning stage). During the preparation stage question variants are created (or reused) and combined to questions, tests and exams (administrator/author responsibility). Learning stage involves students that prepare for the assessment (exams are in practice mode) and later take tests (exams are in assessment/testing mode).

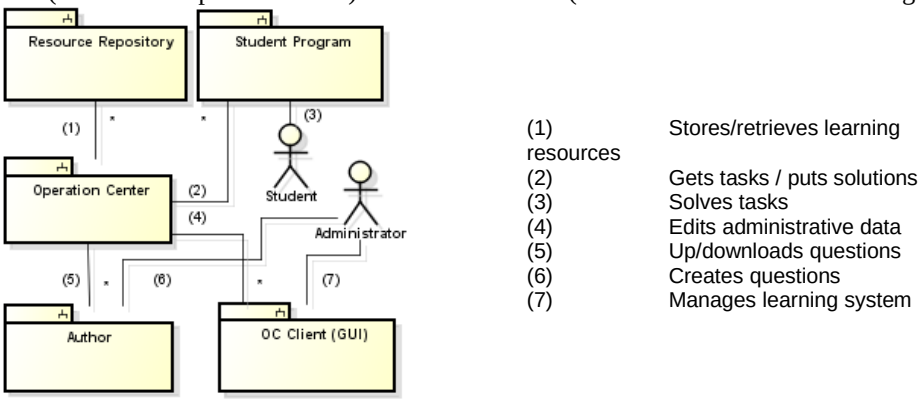


Figure 4. Structure of TestTool 5.3 system

As already mentioned, there are several different roles of TestTool 5.3 actors/users. Super-administrators use OC GUI to manage the descriptions of administrators. Administrators employ the same GUI for registration of courses, users, user groups, tests and exams; they also select the evaluation scenarios. When acting as a domain expert, administrator uses authoring tool GUI to create questions and implement knowledge structure of graphic models. Students have the student GUI that is used to connect to the system, select working mode and take various tasks. In practice mode learner can freely choose tasks and take them multiple times, but results of these actions are not saved. In testing mode each task of the exam can only be taken once and all results are saved in RR.

Operation centre has access to resources in RR and carries out administration of users and tests, it also performs user authentication and checks execution rights for all operations. Access rights are also transferred from OC to resource repository that has no other means of data protection (see figure 4). A single operation centre can use several resource repositories that are accessed in server-client mode. RR service stores questions, student answers to those questions and tools for general resource management. RR uses special scenarios to evaluate the answers. Resource repositories are managed by administrators through GUI of operation centre. Each administrator can be assigned more than one resource repository and RR can belong to several administrators.

The authoring tool is used by administrator, or domain expert, to implement the knowledge structure of graphic model. This authoring tool itself contains many technical features that can be used for that purpose: drawing primitives, elements for simulation control, textual and graphic explanations of errors, metadata and various multimedia effects. It facilitates the creation of active learning objects with different levels of interactivity [4]. Author tool is used for creation of both traditional questions and models of graphic construction (they are coordinated with the contextual graph). These question types also need different scenarios of answer evaluation. The authored questions can be directly uploaded to the selected resource repository or they can be

saved to hard disk and later uploaded through OC. The RR stored learning material can be private for the administrator that uploaded it but also can be made public to all administrators that use the services of that RR.

3.2 Scenarios and their application

The system employs different scenarios for preparation and execution of studies. They facilitate the implementation of flexible structures for assessment and graphic modeling. Arbitrary number of scenarios can be added as system extensions in Java Archive files (*.jar). Figure 5 shows a sample course with 2 exams. Exam2 with traditional questions and graphic model simulation tests in practice mode is created for preparation, Exam1 is for assessment. Each exam is composed of one or more (two in this case) tests that comprise a set of questions. Exams and tests are assigned evaluation scenarios that govern how student answers are treated. Tests have two scenarios: one for questions (different scenarios are required for traditional and simulation questions) and one for the whole test. Exams need just one scenario that sums test results.

The preparation phase covers resource editing and making learning material (exams) ready for studies and assessment. To construct a new exam administrator has to carry out these actions: 1) create exam name, 2) select private (personal) or public question from the storage, 3) create test and assign questions to it, 4) assign test to the exam, 5) choose the scenario for presentation of question variants, 6) choose the scenario for evaluation of questions or model, 7) choose the scenarios for test and exam evaluation, 8) assign student group to the exam. During preparation phase resources can be edited according to the patterns of preparation scenarios.

When learning phase is active, student program uses OC to get exam questions from RR, put results of question answers to RR and receive evaluation from RR. Scenarios for learning phase support:

- Question evaluation – evaluates answers to traditional questions;
- Model evaluation – evaluates answers to questions with graphic models according to information model CG;
- Test or exam evaluation – concludes the evaluation of test or exam, e.g. test questions can be evaluated as weighted or not, ”exam evaluation” scenario can use some expression to calculate the final evaluation of several exams, etc.;
- Presentation of question variants – designates the number of question variants that are presented to the student, the way of variants selection (e.g. random or metadata driven user choice), etc.

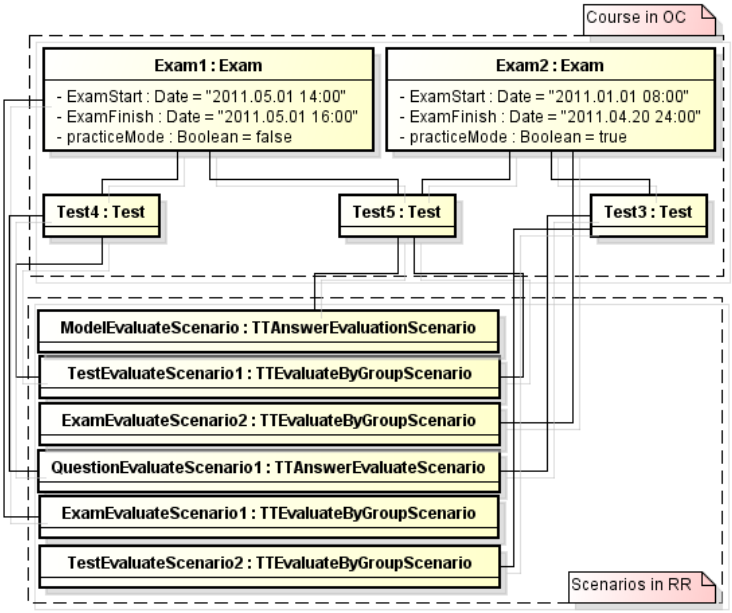


Figure 5. Sample course structure with various evaluation scenarios

Learning material of traditional tests is meant for preparation and self-studies, usually it can't be applied to assessment, in which case extra protection is required. Thus the ability to use different modes of same questions (Test5 in Figure 5) both for preparation (Exam2, simulation mode, in Figure 5) and assessment (Exam1 in Figure 5) is very important.

3.3 System implementation

Resource repository and operation centre are implemented using the same programming technologies: Java objects, relational database and design patterns that have become popular in the domains of architecture, software design, human computer interaction, Web 2.0, organizational structures and pedagogy as a way to

communicate successful practical knowledge. Patterns capture proven solutions for recurrent problems with respect to fitting contexts [12]. The standard enterprise Java pattern of Data Access Object (DAO) [1] was used to clearly separate the business logic and database layers that improved the maintainability and extensibility. DAO design pattern supports the object persistence and data access logic independence from any particular persistence mechanism or API. This approach provides the flexibility to change application's persistence mechanism over time without the need to re-engineer application logic that interacts with the DAO tier [19].

TestTool system provides a simple, consistent API for data access that does not require knowledge of specific technologies, thus widening the base of extension and plugin developers. All database access in TestTool 5.3 system is made through DAO that supports data encapsulation. Each DAO instance is responsible for the single primary domain object or entity, but only if a domain object has an independent lifecycle. At the lowest layer Hibernate is used to connect application and database, it's an excellent ORM framework which allows developers to access database by using familiar object-oriented approach [19]. Hibernate has the following characteristics: transparency in providing mapping between object and relational database, support for multiple databases by using a unified interface, advanced caching mechanism and locking strategy that significantly reduce operations against the database. It's provided under free open source license; source code can be consulted and modified for required functional customization. Hibernate is a lightweight package, it doesn't introduce many complex constructs, so it's easy to debug, that reduces the burden of TestTool maintenance and extension programmers [20].

The whole TestTool system is implemented as a set of program layers; each of them has clear boundaries and interconnections to adjacent layers. Layered structure is essential for evolving system that is constantly extended for research of new e-learning features and adapted to various middleware platforms.

1. Conclusions and future works

1. Creation of high quality e-learning technologies that simulate sophisticated and dynamic systems requires the simplification of learning models by decomposing them to smaller problems that can address individual skills and needs of the learners. The paper presented a new information model that comprises contextual graph and a feature diagram, this model is taken as a base for implementation of experimental tool for ALO construction TestTool 5.3. Simulation, experimentation and practice as well as explanation, assessment and self-control functionality can be implemented using active learning objects (ALOs).
2. ALOs are formed from CG sub-graphs with different number of CE nodes and all paths that lead from these nodes. This creates the ability to consistently change ALO difficulty while ensuring completeness and adequacy of the model. Course with such ALOs (i.e. LO of higher aggregation level) fulfills the completeness requirement by realizing all phases experiential learning cycle.
3. Very effective and fast applications for (self-) assessment and simulation-based active learning can be created for any particular domain. Real world usage of ALO is reinforced by utilization of authentic materials. Noteworthy is the possibility to use tasks with ALO graphic models both in practice and testing modes, it clearly distinguishes such tasks from the traditional tests.
4. The TestTool 5.3 system is implemented as an assessment and graphic modeling engine. Future works include further development of TestTool subsystem for context model design, implementation of CG XML parsing in author and student programs, improving generation of explanations, implementation of standard compliant learning services and transition to service oriented architecture (SOA).
5. The second stage of development includes plans for transition to SOA with reusable services. Grid middleware based distributed service architecture was already successfully used in earlier versions of TestTool [19]. SOA implementation is expected to support the following features:
    - High level of abstraction where business logic is completely hidden from service consumers. Implementation of this feature is facilitated by Date Access Object and logic tiers that hide main business logic and expose only RMI interfaces.
    - Loose coupling: TestTool services have minimal interdependencies and are ready for refactoring to web services.
    - Service statelessness: Resource Repository stores minimal amount of data context information so any invocation of this service is not bound to earlier or later service invocations.

    Resource repository and operation centre can be easily implemented as two individual services that satisfy main requirements of SOA. The upcoming version of TestTool system can be created as EJB container that is accessible through RMI and web services (only RMI interface is available now). Having two separate interfaces are useful for situations where better performance of RMI or standard compliance of web services are required.

REFERENCES

[1] **Alur D.**, Core J2EE Patterns: Best Practices and Design Strategies. Code Design Series, *Sun Microsystems*, inc, 2003

[2] **Aukštakalnis N., Baniulis K. T., Pauliutė J., Slotkienė A.** Graphical model: the means for simulation-based learning. *Journal of Computing and Information Technology (CIT)/University of Zagreb*. *Zagreb : University Computing Centre.* ISSN 1330-1136. 2008, Vol. 16, no. 4, p. 303-309.

[3] **Baniulis K., Pauliutė J.**, **Slotkienė A.** Knowledge Modelling dynamic phenomena using graphical assessment environment. *Jaunųjų mokslininkų darbai. Šiaulių universiteto leidykla*. ISSN 1648–8776. 2007, nr. 2, p. 108–112.

[4] **Baniulis K., Keršienė V., Petreikienė V., Slotkienė A.** A Case Study: Impact of the Interactivity Level to E-Learning Outcomes. *IT 2010: proceedings of the 16th International Conference on Information and Software Technologies, Kaunas, Lithuania, Kaunas University of Technology. Kaunas: Technologija,* 2010, p. 101-107.

[5] **Baniulis K., Tamulynas B.** The use case specification of actions in the goal oriented knowledge based learning environment. *1st International ELeGI Conference on Advanced Technology for Enhanced Learning*, *15-16 March 2005, Naples, Italy, Swindon: British Computer Society*, ISSN1477-9358, 2005, p. 1-8.

[6] **Bauer M. et al.** Using Evidence-Centered Design to Develop Advanced Simulation-Based Assessment and Training. *World Conference on E-Learning in Corp., Govt., Health., & Higher Ed*, 2003, p. 1495-1502.

[7] **Brezillon P.** Context Dynamic and Explanation in Contextual Graphs. *Modeling and Using Context (CONTEXT-03)*, LNAI 2680, Springer Verlag, p. 94-106, 2003.

[8] **Brezillon P.J.** Contextualized Explanations. *Proceedings of International Conference on Expert Systems for Development*, Volume, Issue, 28-31, 1994, p. 119 – 124.

[9] **Brezillon P**. Context modeling: Task model and model of practices. *Modeling and Using Context (CONTEXT-07)*, 2007, p. 122-135.

[10] **Daukilas S., Kačinienė I., Vaišnorienė D., Vaščila V.** Factors that Impact Quality of E-teaching/learning Technologies in Higher Education. *The quality of Higher education*, 2008/5 p. 131-151.

