KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Lina Jankauskaitė, Tadas Karlonas

Grafinės vartotojo sąsajos generavimas naudojant UML diagramas

Magistro darbas

Darbo vadovė
doc. dr. Rita Butkienė

KAUNAS, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Lina Jankauskaitė, Tadas Karlonas

Grafinės vartotojo sąsajos generavimas naudojant UML diagramas

Magistro darbas

Recenzentas
doc. dr. J. Čeponis

2010-05-31

Darbo vadovas
doc. dr. Rita Butkienė

2010-05-31

Atliko
IFM-4/4 gr. stud.
Lina Jankauskaitė
Tadas Karlonas
2010-05-26

KAUNAS, 2010

TURINYS

SANTRAUKA

Grafinės vartotojo sąsajos generavimas naudojant UML diagramas

Šio darbo tikslas - pagreitinti ir palengvinti informacijos sistemų kūrimo procesą, sukuriant grafinės vartotojo sąsajos prototipo generavimo iš UML diagramų algoritmą. Darbe išanalizuota informacinių sistemų kūrimo aplinka, jos vartotojai, grafinės vartotojo sąsajos sudarymo metodai bei grafiniai elementai, taip pat – UML diagramos bei jų sąryšis su grafine vartotojo sąsaja.

Pagal atliktą analizę bei išsikeltus kokybės kriterijus, sukurtas algoritmas, generuojantis grafinę vartotojo sąsają iš panaudojimo atvejų, klasių ir veiklos diagramų. Grafinės vartotojo sąsajos algoritmas realizuotas *Java* programavimo kalba kaip atskira programa, kuri naudoja informaciją, gautą iš *MagicDraw UML* įrankiu sukurto XML projekto failo. Grafinė vartotojo sąsaja gali būti sugeneruojama dviem tipais – aplikacijai ir internetinei sąsajai. Pažymėtina, kad naudojantis sukurtu įrankiu, grafinė sąsaja gali būti ne tik sudaroma, bet ir visapusiškai redaguojama. Taip pat sugeneruojamas aplikacijos (*Java* programavimo kalba) bei internetinės sąsajos (HTML) programinis kodas.

Sukurtas grafinės vartotojo sąsajos generavimo įrankis išbandytas kuriant įvairaus sudėtingumo informacijos sistemų grafines sąsajas. Nustatyta, kad grafinės vartotojo sąsajos įrankį verta naudoti neatsižvelgiant į sistemų sudėtingumą.

Šio darbo tema buvo parašytas straipsnis, kuris pristatytas 15-oje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos (IVUS 2010)“

SUMMARY

Graphical User Interface Generation from UML Diagrams

The aim of this work is to accelerate and facilitate the process of information system development by creating the algorithm for GUI (graphical user interface) prototype generation from UML diagrams. The work analyses the environment of information system development, its users, creation methods and graphical elements of graphical user interface, as well as UML diagrams and their links to graphical user interface.

The algorithm that generates graphical user interface from the use case, class and activity diagrams was created in accordance to the performed analysis and determined quality criteria. The algorithm of graphical user interface was realised with the *Java* programming language as a separate program that uses information received from XML project file created by the *MagicDraw UML* tool. There are two types of graphical user interface generation: for application and for internet interface. It has to be noted that the created tool can be used not only for creation but also for thorough editing of the interface. Besides, a program code for application (*Java* programming language) and internet interface (HTML) is generated.

The created tool for graphical user interface generation is tested by creating graphical interfaces for information systems of various complexity. It has been found that the graphical user interface tool is worth being used in all cases.

An article on the theme of this work has been written and presented in the 15th inter-university conference for Master and PhD students “Information Society and University Studies (IVUS 2010)”.