[11] **Heymans P., Schobbens P.-Y. Trigaux J.-C., Bontemps Y., Matulevicius R., Classen A.** Evaluating formal properties of feature diagram languages. *The Institution of Engineering and Technology, IET Softw.*, 2008, Vol. 2, No. 3, pp. 281–302.

[12] **Kohls C., Wedekind J.** Investigations of E-Learning Patterns: Context Factors, Problems and Solutions. *Knowledge Media Research Center, Germany*, 2010, ISBN1609601440.

[13] **Kolb D.** Experiential learning: experience as the source of learning and development. *Prentice Hall. New Jersey*, 1984.

[14] **Matic D., Butorac D., Kegalj H.** Data Access Architecture in Object Orientes Applications Using Design Patterns, *Dubrovnik, Croatia, IEEE MELECON* 2004, May 12-15, 2004, ISBN0-7803-8271-4, pp. 596-598.

[15] **Reklaitis V., Baniulis K., Okamoto T.** Shaping e-Learning applications for a service-oriented GRID. *2nd International LeGE-WG Workshop on e-Learning and Grid Technologies: A Fundamental Challenge for Europe, Paris, France*. 3rd & 4th March 2003.

[16] **Shute V., Glaser R.** Large-scale evaluation of an intelligent tutoring system. *Smithtown. Interactive Learning Environments*, 1990, vol. 1, p. 51-76.

[17] **Slotkienė A.** Aktyvaus mokymosi objekto projektavimo metodas ir jo tyrimas. *Kauno technologijos universitetas*, daktaro disertacija, 2009.

[18] **Štuikys V., Damaševičius R**. Development of Generative Learning Objects Using Feature Diagrams and Generative Techniques. *Informatics in Education*, 2008, vo 7, no. 2 p.1-12.

[19] **Tamulynas B., Baniulis K.T., Aukstakalnis N.** Enhanced learning and understanding in knowledge testing environments. ITI 2007*: Proceedings of the 29th InternationalConference on Information Technology Interfaces*, June 25-28, 2007, *Dubrovnik, Croatia, University of Zagreb, University Computing Centre*. 2007, ISBN978-953-7138-09-7, pp. 287–292.

[20] **Wu P., Yin K.** Application research on a persistent technique based on Hibernate. *Comput. Center, Henan Univ., Kaifeng, China*, 5th of August, 2010, vol 1, pp. 629-631, ISBN978-1-4244-7164-5.

[21] **Wu P.** Exploration of a Realization Pattern of System Based on Hibernate, *Compur Center Henan University Kaifeng, China, ICCDA 2010: 2010 International Conference on Computer Design and Applications*, June 25-27, 2010, vol 5, pp. 108-110, ISBN978-1-4244-7164-5.

3 PRIEDAS. TESTAVIMO PLANAS

Žemiau yra pateikta testavimo metu naudojamas testavimo planas.

Klasių testai yra pateikiami lentelėse:

1. RESURSŲ SAUGYKLOS VIENETŲ TESTAVIMAS

24 lentelė. AnswersResourceStorageDAO testavimas

Nr.	Testuojama klasė: AnswersResourceStorageDAO (Priklauso resursų saugyklos posistemei) Testavimo rinkinys: AnswersResourceStorageDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
3.	Studento atsakymo gavimas iš resursų saugyklos. Testavimo atvejo metu paleidžiamas metodas (getItem(long)), į įėjimo parametrus paduodamas resurso ID.	Grąžinamas resursas byte pavidalu sutampantis su prieš tai išsaugotu duomenų bazėje.
4.	Studento atsakymo saugojimas resursų saugykloje. Į metodo (storeItem(byte)) įėjimo parametrus paduodamas resursas byte pavidalu (pvz. įvedamas iš failo).	Grąžinamas long ID išsaugoto resurso iš duomenų bazės.
5.	Studento atsakymo naikinimas resursų saugykloje. Kviečiamas resurso naikinimo metodas (removeItem(long)) su resurso, kuris išsaugotas duomenų bazėje id.	Grąžinimas true. Duomenų bazėje nelieka resurso su nurodytu ID. Jei nurodytas neegzistuojantys resurso ID grąžinamas false.

25 lentelė. EvaluationResultsDAO testavimas

Nr.	Testuojama klasė: EvaluationResultsDAO (Priklauso resursų saugyklos posistemei) Testavimo rinkinys: EvaluationResultsDAOTest

	Testavimo atvejis	Laukiamas rezultatas
6.	Įvertinimo rezultato gavimas iš resursų saugyklos. Kviečiamas metodas getEvaluationResult(long), kur į įėjimo parametrus paduodamas ieškomo rezultato ID resursų saugykloje.	Sėkmės atveju gaunama Collection tipo įvertinimo informacija.
7.	Įvertinimo rezultato saugojimas resursų saugykloje Paleidžiamas metodas storeEvaluationResult(long[], long[], String[], String, float, int), kur į įėjimo parametrus paduodamas: 1. Variantų ID (resursų saugykloje). 2. Studentų atsakymų ID (resursų saugykloje). 3. Vertinimo scenarijaus pavadinimas. 4. Taškai (įvertinimo rezultatas). 5. Prikalausomai nuo vertinimo tipo 1 – jei grupės įvertinimas, 0 - jei vieno resurso įvertinimas.	Sėkmės atveju grąžinamas ID išsaugoto rezultato resursų saugykloje.
8.	Įvertinimo rezultato trynimas iš resursų saugyklos Paleidžiamas metodas removeEvaluationResult(long), kur į įėjimo parametrus paduodamas trinamo rezultato ID.	Naikinimo sėkmės atveju grąžinamas true, priešingai false.

26 lentelė. ResourceEvaluateDAO testavimas

Nr.	Testuojama klasė: ResourceEvaluateDAO (Priklauso resursų saugyklos posistemei) Testavimo rinkinys: ResourceEvaluateDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
9.	Resurso įvertinimas pagal pasirinktą scenarijų, Testavimo atvejo metu paleidžiamas metodas evaluate(byte[], byte[], String[], String) su	Metodas grąžino įvertinimo taškus.

	įėjimo parametrais: 1. Variantas (byte formatu). 2. Studento atsakymas (byte formatu). 3. Scenarijaus valdymo parametrai. 4. Scenarijaus pavadinimas. Pastaba: Sistemoje turi egzistuoti paduodamas įvertinimo scenarijus.	
10.	Grupės resursų įvertinimas Testavimo atvejo metu paleidžiamas metodas evaluateByGroup(long[], String[], String) su įėjimo parametrais: 1. Masyvas studento atsakymų ID iš resursų saugyklos. 2. Scenarijaus valdymo parametrai. 3. Scenarijaus pavadinimas.	Metodas grąžino įvertinimo taškus.

27 lentelė. ResourceStorageDAO testavimas

Nr.	Testuojama klasė: ResourceStorageDAO (Priklauso resursų saugyklos posistemei) Testavimo rinkinys: ResourceStorageDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
11.	Parodyti visą sąrašą variantų resursų saugykloje. Testavimo atvejo metu paleidžiamas metodas getAllVariants().	Gautas sąrašas List tipo visų variantų (resursų) saugomų resursų saugykloje.
12.	Parodyti visus prieinamus paruošimo scenarijus resursų saugykloje. getAvailablePrepareScenarios().	Išvestas ne tuščias String masyvas, kuriame yra visi pasiekiami serveryje paruošimo scenarijai.
13.	Parodyti visus prieinamus grupės resurso vertinimo scenarijus resursų saugykloje. Testavimo atvejo metu paleidžiamas metodas getAvalaibleEvaluateByGroupScenario().	Išvestas ne tuščias String masyvas, kuriame yra visi pasiekiami serveryje grupės variantų įvertinimo scenarijai.
14.	Parodyti visus prieinamus vertinimo	Išvestas ne tuščias String

	scenarijus resursų saugykloje. Testavimo atvejo metu paleidžiamas metodas getAvalaibleEvaluationScenario().	masyvas, kuriame yra visi pasiekiami serveryje vertinimo scenarijai.
15.	Gauti resursą Testavimo atvejo metu paleidžiamas metodas getItem(long), į įėjimo parametrus paduodamas išsaugoto resurso ID.	Gautas pasirinktas resursas byte formatu.
16.	Paruošti resursą pagal nurodytą scenarijų Testavimo atvejo metu paleidžiamas metodas prepareByItem(long, String[], String) su įėjimo parametrais: 1. Resurso ID. 2. Pasirinkto scenarijaus parametrai. 3. Pasirinkto paruošimo scenarijaus pavadinimas.	Naujai išsaugoto resursų saugykloje ID grąžintas vykdant metodą.
17.	Panaikinti resursą iš resursų saugyklos Testavimo atvejo metu paleidžiamas metodas removeItem(long) pagal ID.	Grąžina true jei resursas sėkmingai surastas ir panaikintas, priešingai false.
18.	Vykdyti varianto (resurso) paiešką. Galima leisti šios metodus: searchVariants(long) – paieška pagal resursus, kurie priklauso nurodytam kursui ID (iš operacijų centro). searchVariants(long,String) - paieška pagal kursą ir raktą. searchVariants(long, String, String) – paieška pagal kursą, raktą ir papildomai nurodytą informaciją. Pastaba: šie metodai inicijuojami išskirtinai iš operacijų centro. Šie metodai paleidžiami be Operacijų centro išskirtinai testavimo metu.	Vykdant metodus grąžintas sąrašas (List) variantų (resursų).
19.	Išsaugoti resursą resursų saugykloje. Testavimo atvejo metu paleidžiamas vienas iš metodų: storeItem(byte[]) – resursas byte formatu,	Grąžintas išsaugoto resurso ID.

	storeItem(byte[],String) – resursas byte, formatu su dėstytojo raktu, storeItem(TOAVariant) – resursas byte formatu su Varianto siuntimo objektu. Pastaba: storeItem(byte[],String) ir storeItem(TOAVariant) metodai paleidžiami be Operacijų centro išskirtinai testavimo metu.	

2. OPERACIJŲ CENTRO VIENETŲ TESTAVIMAS

28 lentelė. OCAdministratorDAO testavimas

Nr.	Testuojama klasė: OCAdministratorDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCAdministratorDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
20.	Pridėti naują administratorių. Testavimo atvejo metu paleidžiamas metodas createOrUpdateAdministrator(TOAdministrator) į įėjimo parametrus paduodant suformuotą administratoriaus siuntimo objektą be ID jame.	Grąžinamas administratoriaus siuntimo objektas, kuriame yra ID išsaugotos administratoriaus. Išsaugota informacija pagal nurodytus įėjimo parametrus.
21.	Atnaujinti esančio administratoriaus duomenis. Testavimo atvejo metu paleidžiamas metodas createOrUpdateAdministrator(TOAdministrator) į įėjimo parametrus paduodant suformuotą administratoriaus siuntimo objektą, kuriame nurodytas ID sutampa su keičiamo administratoriaus ID duomenų bazėje.	Grąžinamas administratoriaus siuntimo objektas, kuriame yra ID išsaugotos administratoriaus. Pakeista informacija pagal nurodytus įėjimo parametrus.
22.	Gauti administratoriaus duomenis pagal ID. Testavimo atvejo metu paleidžiamas metodas getAdministratorById(long), į įėjimo parametrus paduodant ID iš duomenų bazės,	Grąžintas administratoriaus siuntimo objektas.
23.	Panaikinti administratorių pagal ID. Testavimo atvejo metu paleidžiamas metodas	Grąžina true naikinimo sėkmės atveju, priešingai false.

	removeAdministrator(long), įėjimo parametras – duomenų bazėje saugomo administratoriaus objekto ID.	
24.	Gauti visus kurso administratorius Testavimo atvejo metu paleidžiamas metodas searchAdministrator(String, String, TOCourse) su įėjimo parametrais: 1. Paieškos laukas, 2. Paieškos raktas, 3. Kurso siuntimo objektas su nurodytu kurso ID.	Grąžina administratoriaus siuntimo objektų masyvą.
25.	Vykdyti administratoriaus paieška. Testavimo atvejo metu paleidžiamas metodas searchAdministrator(String, String, String, boolean) su įėjimo parametrais: 1. Paieškos laukas, 2. Paieškos raktas, 3. Rezultatus rūšiuoti pagal lauką, 4. Dydejančiai (true) arba mažėjančiai (false).	Grąžina administratoriaus siuntimo objektų masyvą.

29 lentelė. OCCourseDAO testavimas

Nr.	Testuojama klasė: OCCourseDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCCourseDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
26.	Priskirti administratorių prie kurso. Testavimo atvejo metu paleidžiamas metodas addCourseToAdministrator(long, long) su įėjimo parametrais: 1. Kurso ID, 2. Administratoriaus ID.	Grąžina kurso siuntimo objektą..
27.	Sukurti naują arba atnaujinti kursą. Testavimo atvejo metu paleidžiamas metodas createOrUpdateCourse(TOCourse) į įėjimo	Grąžina kurso siuntimo objektą su atnaujinta informacija.

	parametrą paduodant suformuotą kurso siuntimo objektą.	
28.	Gauti visus kursus. getAllCourseList().	Grąžina visų prieinamų kursų siuntimo objektų masyvą.
29.	Gauti kursą pagal ID. Testavimo atvejo metu paleidžiamas metodas getCourseById(long) į įėjimo parametrą paduodant kurso ID.	Grąžina kurso siuntimo objektą su kurso informacija.
30.	Gauti kursą pagal administratoriaus ID. Testavimo atvejo metu paleidžiamas metodas getCoursesByAdministratorId(long) į įėjimo parametrą paduodant egzistuojančio duomenų bazėje administratoriaus ID.	Grąžina kurso siuntimo objektą.
31.	Panaikinti kursą. Testavimo atvejo metu paleidžiamas metodas removeCourse(long) su įėjimo parametrais:	Grąžina true panaikinimo sėkmės atveju, priešingai false.
32.	Atimti iš administratoriaus teises administruoti nurodyta kursą. Testavimo atvejo metu paleidžiamas metodas removeCourseFromAdministrator(long, long) su įėjimo parametrais: 1. Kurso ID, 2. Administratoriaus ID.	Grąžina true jei pavyko, priešingai false.
33.	Ieškoti kursą pagal pavadinimą. Testavimo atvejo metu paleidžiamas metodas searchCourseByName(String, String, boolean) su įėjimo parametrais: 1. Kurso pavadinimo dalis. 2. Rūšiuoti pagal lauką (courseId arba courseName). 3. Pateikti dydėjančiai true, arba mažėjančiai false. Pastaba:	Grąžina rastų kursų siuntimo objektų masyvas.

30 lentelė. OCExamDAO testavimas

Nr.	Testuojama klasė: OCExamDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCExamDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
34.	Sukurti arba atnaujinti egzaminą. Testavimo atvejo metu paleidžiamas metodas createOrUpdateExam(TOExam, TOCourse) su įėjimo parametrais: 1. Egzamino siuntimo objektas su pažymėtu ID, jei norima atnaujinti informaciją. 2. Kurso siuntimo objektas su pažymėtu ID.	Grąžinamas egzamino siuntimo objektas su atnaujinta informacija.
35.	Gauti visus prieinamus egzamino organizavimo scenarijus. Testavimo atvejo metu paleidžiamas metodas getAvalaibleExamScenarios().	Išvestas ne tuščias String masyvas, kuriame yra visi pasiekiami serveryje klausimų variantų organizavimo scenarijai.
36.	Gauti egzamino informaciją pagal ID. Testavimo atvejo metu paleidžiamas metodas getExamById(TOExam) į įėjimo parametrą paduodant tuščią egzamino siuntimo objektą, tik tai su pažymėtų ID.	Grąžinamas surastas egzamino siuntimo objektas su egzamino informacija iš duomenų bazės.
37.	Gauti egzaminų sąrašą pagal pasirinktą kursą. Testavimo atvejo metu paleidžiamas metodas getExamListByCourse(TOCourse) į įėjimo parametrą paduodant kurso siuntimo objektą su nurodytu ID.	Grąžinamas surastų egzamino siuntimo objektų masyvas.
38.	Panaikinti nurodytą egzaminą. Testavimo atvejo metu paleidžiamas metodas removeExam(TOExam, TOCourse) su įėjimo parametrais: 1. Egzamino siuntimo objektas su pažymėtų ID. 2. Kurso siuntimo objektas su pažymėtu ID. Pastaba: naikinami egzaminai turėtų priklausyti nurodytam kursui.	True naikinimo sėkmės atveju, priešingai false.

31 lentelė. OCGroupDAO testavimas

Nr.	Testuojama klasė: OCGroupDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCGroupDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
39.	Pridėti studentus į grupę. Testavimo atvejo metu paleidžiamas metodas addStudentsToGroup(List<TOStudent>, TOGroup) su įėjimo parametrais: 1. Sąrašas studentų siuntimo objektų kuriuos, reikia prideti į duotą grupę. 2. Grupės siuntimo objektas su pažymėtu ID.	Grąžinamas grupės siuntimo objektas.
40.	Sukurti arba atnaujinti grupės informaciją. Testavimo atvejo metu paleidžiamas metodas CreateOrUpdateGroup(TOGroup, TOCourse) su įėjimo parametrais: 1. Grupės siuntimo objektas su pažymėtu ID, jei norima atnaujinti informaciją. 2. Kurso siuntimo objektas su pažymėtu ID.	Grąžinamas grupės siuntimo objektas su naujai sukurta arba atnaujinta informacija.
41.	Gauti visas grupes, kurios priklauso nurodytam kursui. Testavimo atvejo metu paleidžiamas metodas getAllGroupsByCourse(TOCourse) į įėjimo parametrą paduodant kurso siuntimo objektą su pažymėtu ID. Pastaba:	Grąžinamas rastų grupių siuntimo objektų masyvas.
42.	Gauti grupės informaciją pagal jos ID. Testavimo atvejo metu paleidžiamas metodas getGroupById(long) į įėjimo parametrą paduodant saugomos duomenų bazėje grupės ID.	Rastos grupės siuntimo objektas.
43.	Panaikinti grupę. Testavimo atvejo metu paleidžiamas metodas	True naikinimo sėkmės atveju, priešingai false.

	removeGroup(long) su įėjimo parametru ID saugomos grupės.	
44.	Vykdyti grupės paiešką pagal jos vardą. Testavimo atvejo metu paleidžiamas metodas searchGroupByName(String) į įėjimo parametrą paduodant paieškos raktą.	Grąžinamas rastų grupių siuntimo objektų masyvas.

32 lentelė. OCQuestionDAO testavimas

Nr.	Testuojama klasė: OCQuestionDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCQuestionDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
45.	Patikrinti ar sukurti klausimai turi variantus resursų saugykloje. Testavimo atvejo metu paleidžiamas metodas checkStorageSynchronization(List<TOQuestion>) į įėjimo parametrą paduodant sąrašą klausimo siuntimo objektų su pažymėtu ID. Pastaba: turi būti sukurtas kursas ir jam priskirta resursų saugykla. Patartina sukurti pora šio testavimo atvejo tikrinamų duomenų variantų: kai sukurtiems klausimams atitinka variantai (resursai) resursų saugykloje, kai neatitinka.	Grąžinamas klausimų siuntimo objektų masyvas, kurie neturi variantų atitikmens, jei tokiu nerasta grąžinamas tuščias masyvas.
46.	Sukurti naują klausimą arba atnaujinti esamo informaciją. Testavimo atvejo metu paleidžiamas metodas createOrUpdateQuestion(TOQuestion, TOAVariant, TOStorage, boolean) su įėjimo parametrais: 1. Klausimo siuntimo objektas su pažymėtu ID, jei informacija atnaujinama. 2. Saugyklos siuntimo objektas, kurioje randasi variantai. 3. True, jei norima išsaugoti kaip naują variantą	Grąžinamas klausimo siuntimo objektas su nauja arba atnaujinta informacija iš duomenų bazės.

	resursų saugykloje. Pastaba: kuriant klausimą, prieš tai turi būti įsitikinama, kad resursų saugykloje, kuri yra priskirta prie esamo kurso išsaugotas nurodomas variantas.	
47.	Gauti klausimo informaciją pagal jo ID. Testavimo atvejo metu paleidžiamas metodas getQuestionById(long) į įėjimo parametrą paduodant klausimo ID.	Grąžinamas klausimo siuntimo objektas su informacija iš duomenų bazės.
48.	Panaikinti klausimą (nenaikinant varianto resurso saugykloje). Testavimo atvejo metu paleidžiamas metodas removeQuestion(TOQuestion) į įėjimo parametrą paduodant klausimo siuntimo objektą su pažymėtu ID.	Grąžinamas true naikinimo sėkmės atveju, priešingai false.
49.	Panaikinti klausimą (naikinant variantą iš resurso saugyklos). Testavimo atvejo metu paleidžiamas metodas removeQuestion(TOQuestion, TOStorage) su įėjimo parametrais: 1. Klausimo siuntimo objektas su pažymėtu ID. 2. Resurso saugyklos siuntimo objektas su pažymėtu ID.	Grąžinamas true naikinimo sėkmės atveju, priešingai false.
50.	Vykdyti klausimų paiešką pagal nurodytą raktą. Testavimo atvejo metu paleidžiamas metodas searchQuestionsByName(String) į įėjimo parametrą paduodant paieškos raktą.	Grąžinamas rastų klausimo siuntimo objektų masyvas su informacija iš duomenų bazės.

33 lentelė. OCResourceStorageInformationDAO testavimas

Nr.	Testuojama klasė: OCResourceStorageInformationDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCResourceStorageInformationDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
51.	Pridėti prie kurso resursų saugykla arba	Grąžinamas atnaujintas resurso

	atnaujinti jos informaciją. Testavimo atvejo metu paleidžiamas metodas createOrUpdateResourceStorage(TOStorage, TOCourse) su įėjimo parametrais: 1. Resursų saugyklos siuntimo objektas, atnaujinimo atveju paduodamas su nurodytu ID. 2. Kurso siuntimo objektas su nurodytu ID.	saugyklos siuntimo objektas su informaciją iš duomenų bazės.
52.	Gauti resurso saugyklos informaciją pagal jos ID. Testavimo atvejo metu paleidžiamas metodas getResorceStorage(long) į įėjimo parametrą paduodant resurso saugyklos ID iš duomenų bazės.	Grąžinamas resurso saugyklos siuntimo objektas su informacija iš duomenų bazės.
53.	Gauti resurso saugyklos informaciją pagal nurodyto kurso ID. Testavimo atvejo metu paleidžiamas metodas getResourceStorageConnectedToCourse(long) į įėjimo parametrą paduodant kurso ID.	Grąžinamas resurso saugyklos siuntimo objektas su informacija iš duomenų bazės.
54.	Naikinti nurodyta resursų saugyklą. Testavimo atvejo metu paleidžiamas metodas removeResourceStorage(TOStorage) į įėjimo parametrą paduodant resurso saugyklos siuntimo objektą su pažymėtu ID.	Grąžinamas true naikinimo sėkmės atveju, priešingai false.

34 lentelė. OCStudentDAO testavimas

Nr.	Testuojama klasė: OCStudentDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCStudentDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
55.	Sukurti naują arba atnaujinti esamą studentą. Testavimo atvejo metu paleidžiamas metodas createUpdateStudent(TOStudent) į įėjimo parametrą paduodant studento siuntimo objektą	Grąžinamas studento siuntimo objektas su atnaujinta informacija iš duomenų bazės.

	atnaujinimo atveju pažymint ID.	
56.	Gauti studento informaciją pagal jo ID. Testavimo atvejo metu paleidžiamas metodas getStudentById(long) į įėjimo parametrą paduodant ID.	Grąžinamas studento siuntimo objektas su informacija iš duomenų bazės.
57.	Gauti visus nurodytos grupės studentus. Testavimo atvejo metu paleidžiamas metodas getStudents(TOGroup) į įėjimo parametrą paduodant egzistuojančios grupės siuntimo objektą su pažymėtu ID iš duomenų bazės.	Grąžinamas studentų siuntimo objektų masyvas.
58.	Panaikinti nurodyto studento informaciją. Testavimo atvejo metu paleidžiamas metodas removeStudent(long) į įėjimo parametrą paduodant egzistuojančio studento ID iš duomenų bazės.	Grąžinamas true naikinimo sėkmės atveju, priešingai false.

35 lentelė. OCTestDAO testavimas

Nr.	Testavimo klasė: OCTestDAO (Priklauso operacijų centro posistemei) Testavimo rinkinys: OCTestDAOTest	
	Testavimo atvejis	Laukiamas rezultatas
59.	Sukurti naują testą. Testavimo atvejo metu paleidžiamas metodas createOrUpdateTest(TOTest, TOCourse, List<TOQuestion>) su įėjimo parametrais: • Testo siuntimo objektas su pažymėtu ID atnaujinimo atveju. • Kurso siuntimo objektas su pažymėtu ID, kuriam priklausys testas. • Sąrašas klausimų įeinančių į kuriamą ar pasirinktą testą.	Grąžinamas testo siuntimo objektas su atnaujintą informaciją.
60.	Panaikinti testą. Testavimo atvejo metu paleidžiamas metodas removeTest(TOTest) į įėjimo parametrą	Grąžinamas true naikinimo sėkmės atveju, priešingai false.

	paduodant testo siuntimo objektą su pažymėtu ID.	