1. ĮVADAS

Pastaraisiais metais sukuriama itin daug įvairaus pobūdžio informacinių sistemų (IS), kompiuterizuojančių svarbius verslo procesus. Informacinių sistemų naudojimas versle palengvina darbuotojams užduočių atlikimą bei pagreitina jų darbą, todėl mažėja įmonės sąnaudos, o veiklos efektyvumas didėja.

Informacinių sistemų kūrimo procesai padeda sistemų inžinieriams identifikuoti kompiuterizuojamus panaudojimo atvejus, išanalizuoti verslo problemas ir jas spręsti sukuriant informacinę sistemą. Kuriant tokią sistemą svarbiausia yra surinkti, specifikuoti ir išanalizuoti užsakovo pateikiamus reikalavimus.

Šiame darbe išanalizuota informacinių sistemų kūrimo aplinka, jos vartotojai, grafinės vartotojo sąsajos sudarymo metodai bei grafiniai elementai, taip pat – UML diagramos bei jų sąryšis su grafine vartotojo sąsaja.

Išanalizavus IS kūrimo aplinką, nustatyta, kad kūrime dalyvauja reikalavimų inžinieriai, projektuotojai, programuotojai, testuotojai ir kt. Kuriant informacijos sistemas labai svarų indėlį į darbo procesą įdeda ir užsakovas, kuris vertina, peržiūri ir pateikia komentarus apie kuriamą sistemą iš vartotojo pusės. Svarbiausia projektuotojų užduotis yra sumodeliuoti būsimą sistemą, tačiau jie nesukuria jokio tos sistemos prototipo, kurį būtų galima dar projektavimo proceso metu pademonstruoti vartotojams bei užsakovams. Sistemų kūrėjai susiduria su prototipo sudarymo problema. Specialistai siekia kuo anksčiau užsakovui pademonstruoti būsimos sistemos vartotojo sąsajos prototipą, o jam sukurti reikalingos papildomos priemonės ir papildomas sistemos projektuotojų darbas. Turint omenyje, kad sistemos projektuotojai tuo pačiu turi kurti ir sistemos reikalavimų specifikaciją bei projekto modelį, o dažniausiai projekto sąmatoje papildomi ištekliai (finansiniai bei laiko) prototipo kūrimui yra nenumatomi, šį procesą palengvintų algoritmas, automatiškai generuojantis grafinės vartotojo sąsajos prototipą. Toks algoritmas leistų anksčiau identifikuoti sistemos trūkumus bei pranašumus. Automatinis tokio prototipo kūrimas naudingas tiek užsakovui, tiek jo kūrėjams.

Atlikus lyginamąją egzistuojančių grafinę sąsają generuojančių įrankių ir metodų analizę nuspręsta kurti įrankį, leidžiantį automatiškai generuoti grafinę vartotojo sąsają iš UML diagramų neturint programavimo įgūdžių bei nurodant kuo mažiau papildomų nustatymų. Pabrėžtina, kad visi išanalizuoti įrankiai remiasi ne projektavimo etape projektuotojų bet kuriuo atveju sudaromomis UML diagramomis, o siekiama sukurti tokį įrankį, kuris grafinei vartotojo sąsajai sugeneruoti naudotų tik diagramų informaciją bei nustatymus, kuriuos galėtų atlikti bet kuris prie sistemos kūrimo dirbantis specialistas.

Pagal atliktą analizę, išsikeltus kokybės kriterijus bei reikalavimus sąsajos generavimo įrankiui, sukurtas algoritmas, generuojantis grafinę vartotojo sąsają iš panaudojimo atvejų, klasių ir veiklos diagramų.