3. INTEGRAVIMO TESTAVIMAS

36 lentelė. Klasių (modulių) sąveikos testavimo veiksmai

Nr.	Veiksmų grupė: Integravimo testavimas	
	Testavimo atvejis	Laukiamas rezultatas
61.	Sukuriamas naujas varianto failas (klausimas) su keliais variantais jame. Toliau šis failas naudojamas vienetų testavime.	Nesukeliantys klaidų susijusių su varianto failu veikimas arba normalus duotų operacijų vykdymas.
62.	Egzamino pasirinkimas ir sprendimas studento programoje. Po naujo egzamino sukūrimo (ir kitų veiksmų, kurie reikalingi egzamino sukūrimui) , šį egzaminą sprendžia studentas Studento programoje.	Studentas sėkmingai pasirinko duotą egzaminą ir pradėjo jį spręsti.
63.	Egzamino rezultatų įvertinimas. Po naujo egzamino sukūrimo (ir kitų veiksmų, kurie reikalingi egzamino sukūrimui) , šį egzaminą sprendžia studentas Studento programoje, bei pabaigia testavimą.	Studentas Studento programoje gavo įvertinimą.
64.	Egzaminas vykdomas praktikos režimu Sukuriamas testas (arba egzaminas, bei kiti veiksmai, kurie reikalingi egzamino sukūrimui), įjungiamas praktikos režimas.	Studentui Studento programoje leidžiamas praktikos režimas. Studentui pradėjus testavimą praktikos režimas vyksta įprastas testavimas praktikos režimu.
65.	Egzamino metu Studento programoje leidžiama vaikščioti po klausimus.	Studento programoje vykdant testavimą, studentas gali vaikščioti po testo klausimus.
66.	Egzamino metu Studento programoje neleidžiama vaikščioti po klausimus.	Studento programoje vykdant testavimą, studentas negali vaikščioti po testo klausimus.

4 PRIEDAS. VIENETŲ TESTAVIMO REZULTATAI.

Vienetų testavimo metu įvykdytų testų rezultatų ataskaita, rezultatai yra pateikiami žemiau lentelių pavidalu.

1. RS MODULIS

37 lentelė. RS Factory objekto testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Factory	DAOFactoryTest		0,031
Testavimo atvejis, pastabos	**Laukiamas rezultatas**	**Testo Rezultatas**	**Laikas, s**
testFactory	Objektas ne nulis.	Sėkmingas	0,031

38 lentelė. RS DAO objektų sukurimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
DAO objektų sukurimas	`HibernateDAOFactoryTest`		0,344
Testavimo atvejis, pastabos	**Laukiamas rezultatas**	**Testo Rezultatas**	**Laikas, s**
testGetRSAdministrationDAO, gauti resursų saugyklos dao objektą	Objektas ne nulis.	Sėkmingas	0,078
testGetRvsDAO, gauti studentų atsakymo saugyklos dao objektą.	Objektas ne nulis.	Sėkmingas	0,063
testGetReDAO, resursų vertinimo dao objektas	Objektas ne nulis.	Sėkmingas	0,047
testGetErDAO, vertinimo rezultatų dao objektas.	Objektas ne nulis.	Sėkmingas	0,062
GetLogInInstance, log klasės objektas.	Objektas ne nulis.	Sėkmingas	0,047
TestFactory	Objektas ne nulis.	Sėkmingas	0,047

39 lentelė. RS resurso išsaugojimo testavimas

Data: 2010.11.15		
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas	Bendras laikas, s
Resurso išsaugojimas	`RSDAOstoreItem`	5,828

Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testStoreItem, testuoti vieneto saugojimą, testavimas juodos dėžės metodu Testas užtruko šiek tiek daugiau, nes per pirmą operaciją yra sukuriama duomenų bazės struktūra.	Rezultatas != -1.	Sėkmingas	3,937
testStoreItemPlogIn, testuoti log operaciją, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,219
testStoreItemPlogOut, testuoti log operaciją, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,188
testStoreEmptyItem, saugoti tuščią vienetą, testavimas baltos dežės metodu ir juodos	Testuojama funkcija grąžino -1.	Sėkmingas	0,140
testStoreItemWithInfo, saugoti vienetą su papildoma informacija, testavimas baltos dežės metodu ir juodos dežės metodu	Rezultato apimtis = 1.	Sėkmingas	0,328
testStoreItemAsTOVariant, saugoti vienetą paduodant persiuntimo objektą TOVariant, testavimas baltos dežės metodu ir testavimas juodos dežės metodu	Grąžintas id !=-1 ir apimtyje grąžintas tik vienas rezultatas.	Sėkmingas	0,250

testGetRSAdministrationDAO	Objektas ne nulis.	Sėkmingas	0,125
testGetRvsDAO	Objektas ne nulis.	Sėkmingas	0,125
testGetReDAO	Objektas ne nulis.	Sėkmingas	0,125
testGetErDAO	Objektas ne nulis.	Sėkmingas	0,141
GetLogInInstance	Objektas ne nulis.	Sėkmingas	0,109
TestFactory	Objektas ne nulis.	Sėkmingas	0,141

40 lentelė. RS resurso gavimo testavimas.

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Resurso gavimas	RSDAOGetItem		1,906
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testGetItem, gauti vienetą	byte[] rezultatas ne nulis.	Sėkmingas	0,500
testGetItemlogIn, prisijungimas prie log, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,172
testGetItemlogOut, atsijungimas nuo log, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,203
testGetWrongItem, klaidingo vieneto gavimas, testavimas juodos dežės metodu	Grąžintas nulis.	Sėkmingas	0,250
testGetRSAdministrationDAO, gauti resursų saugyklos dao objektą	Objektas ne nulis.	Sėkmingas	0,156
testGetRvsDAO	Objektas ne nulis.	Sėkmingas	0,110
testGetReDAO	Objektas ne nulis.	Sėkmingas	0,125

testGetErDAO	Objektas ne nulis.	Sėkmingas	0,140
GetLogInInstance	Objektas ne nulis.	Sėkmingas	0,110
TestFactory	Objektas ne nulis.	Sėkmingas	0,140

41 lentelė. RS resurso naikinimo testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Resurso naikinimas	RSDAORemoveItem		1,813
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testRemoveItem, naikinti vienetą	true	Sėkmingas	0,313
testRemoveItemlogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,281
testRemoveItemlogOut, log testavimas, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,297
testRemoveWrongItem, testavimas juodos dežės metodu, klaidingo vieneto naikinimas	Gražintas rezultatas false.		0,141
testGetRSAdministrationDAO, gauti resursų saugyklos dao objektą	Objektas ne nulis.	Sėkmingas	0,140
testGetRvsDAO	Objektas ne nulis.	Sėkmingas	0,110
testGetReDAO	Objektas ne nulis.	Sėkmingas	0,156
testGetErDAO	Objektas ne nulis.	Sėkmingas	0,125
GetLogInInstance	Objektas ne nulis.	Sėkmingas	0,125
TestFactory	Objektas ne nulis.	Sėkmingas	0,125

42 lentelė. RS scenarijų gavimo testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Scenarijų gavimas	RSDAOgetAvailableScenarios		1,250
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testGetEvalScenarios, testavimas juodos dežės metodu, gauti visus vieno klausumo vertinimo scenarijus	Rezultato apimtys > 0.	Sėkmingas	0,125
testGetEvalByGroupScenarios, testavimas juodos dežės metodu, gauti visus galutinio įvertinimo scenarijus.	Rezultato apimtys > 0.	Sėkmingas	0,203
testGetPrepareScenarios, testavimas juodos dežės metodu, gauti visus prieinamus paruošimo scenarijus.	Rezultato apimtys > 0.	Sėkmingas	0,156
testGetRSAdministrationDAO, gauti resursų saugyklos dao objektą	Objektas ne nulis.	Sėkmingas	0,141
testGetRvsDAO	Objektas ne nulis.	Sėkmingas	0,125
testGetReDAO	Objektas ne nulis.	Sėkmingas	0,125
testGetErDAO	Objektas ne nulis.	Sėkmingas	0,125
GetLogInInstance	Objektas ne nulis.	Sėkmingas	0,125
TestFactory	Objektas ne nulis..	Sėkmingas	0,125

43 lentelė. RS resursų (variantų) paieška

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Resursų (variantų) paieška	RSDAOsearchVariants		1,547
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s

testSearchKey, testavimas juodos ir baltos dežės metodu, ieškoti pagal raktą	Bus rastas 1 rezultatas.	Sėkmingas	0,281
testSearchInfo, testavimas juodos ir baltos dežės metodu, ieškoti pagal papildomą informaciją.	Bus rastas 1 rezultatas.	Sėkmingas	0,250
testSearchAll, testavimas juodos dežės metodu, testavimas baltos dežės metodu, ieškoti pagal abejus parametrus.	Nebus rasta nei vieno rezultato.	Sėkmingas	0,234
testGetRSAdministrationDAO, gauti resursų saugyklos dao objektą	Objektas ne nulis.	Sėkmingas	0,141
testGetRvsDAO	Objektas ne nulis.	Sėkmingas	0,141
testGetReDAO	Objektas ne nulis.	Sėkmingas	0,109
testGetErDAO	Objektas ne nulis.	Sėkmingas	0,141
GetLogInInstance	Objektas ne nulis.	Sėkmingas	0,125
TestFactory	Objektas ne nulis.	Sėkmingas	0,125

2. OC MODULIS

Buvo atliktas dalinis OC modulio testavimas.

Buvo testuojamos šios funkcijos:

- Operacijų kurios susijusios su kursu automatinis testavimas, buvo taikomos baltos ir juodos dėžės testavimo strategijos.
- Administratorių objektų automatinis testavimas

Toliau pateikiamos operacijų susijusių su kursu automatinio vienetų testavimo rezultatai.

44 lentelė. Kurso sukurimo arba atnaujinimo testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Kurso sukurimas arba atnaujinimas	OCCOURSEcreateOrUpdate		16,250
Testavimo atvejis,	Laukiamas rezultatas	Testo Rezultatas	Laikas, s

pastabos			
testCreateCourse, testavimas juodos dežės metodu	Grąžintas neneigiamas ir nenulinis sukurto objekto identifikatorius.	Sėkmingas	14,531
testUpdateCourse, testavimas juodos dežės metodu	Grąžintas neneigiamas ir nenulinis sukurto objekto identifikatorius.	Sėkmingas	0,360
testCreateUpdateCourseLogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,281
testCreateUpdateCourseLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,297
testCreateEmptyCourse	Tikimasi gauti HibernateOCAdministratorDAOException išimtį.	Sėkmingas	0,265
testGetOCAdministrationDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,250
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,266

45 lentelė. Kurso priskyrimo administratoriui testavima

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Kurso priskyrimas administratoriui	OCCOURSEaddCourseToAdministrator		3,328
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testAddNewCourseToDi	Sėkmingai įvykdytos	Sėkmingas	0,734

fferentAdministrator, testuojamas skirtingų administratorių priskyrimas kursui	priskyrimo operacijos.		
testAddNewCourseToDifferentAdministratorLogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,328
testAddNewCourseToDifferentAdministratorLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,297
testAddWrongCourseToValidAdministrator, testavimas juodos dežės metodu, bandoma priskirti egzistuojantį administratorių prie neegzistuojančio kurso.	Nesėkiminga priskyrimo operacija.	Sėkmingas	0,282
testAddWrongAdminToValidCourse, testavimas juodos dežės metodu, prie egzistuojančio kurso bandoma priskirti neegzistuojantį administratorių (neteisingas id)	Nesėkminga prisikyrimo operacija.	Sėkmingas	0,281
testAddWrongAdminToWrongCourse, testavimas juodos dežės metodu, testuojamas priskyrimo operacijos veikimas, kai paduoti	Nesėkminga priskyrimo operacija.	Sėkmingas	0,265

neteisingi priskyrimo duomenys (neegzistuojančių objektų id)			
testAddNewCourseToSameAdministrator, testavimas juodos dežės metodu, testuojama priskyrimo operacija, kai tam pačiam administratoriui priskiriama daugiau vieno kurso, kai kursas turi daugiau negu 1 administratorių	Visos priskyrimo operacijos sėkmingos.	Sėkmingas	0,579
testGetOCAdministrationDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,265
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,297

46 lentelė. Kurso gavimo pagal jo id testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Kurso gavimas pagal jo identifikatorių	OCCOURSEgetCourseById		1,391
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testGetCourse, sukuriami keli nauji kursai, bandoma gauti kiekvieną pagal jo id.	Gauto kurso id nenulis.	Sėkmingas	0,328
testGetCourseLogIn, testavimas baltos dežės	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai	Sėkmingas	0,266

metodu	testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.		
testGetCourseLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,265
testGetOCAdministratioNDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,266
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,266

47 lentelė. Kurso paieškos pagal jo vardą testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Vykdyti kurso paiešką pagal jo vardą	OCCOURSEsearchCourseByName		1,422
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testGetCourseListByName, ieškoma kurso kurio pavadinimas bet kokia pradžia + fizika + bet koks galas.	Pagal nurodytą paieškos raktą turi būti grąžinta 2 rezultatai, tikrinamas grąžinto paieškos rezultato masyvo apimtis.	Sėkmingas	0,343
testGetCourseListByNameLogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,266
testGetCourseListByNameLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo	Sėkmingas	0,266

	log modulyje darbo pabaigą, priešingų atveju ne		
testGetOCAdministrationDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,265
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,282

48 lentelė. Visų kursų gavimo testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Gauti visą kursų sąrašą	OCCOURSEgetAllCourseList		1,375
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testGetAllCourses	Grąžintas 1 ir daugiau rezultatų.	Sėkmingas	0,281
testGetAllCoursesLogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,281
testGetAllCoursesLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,281
testGetOCAdministrationDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,266
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,266

49 lentelė. Kurso gavimas pagal administratoriaus id testavimas

Data: 2010.11.15

Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Gauti kursus pagal administratoriaus identifikatorių.	OCCOURSEgetCoursesByAdministratorId		1,390
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testGetCoursesByAdministratorId, bandoma gauti visus kursus, kurie priklauso administratoriui, kuriuo id yra 5.	Grąžinti 2 rezultatai.	Sėkmingas	0,312
testGetCoursesByAdministratorIdLogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,281
testGetCoursesByAdministratorIdLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,266
testGetOCAdministrationDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,266
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,265

50 lentelė. Administratoriaus atjungimo operacijos nuo kurso testavimas

Data: 2010.11.15		
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas	Bendras laikas, s
Pašalinti administratorių nuo kurso	OCCOURSEremoveCourseFromAdministrator	1,485

Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testremoveCourseFrom Administrator,administr atorius su id 5 šalinamas iš visų kursų prie kurių jis priskirtas.	Administratorius, kurio id yra 5 neturi jam priskirtų kursų.	Sėkmingas	0,422
testremoveCourseFrom AdministratorLogIn, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.	Sėkmingas	0,266
testremoveCourseFrom AdministratorLogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,265
testGetOCAdministratio nDAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,266
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,266

51 lentelė. Kurso šalinimo testavimas

Data: 2010.11.15			
Testuojamas programavimo vienetas	Testavimo rinkinio pavadinimas		Bendras laikas, s
Kurso šalinimas	OCCOURSEremoveCourse		2,140
Testavimo atvejis, pastabos	Laukiamas rezultatas	Testo Rezultatas	Laikas, s
testRemoveEmptyCourse, tuščio kurso naikinimas (kai kursas neturi savyje kitų objektų arba ryšių)	Sėkminga operacijos pabaiga.	Sėkmingas	0,375
testRemoveEmptyCourse	Jei nustatymuose įjungtas	Sėkmingas	0,265

LogIn. testavimas baltos dežės metodu	„debug“ tipo sekimas, tai testuojamas metodas pažymėjo darbo pradžią log modulyje, priešingų atveju ne.		
testRemoveEmptyCourse LogOut, testavimas baltos dežės metodu	Jei nustatymuose įjungtas „debug“ tipo sekimas, tai testuojamas metodas pažymėjo log modulyje darbo pabaigą, priešingų atveju ne.	Sėkmingas	0,281
testGetDeletedOrWrongCourse, tikrinama ar sunaikintas kursas egzistuoja duomenų bazėje.	Pagal nurodyta id, grąžintas nulis.	Sėkmingas	0,266
testRemoveUnEmptyCourse, bandymas ištrinti netuščią kursą (kursas turi savyje grupių)	Nesėkmingas naikinimas.	Sėkmingas.	0,406
testGetOCAdministration DAOs - Sukuriami visų OC posistemės DAO objektai	Nei vienas iš sukurtų objektų nėra nulinis.	Sėkmingas	0,266
testFactory	Fabriko objejtas ne nulis.	Sėkmingas	0,281

5 PRIEDAS. FUNKCINIAI REIKALAVIMAI

Funkciniai reikalavimai kuriamiems moduliams yra sudaryti pagal Volere šabloną.

52 lentelė. Bendro, kelių administratorių valdomo kurso realizacijos reikalavimas

Reikalavimas: *1*	**Reikalavimo tipas:**	1	**Panaudojimo atvejis:**	*2, 4*
Aprašymas:	Turi būti realizuota galimybė, kad kursui galima būtų priskirti kelis administratorius. vienas administratorius galėtų turėti daugiau nei vieną kursą. Tokiu būdu tarp kurso ir administratoriaus būtu organizuotas duomenų ryšis daug-su-daug.			
Pagrindimas:	Dažnai pasitaiko, kad 1 kursą veda keli instruktoriai/dėstytojai, tokiu būdu dažnai atsitinka, kai du vartotojai pradeda naudotis viena prisijungimo paskyra.			
Šaltinis:	Ruslanas Sobolevas			
Tikrinimo kriterijus:	Nėra			
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:**	*4*	
Priklausomybės:	*4*	**Konfliktai:**	Nėra	
Papildoma medžiaga:	Nėra			
Istorija:	Užregistruotas 2010.03.01			

53 lentelė. Apsaugos nuo šiuklių atsiradimo kurse realizacijo sreikalavimas

Reikalavimas: *2*	**Reikalavimo tipas:**	1	**Panaudojimo atvejis:**	*2, 4*
Aprašymas:	Naikinant administratorių turi būti užtikrinama, kad kursas neliks be administratorių.			
Pagrindimas:	Jei kursas liks be administratorių, tai gali kauptis šiukšlės, kurios priklauso kursui (nenaudojami resursai).			
Šaltinis:	Ruslanas Sobolevas			
Tikrinimo kriterijus:	Tikrinamas administruojančių kursą administratorių kiekis.			
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:**	*4*	
Priklausomybės:	*4*	**Konfliktai:**	Nėra	
Papildoma medžiaga:	Nėra			
Istorija:	Užregistruotas 2010.03.01			

54 lentelė. Klausimų varianto scenarijaus reikalavimas

Reikalavimas: *3*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *8, 9, 10*
Aprašymas:	Administratorius gali nustatyti egzamine scenarijų, kuris nurodis tvarką, pagal kurią egzamino metu pateikiami klausimų variantai.	
Pagrindimas:	Tai leistų lanksčiau organizuoti klausimo variantų pateikimo tvarką.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Įvairūs objektai, priklausomai nuo klausimų variantų organizavimo ir pasirinkto scenarijaus vykdomos veiksmų sekos.	
Užsakovo tenkinimas:	*5*	**Užsakovo netenkinimas:** *5*
Priklausomybės:	*Priklausomai nuo scenarijaus.*	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

55 lentelė. RS ir OC modulių atskirymo reikalavimas

Reikalavimas: *4*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *6*
Aprašymas:	Resursų saugykla veiktia atskirai nuo pagrindinio administravimo serverio. Į šią saugyklą iškeliami tokie mokymosi resursai kaip klausimai su variantais, resursų vertinimas, paruošimas.	
Pagrindimas:	Tai padidins mokymosi resursų pakartotino panaudojimo galimybes, praplės resursų tipus, paskirstys apkrovimą, sudarys sąlygas sistemos plėtojimui.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Nėra	
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:** *2*
Priklausomybės:	Nėra	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

56 lentelė. Keleto RS modulių vienam administratoriui prisikyrimo galimybės reikalavimas

Reikalavimas: *5*	**Reikalavimo tipas:**	1	**Panaudojimo atvejis:**	*6, 4, 2*
Aprašymas:	Vienam administratoriui gali priklausyti keletas resursų saugyklų, bei keli administratoriai vienu metu gali naudotis viena resursų saugykla.			
Pagrindimas:	Administratorių patogumui kiekvienas galės naudoti savo resursų saugyklą, kas užtikrins jo resursų atskiriamumą ir nepriklausomumą nuo įvairių aplinkybių. Pavyzdžiui administratorius gali pageidauti saugykla pakurti savo kompiuteryje arba serveryje.			
Šaltinis:	Ruslanas Sobolevas			
Tikrinimo kriterijus:	Resursų saugyklos administratorius, resursų saugykla, administratoriui priklausantys kursai.			
Užsakovo tenkinimas:	*5*	**Užsakovo netenkinimas:**	*3*	
Priklausomybės:	*4,2*	**Konfliktai:**		
Papildoma medžiaga:	Nėra			
Istorija:	Užregistruotas 2010.03.01			

57 lentelė. Resursų įvertinimo scenarijaus reikalavimas

Reikalavimas: *6*	**Reikalavimo tipas:**	1	**Panaudojimo atvejis:**	*16*
Aprašymas:	Resursų įvertinimą organizuoja įvertinimo scenarijai			
Pagrindimas:	Lankstesniam vertinimui organizuoti. Gali būti naudojami svoriai ir pan.			
Šaltinis:	Ruslanas Sobolevas			
Tikrinimo kriterijus:	Scenarijaus egzistavimas ir jo pavadinimas.			
Užsakovo tenkinimas:	*5*	**Užsakovo netenkinimas:**	*3*	
Priklausomybės:	Nėra	**Konfliktai:**	Nėra	
Papildoma medžiaga:	Nėra			
Istorija:	Užregistruotas 2010.03.01			

58 lentelė. Paruošimo scenarijų funkcijos reikalavimas

Reikalavimas: *7*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *14*
Aprašymas:	Resursų paruošima organizuoja mokymosi resursų paruošimo scenarijai.	
Pagrindimas:	Sukoncentruoti įvairius konvertavimo, resursų paruošimo įrankius vienoje vietoje.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Scenarijaus egzistavimas ir jo pavadinimas.	
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:** *3*
Priklausomybės:		**Konfliktai:**
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

59 lentelė. El. pašto išlaikimo scenarijaus RS reikalavimas

Reikalavimas: *8*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *15*
Aprašymas:	El. pašto siuntimo modulis. Gali būti naudojamas specialus scenarijus, kuris savo ruožtu naudoja pašto siuntimo komponentus.	
Pagrindimas:	Studentas galėtų gauti savo testo sprendimo išklotinę į el. paštą. Administratorius/instruktorius galėtų nusiųsti studentų rezultatus el. paštu.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Nėra	
Užsakovo tenkinimas:	*1*	**Užsakovo netenkinimas:** *0*
Priklausomybės:	*7*	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

60 lentelė. Klausimų svorių palaikymo galimybės reikalavimas

Reikalavimas: *9*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *7*
Aprašymas:	Klausimų svorių panaudojimas ir vertinimas atsižvelgiant į jas. Realizuoti galima būtu specialų scenarijų, kuris atitinkamai kontroliuotų vertinimo mechanizmą, bei įvertinimą surištų su klausimų svoriais.	
Pagrindimas:	Vienas didžiausių apribojimų lanksčiam studento žinių testavimui dabartinėje testavimo sistemos versijoje.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Nėra	
Užsakovo tenkinimas:	*5*	**Užsakovo netenkinimas:** *3*
Priklausomybės:	Nėra	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

61 lentelė. Privačių resursų kurimo galimybės reikalavimas

Reikalavimas: *10*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *6*
Aprašymas:	Sudaryti galimybes dėstytojui apriboti resursų naudojimą nuo kito dėstytojo, kuris neturi teisės valdyti atitinkamų kursų. Organizuoti atitinkamą resursų atskirimą arba apsaugą.	
Pagrindimas:		
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Dėstytojo objektas.	
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:** *3*
Priklausomybės:	*11, 12*	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

62 lentelė. Lanksčios mokymosi resursų paieškos reikalavimas

Reikalavimas: *11*	**Reikalavimo_tipas:** 1	**Panaudojimo_atvejis:** *11*
Aprašymas:	Organizuoti lanksčią resursų paiešką resursų saugykloje	
Pagrindimas:	Užtikrinti lengvą, lanksčią paiešką resursų saugyklose.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo_kriterijus:	Nėra	
Užsakovo_tenkinimas:	*5*	**Užsakovo_netenkinimas:** *3*
Priklausomybės:	*15, 12, 10, 9, 8, 6, 5, 4, 2, 1.*	**Konfliktai:** Nėra
Papildoma_medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

63 lentelė. Aktyvių egzaminų saugojimo reikalavimas

Reikalavimas: *12*	**Reikalavimo_tipas:** 1	**Panaudojimo_atvejis:** *8, 9*
Aprašymas:	Turi būti sudarytos galimybės stebėti vykdomus egzaminus	
Pagrindimas:	Administratoriui turi būti sudarytos galimybės stebėti einamu metu vykdomus testus ir/arba egzaminus tam, kad įvertinti testo arba egzamino būseną bei studentų aktyvumą.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo_kriterijus:	Tikrinamas dabartinis laikas ir egzamino pradžios/pabaigos laikas. Jei einamas laiko tarpas atitinka egzamino laiką, egzaminas atvaizduojamas kaip vykdomas, kartu gali būti pateikiama informacija apie studentus (15 reikalavimas).	
Užsakovo_tenkinimas:	*3*	**Užsakovo_netenkinimas:** *5*
Priklausomybės:	*1*	**Konfliktai:** Nėra
Papildoma_medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

64 lentelė. Egzaminų statistikos reikalavimas

Reikalavimas: *13*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *8,9*
Aprašymas:	Turi būti sudarytos galimybės stebėti statistiką egzaminų, kiek studentų pasijungė, kiek laikė ir pan.	
Pagrindimas:	Galimybė administratoriui įvertinti testo arba egzamino eigą.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Nėra	
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:** *3*
Priklausomybės:	9	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

65 lentelė. Egzaminų režimo reiakalavimas

Reikalavimas: *14*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *9*
Aprašymas:	Užtikrinti atskirimą egzaminų nuo praktikos, kai testavimas vykdomas pažymiui ir treniruotės režimu.	
Pagrindimas:		
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Nėra	
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:** *3*
Priklausomybės:	*10*	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

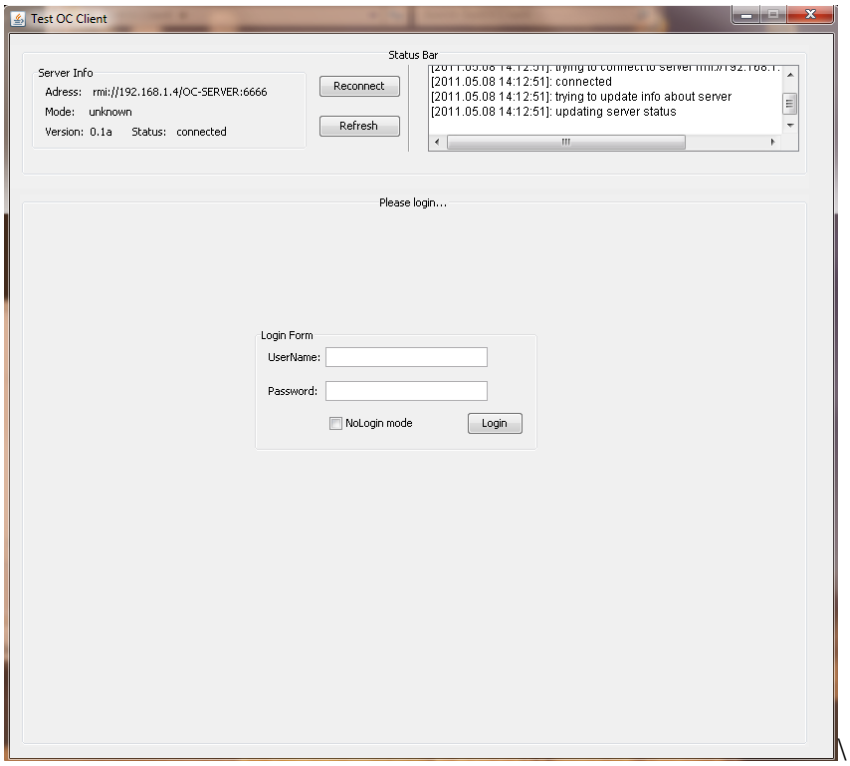
66 lentelė. Korektiškų duomenų įvedimo reikalavimas

Reikalavimas: *15*	**Reikalavimo tipas:** 1	**Panaudojimo atvejis:** *Visi*
Aprašymas:	Užtikrinti teisingų ir korektiškų duomenų įvedimą į sistemą.	
Pagrindimas:	Korektiškų duomenų įvedimas būtinas teisingam ir pastoviam sistemos veikimui.	
Šaltinis:	Ruslanas Sobolevas	
Tikrinimo kriterijus:	Nėra	
Užsakovo tenkinimas:	*3*	**Užsakovo netenkinimas:** *3*
Priklausomybės:	*Visi.*	**Konfliktai:** Nėra
Papildoma medžiaga:	Nėra	
Istorija:	Užregistruotas 2010.03.01	

6 PRIEDAS. PRIĖMIMO TESTAVIMO 2 DALIS

Testuojama sukurta PĮ: Operacijų centras, resursų saugykla.

Testavimas vykdomas, panaudojant testinę grafinę sąsają - klientą, kuri sukurta specialiai šiam testavimui, todėl sąsajos tikslai išskirtinai sukurtų modulių testavimas, o ne draugiškumas ir suprantamumas vartotojui. Sąsajos kalba yra anglų. Šios sąsajos langas yra parodytas žemiau 63 pav.



63 pav. Testinio kliento prisijungimo lango vaizdas

Resursų saugykla yra paleista, jos ip: 192.168.1.3.

Operacijų centras yra paleistas, jo ip: 192.168.1.4.

Abu serveriai yra vidiniame tinkle, už NAT, duomenų bazė yra MySQL, kuri yra paleista dedikuotame kompiuteryje ip - 193.219.159.195. Vidinio tinklo pralaidumas 100Mb/s. Duomenų bazė ir TestTool serveriai yra paleisti skirtingose potinklyse, kurios aptarnauja skirtingi paslaugų tiekėjai. Tokios testavimo sąlygos yra parinktos, kadangi artimos realiom, kai moduliai bus naudojami užsakovo įmonėje.

Operacijų centro duomenų bazė yra ne tuščia ir užpildyta tam tikromis duomenimis. Yra sukurtas administratorius ir super-administratorius, informacija iš duomenų bazės parodyta 64 pav. žemiau.

ADMIN_ID	NAME	PASSWORD	SURNAME	USERNAME	SUPER_ADM	UUID
1	Ruslanas	16b15fe4a5b173f44110e22e2843116	Sobolevas	ruslanas	0	0bbbaf20-22fb-11e0-9177-1c6f6546675f
2	admin	21232f297a57a5a743894ae4a801fc3	admin	admin	1	cea8b430-2704-11e0-9c1f-1c6f6546675f

64 pav. Administratorių informacija esanti MySQL duomenų bazėje

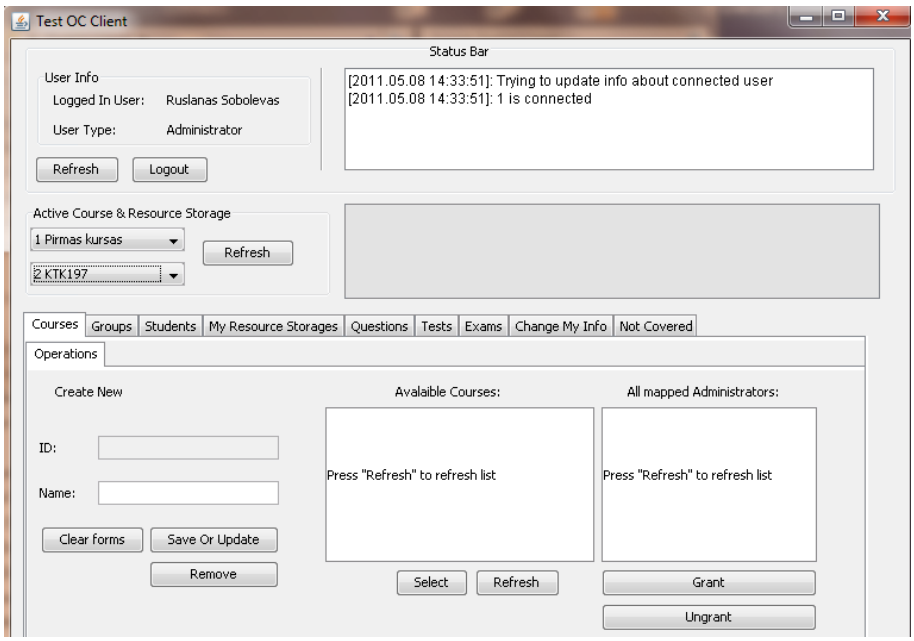
Kaip matome slaptažodis yra užkoduotas md5, kas atitinka saugumo reikalavimus keliamai sistemai.

Taip pat yra sukurti testiniai kursai, studentai, testai, egzaminai ir kt.

Šio testavimo metų bus išbandyti žemiau išvardinti veiksmai:

1. Vartotojų (administratoriaus) prisijungimas.
2. Resursų saugyklų pridėjimas ir jų turinio apžvalga.
3. Esamo turinio šalinimas
4. Naujo turinio įkėlimas
5. Naujo studento įkėlimas ir esamo šalinimas.
6. Naujos grupės sukūrimas ir esamos šalinimas.
7. Klausimų, testų, egzaminų kūrimas.
8. Prieinamų scenarijų apžvalga.
9. Savo informacijos pakeitimas.
10. Naujo administratorius sukūrimas ir priskyrimas esamo bei naujai sukurto kurso.
11. Sistemos darbo stebėjimo failų patikra.

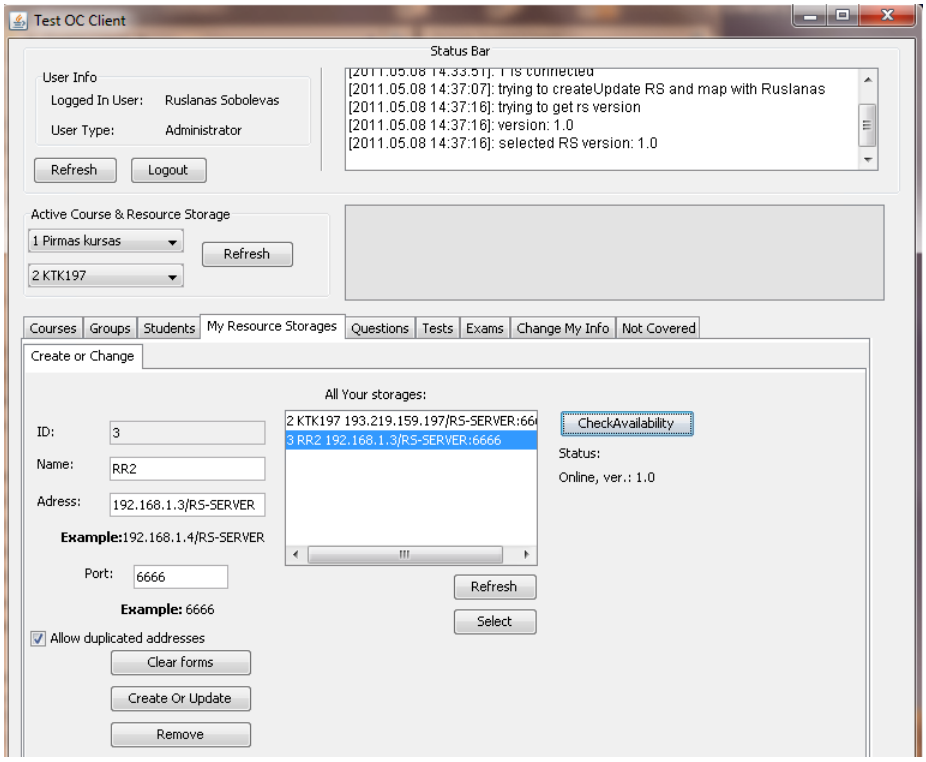
Panaudojant aukščiau paminėta testinę programa prisijungsime prie operacijų centro serverio (63 pav). Sėkmingai prisijungus matome pasiekiamus veiksmus, 65 pav.



65 pav. Testinio kliento lango vaizdas po administratoriaus prisijungimo

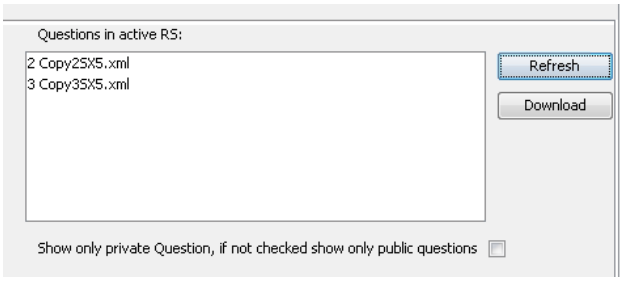
Toliau testuosime naujos resursų saugyklos pridėjimas.

Pridėsime šiam vartotojui mūsų paleista resursų saugyklą (192.168.1.3), pavadinsime ją RR2. Pažiūrėjus į statuso langą, matome, kad nurodyta resursų saugykla sėkmingai sukurta, papildomai patikriname ar nurodyta resursų yra saugykla, gauname teigiama atsakymą, tai įrodanti programos lango kopija yra pateikta 66 pav.



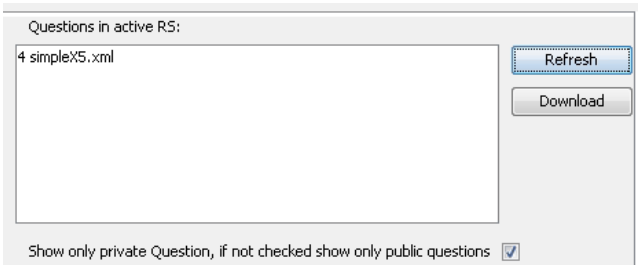
66 pav. Esamų pakeitimo arba naujų resursų saugyklų pridėjimo kortelės vaizdas testiniame kliente

Pasirenkame šį saugyklą kaip aktyvią, ir pažiūrime jos turinį, pasirenkant Questions kortelę (72 pav.), matome 2 viešus klausimų resursus (67 pav.)



67 pav. Viešų resursų sąrašas, esančių pasirinktame resursų saugykloje, vaizdas iš testinio kliento

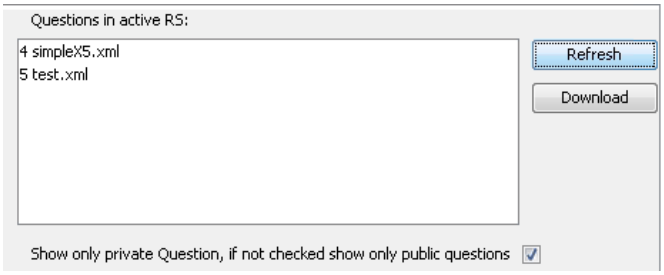
Pažymėjus varnelę, pamatysime privačius resursus, lango dalies kopija parodyta 68 paveiksle.



68 pav. Privačių resursų sąrašas, esančių pasirinktame resursų saugykloje, vaizdas iš testinio kliento

Matome, kad šiame resursų saugykloje yra įkelti 2 vieši resursai ir vienas privatus. Toliau galime parsiųsti šiuos resursus, tam, kad atidaryti Autoriaus programoje. Parsiuntus resursą ir atidarius su Autoriaus programa, matome, kad resursas nepakito, kas įrodo saugojimo funkcijų veikimo korektiškumą. Taip pat dabar galime pereiti prie klausimų ir testų sukūrimo arba įkelti naujus resursus.

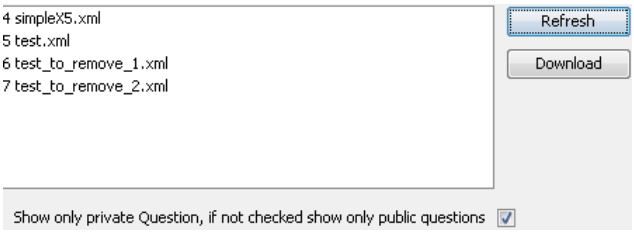
Įkelsime savo naują resursą, padarysime jį privačiu ir priskirsime jam pavadinimą „test.xml“. Atnaujiname privačių resursų sąrašą, langa dalie kopija parodyta 69 pav.



69 pav. Viešų resursų sąrašas, esančių pasirinktame resursų saugykloje, vaizdas iš testinio kliento

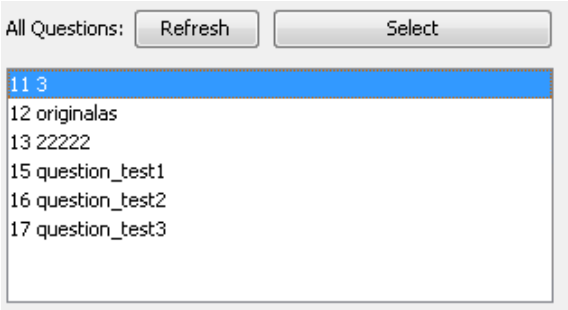
Matome atsiradusį mūsų resursą. Įkelsime dar pora resursų, kuriuos pavadinime test_to_remove_1.xml ir test_to_remove_2.xml, kurios vėliau pabandysime pašalinti.

Resursų lange matome (70 pav.), atsiradusius naujus resursus.



70 pav. Privačių resursų sąrašas, esančių pasirinktame resursų saugykloje, vaizdas iš testinio kliento

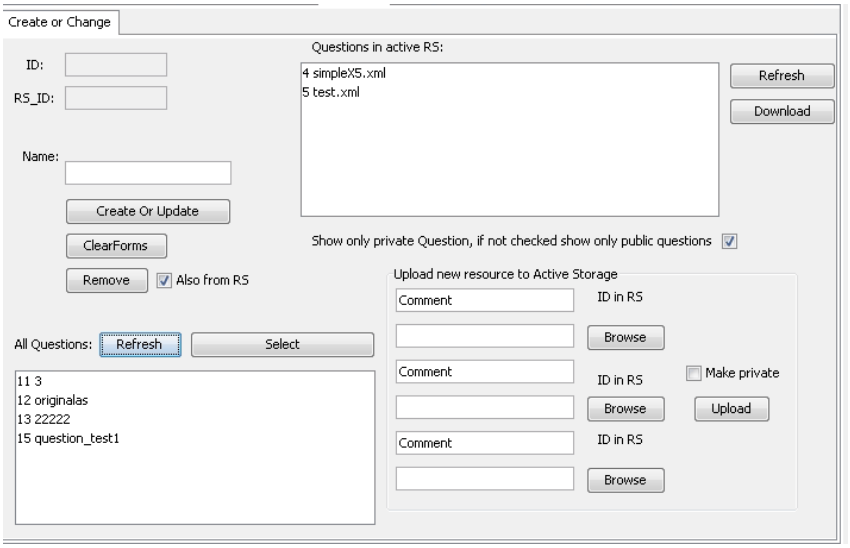
Pabandome sukurti naujus klausimus įtraukiant test.xml. bei naujai įkeltus 2 resursus, sukuriame tris klausimus – question_test1, question_test2, question_test3.



71 pav. Klausimų sąrašo vaizdas testiniame kliente

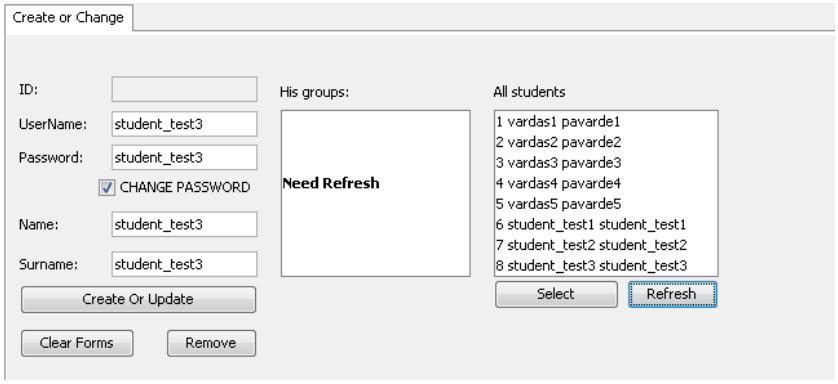
Matome sėkmingai atsiradusius naujus klausimus (71 pav.), peržvelgus kiekvieno sukurto klausimo resursų įsitikiname, kad nurodytus klausimus iš tikrųjų sudaro pasirinkti resursai.

Dabar panaikinsime questiom_test2 ir question_test3, kartu šalinant ir susijusius resursus iš saugyklos, ekrane stebime pakitimus, klausimų sąraše ir privačių resursų sąraše atitinkamai nematome nurodytų klausimų ir resursų, 72 pav.



72 pav. Operacijų centro klausimų surišimas su resursų saugykloje esančiais resursais

Dabar pereisime prie studentų kortelės, matome sukurta studentų. Papildomai pridėsime dar keletą naujų – student_test1, student_test2, student_test3 (vardo, pavardes ir slaptažodžių informacija sutampa). , tai įrodanti lango dalie kopija yra parodyta žemiau.



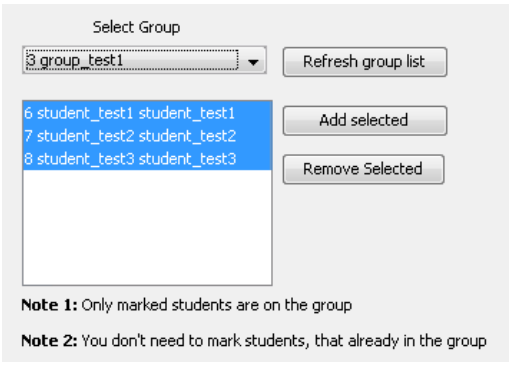
73 pav. Esamo arba naujo studento informacijos įvedimo kortelė testiniame kliente

Pašalinsime esamus studentus, palikime tik tuos kuriuos sukūrėme dabar. Testinio kliento operacijų stebėjimo žurnalas parodo, kad nurodytos operacijos sėkmingai atliktos:

[2011.05.08 15:24:47]: removing of student was successful
[2011.05.08 15:24:51]: removing of student was successful
[2011.05.08 15:24:54]: removing of student was successful
[2011.05.08 15:24:57]: removing of student was successful
[2011.05.08 15:24:59]: removing of student was successful

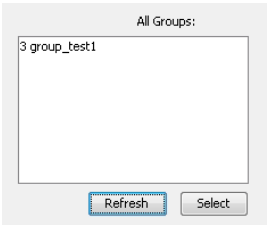
Tą pati taip at įrodo pasikeitusi studentų informacija, bei duomenų bazės vaizdas. Toliau pereiname prie grupių valdymo kortelės.

Sukuriame nauja grupę – group_test1, į kurią priskiriame ankstesniame žingsnyje sukurtus studentus. 74 paveiksle matome, kad šie veiksmai sėkmingai buvo atlikti.



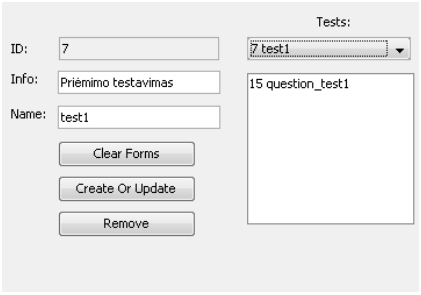
74 pav. Studentų priskyrimas į grupes testiniame kliente

Pašaliname kitas grupes. Grupių valdymo kortelėje, grupių sąraše matome tik mūsų sukurtą grupę – 75 pav.



75 pav. Grupių sąrašas testiniame kliente, lango dalis

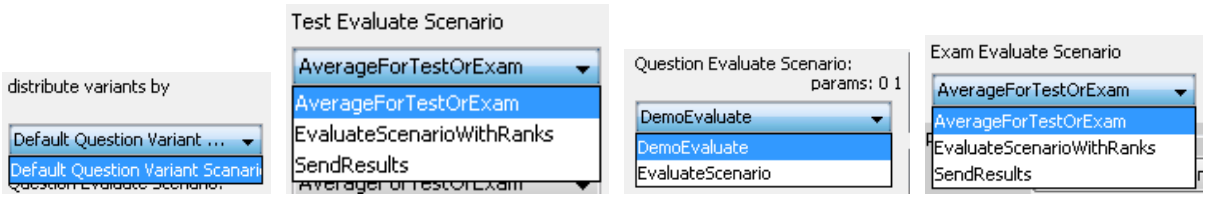
Toliau pereiname prie testų kūrimo kortelės ir sukuriame naują testą – test1, priskiriant šiame testavime sukurtą klausimą – question_test1. 76 pav. matome sėkmingai atliktą operaciją.



76 pav. Testo sukūrimas arba esamo pakeitimas testiniame kliente

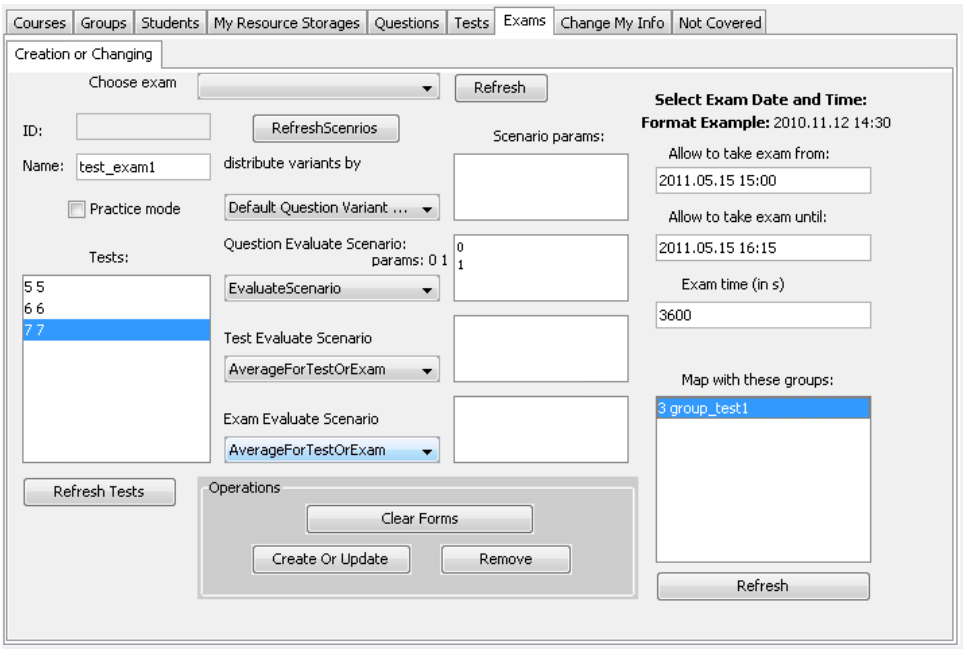
Pereiname prie egzaminų kūrimo kortelės. Matome įvairius egzamino nustatymus. Atnaujiname prieinamų scenarijų, testų ir grupių sąrašą.

77 paveiksle žemiau galima matyti, kad pasirinkta resursų saugykla sėkmingai grąžino scenarijų sąrašus.



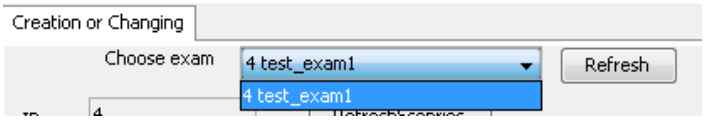
77 pav. RS prieinamų scenarijų pasirinkimas egzamino pakėtimo arba sukūrimo metu

Testinio kliento naujo egzamino nustatymų langas parodytas žemiau 78 pav.



78 pav. Naujo egzamino sukūrimo arba pakeitimo kortelė testiniame kliente

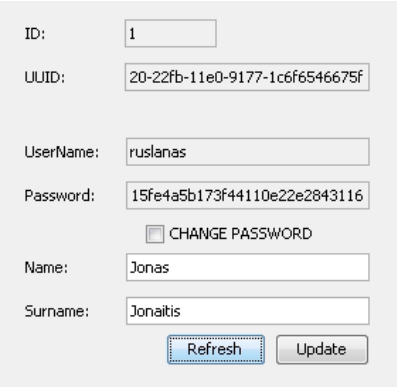
Paliekame nustatymus pagal nutylėjimą, tuščius laukus užpildome pavyzdinę informaciją, kaip testą ir grupę pasirenkame aukščiau sukurtus vienetus. 79 paveiksle matome, kad egzaminas sėkmingai yra sukurtas.



79 pav. Egzaminų sąrašas testiniame kliente

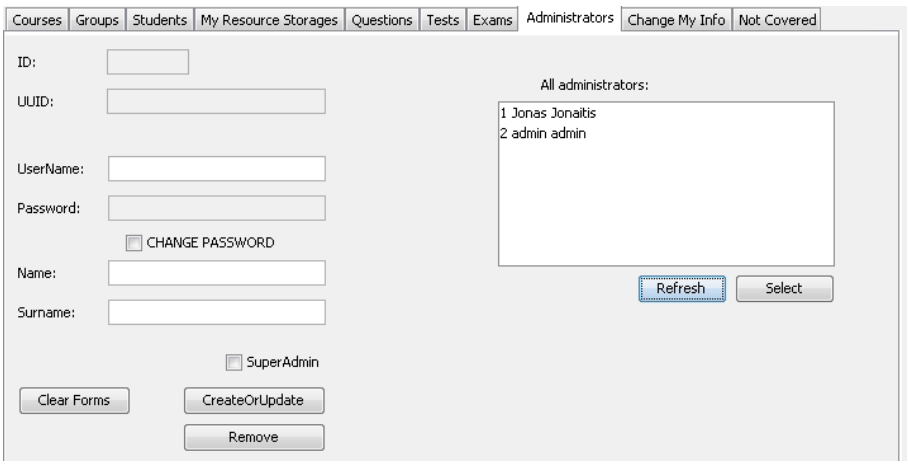
Pabandysime pakeisti savo informaciją, pereiname prie testavimo kliento savo informacijos pakeitimo kortelės „Change My Info“

Vardą ir pavardę pakeičiame į „Jonas Jonaitis“, atnaujiname informaciją. 80 pav. rodo pakitusią asmeninę informaciją.



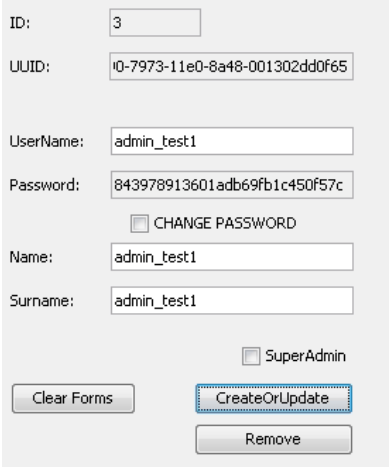
80 pav. Testinio kliento asmeninės informacijos pakeitimo kortelė

Dabar sukursime naują administratorių ir kursą jam. Tam turime atsijungti nuo administratoriaus paskyros ir persijungti su super administratoriaus paskyra, kurio vartotojo informacija yra admin/admin. Šis vartotojas jau buvo sukurtas prieš testavimą, kitu atveju jis gali turėti kitą vartotojo vardą ir slaptažodį. Prisijungus super administratoriaus teisėmis matome papildomą kortelę „Administrators“, kur galime kurti redaguoti esamus arba kurti naujus administratorius, testinio kliento administratorių valdymo kortelė parodyta 81 pav.



81 pav. Testinio kliento administratorių valdymo kortelės vaizdas

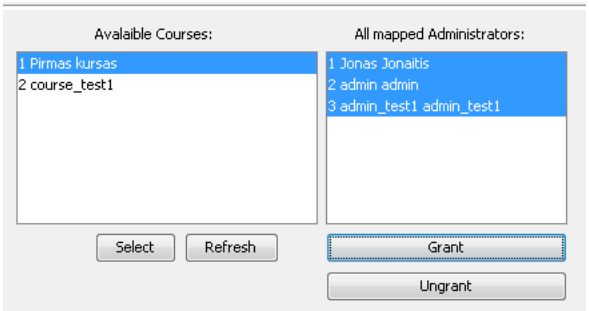
Sukuriame naują administratorių, visų laukų vartotojo informaciją bus admin_test1, tai bus paprastas administratorius, kuris neturi teisės valdyti kitų administratorių informacijos ir kursų sąryšių. Atsinaujinusios administratoriaus sukūrimo formos įrodo, kad sukurtas naujas administratorius – *82* pav.



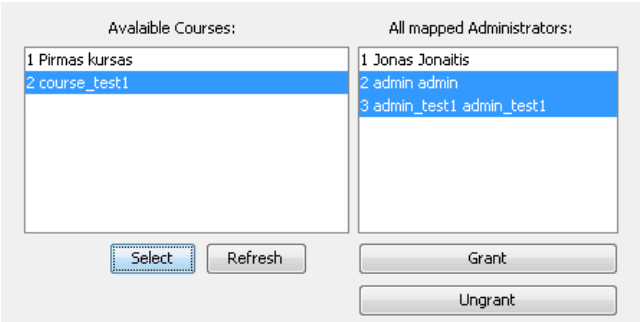
82 pav. Testinio kliento administratoriaus informacijos redagavimo formos

Dabar pereiname prie kurso valdymo kortelės, kur surišame šį sukurtą administratorių su esamu kursu, kur buvo vykdomi aukščiau minėtieji veiksmai, bei sukuriame papildomai naują kursą course_test1. Primename, kad po šio veiksmo sistemoje esantys kursas „Pirmas kursas“ turės du administratorius ir vieną esamą super-administratorių.

83 pav. rodo, kad esamam kursui yra priskirti trys administratoriai, tuo tarpu naujai sukurtam kursui yra priskirti tik du administratoriai, 84 pav.

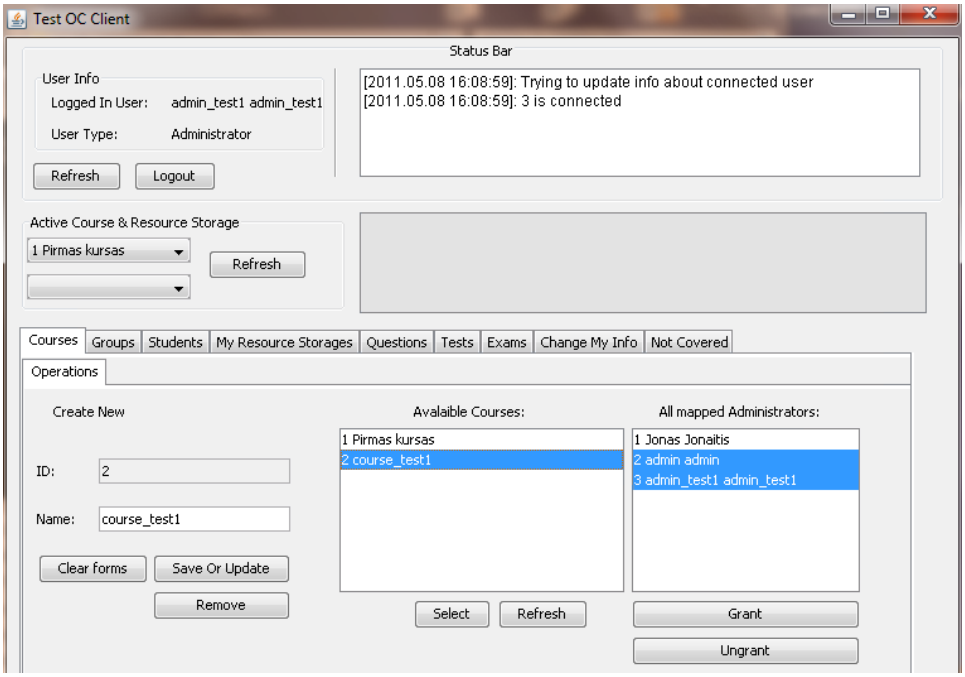


83 Kursų ir administratorių surišimas testiniame kliente



84 pav. Kursų ir administratorių surišimas testiniame kliente

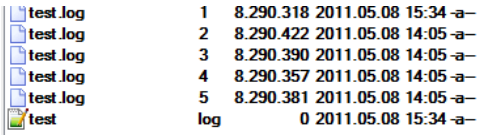
Galime patikrinti sukurtą administratorių ir jo kursą atsijungiant nuo super administratoriaus ir prisijungiant su admin_test1 prisijungimo duomenimis. 85 pav. matome testinio kliento langą prisijungus naujai sukurto administratoriaus vardu.



85 pav. Testinio kliento kursų valdymo kortelė

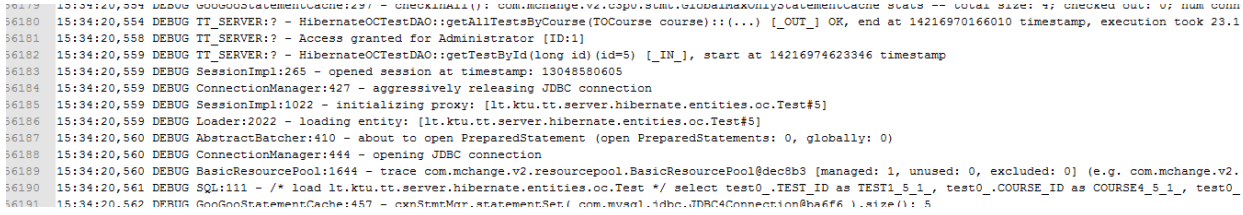
Kaip ir parodyta aukščiau 85 pav. administratorius mato ir kitus administratorius kurie priskirti jo kursui. Jis gali atjungti juos nuo kurso, bet negali naikinti ar keisti kitų administratorių duomenų, beje šis administratorius mato tik jam priskirtus kursus, bet ne kitus kursus, kuriems priskirti yra kiti administratoriai, jei jam šie kursai nėra priskirti.

Dabar atsižvelgiant į šio testavimo planą pereiname prie operacijų posistemės serverio dalies, pažiūrime log aplankalą, paveikslas *86* rodo susikaupusią stebėjimo informaciją



86 pav. Operacijų centro darbo stebėjimo ataskaitų failai

Pasirenkame 1 test log failą ir pažiūrime jame esančią informaciją. 87 pav. parodyta ištrauka iš šio failo.



87 pav. Ištrauka iš operacijų centro darbo stebėjimo ataskaitos

Šiuo metu serverio stebėjimas nustatytas į DEBUG detalumo lygį, matome kad serveris sėkmingai stebi savo darbą ir rašo vykdomus veiksmus į failą. Patikrinus Resursų saugyklos elgsenos stebėjimo rezultatus, neatitikimų taip pat nebuvo rasta.

7 PRIEDAS. APKLAUSŲ ANKETOS

Projekte naudojamos technologijos.

67 lentelė. Projekte naudojamų technologijų vertinimas

	Sunku pasakyti/ nežinau	Manau priešingai	Iš dalies sutinku	Iš dalies nesutinku	Visiškai pritariu
Realizacija leidžia naudoti skirtingas duomenų bazes.					X
Duomenų apdorojimo operacijos vyksta pakankamai greitai.					X
Realizacija leidžia programuotojams kurti ir lengvai diegti serveryje savo vertinimo arba paruošimo scenarijus.				X	
Posistemių veikla yra fiksuojama log failuose					X

Dokumentacijos vertinimas.

68 lentelė. Dokumentacijos vertinimas

	Sunku pasakyti/ nežinau	Manau priešingai	Iš dalies sutinku	Iš dalies nesutinku	Visiškai pritariu
Projekto dokumentacijoje aprašytos kuriamos posistemės.					X
Dokumentacijoje aprašomos naudojamas technologijos.					X
Dokumentacija pateikia posistemių panaudojimo atvejus, aktorius.					X
Dokumentacijoje pateikiama licencija ir/arba naudojimo sąlygos.					X
Dokumentacija nustato funkcinius ir nefunkcinius reikalavimus sistemai.					X

Dokumentacijoje apžvelgtos rizikos.					X
Naudotojo dokumentacija padeda sėkmingai paleisti projekto programinę įrangą.					X
Dokumentacijoje aprašyta kuriamų sistemų sudėtis ir paaiškinta kuriamų posistemių rolė visos TestTool sistemos atžvilgiu.					X
Naudotojo dokumentacija aprašo dažniausiai kylančias problemas ir jų sprendimo būdus.					X
Naudotojo dokumentacija aprašo realizacijos katalogų ir nustatymų failų paskirtį.					X

Realizacijos vertinimas.

69 lentelė. Realizacijos vertinimas

	Sunku pasakyti/ nežinau	Manau priešingai	Iš dalies sutinku	Iš dalies nesutinku	Visiškai pritariu
Suprogramuotos funkcijos veikia korektiškai.					X
Yra realizuota galimybės sėkmingam pirmam paleidimui ir pradiniam sistemos nustatymui.					X
Yra realizuoti posistemių konfiguraciniai failai.					X
Yra realizuota studento posistemės programinė sąsaja operacijų centre.					X
Yra realizuota operacijų centre administratoriaus programinė sąsaja.					X
Realizacija veikia stabiliai					X
Vartotojų slaptažodžiai nelaikomi					X

atviru tekstu.					