Vartotojo sąsajos generavimo algoritmas realizuotas *Java* programavimo kalba kaip atskira programa, kuri naudoja informaciją, gautą iš *MagicDraw* įrankiu sukurto XML projekto failo. Panaudojimo atvejų, klasių bei veiklos diagramos, nubraižytos *MagicDraw UML* įrankiu įkeliamos į sukurtą įrankį XML formatu. Apdorojus XML kalba aprašytą diagramų informaciją, ji transformuojama į langų metaklases, šios metaklasės transformuojamos į grafinius elementus, o pastarieji atvaizduojami grafinėje sąsajoje. Transformacijoms atlikti remtasi moksliniuose straipsniuose pateiktais UML ir grafinės vartotojo sąsajos metamodeliais. Diagramų elementams, kurie aktualūs grafinei sąsajai generuoti, išskyrimui naudojamas specialiai grafinės sąsajos generavimo sistemai sukurtas *MagicDraw UML* įrankio *GUI* profilis, kuriame aprašomi specifiniai stereotipai, pagal kuriuos programa nusprendžia, kokius grafinės sąsajos langus kurti, kokius elementus tuose languose atvaizduoti bei kaip sukurti navigaciją tarp pačių sąsajos langų. Grafinė vartotojo sąsaja gali būti sugeneruojama dviem tipais – aplikacijai ir internetinei sąsajai.

Pažymėtina, kad naudojantis grafinės vartotojo sąsajos įrankiu sąsaja generuojama tik vieną kartą ir toliau gali būti visapusiškai redaguojama naudojantis langams bei elementams įkelti, redaguoti bei šalinti skirtomis funkcijomis. Pagal projekto sąsajos nustatymus taip pat sugeneruojamas programinis kodas tiek aplikacijos, tiek internetinei grafinei sąsajai.

Sukurtą grafinės vartotojo sąsajos įrankį nesudėtinga įdiegti bet kuriame kompiuteryje. Korektiškam sistemos paleidimui tereikia, kad kompiuteryje būtų įdiegta ne mažesnė nei *Java 1.6* versija. Sistemos programinius failus išsaugojus pasirinktoje kategorijoje, galima naudoti įrankį ir įkėlus duomenis generuoti, redaguoti bei peržiūrėti rezultatus.

Sukurtas grafinės vartotojo sąsajos generavimo įrankis išbandytas kuriant įvairaus sudėtingumo informacijos sistemų grafines sąsajas. Eksperimente inicijuotas penkių sistemų, kurių specifikacija aprašyta įvairiu panaudojimo atvejų, klasių bei veiklos diagramų elementų kiekiu, grafinių sąsajų kūrimas. Nustatyta, kad grafinės vartotojo sąsajos įrankį verta naudoti įvairaus sudėtingumo sistemų grafinėms sąsajoms generuoti. Šis įrankis naudingas tuo atveju, kai sistemos projektavimo etape sudaromos UML diagramos. Bandymuose atsižvelgta ne tik į sistemų sudėtingumą ir apimtį, bet ir į grafinės sąsajos sukūrimo tikslumą bei laiką, kurį užtruktų sąsają informacinių sistemų specialistas sudarydamas ranka ir naudojantis grafinės sąsajos generavimo įrankiu. Eksperimento rezultatai pateikti lentelėse bei juos vaizduojančiose diagramose.

Darbą sudaro 9 skyriai ir 5 priedai. Pirmame skyriuje analizuojami sąsajos generavimo įrankiai ir metodai, aplinka, vartotojai, UML diagramos, vartotojo sąsajos grafiniai elementai ir pačios sąsajos prototipo sudarymas. Antrame skyriuje pateikiama reikalavimų specifikacija, esybių ir kt. modeliai. Algoritmo aprašymą sudaro UML diagramų ir jų sąsajos su grafine vartotojo sąsaja aprašai, naudojamų UML diagramų bei grafinės vartotojo sąsajos metamodeliai, transformacijų tarp diagramų metaklasių ir langų metaklasių aprašai ir kt. Penktame skyriuje pateikiamas grafinės vartotojo sąsajos generavimo metodo pagrindimas bei projekto modeliai ir operacijų aprašai. Realizacijos skyriuje aprašomas diagramų paruošimas įrankiui, pačio įrankio veikimo principai ir funkcijos, pateikiamas testavimo modelis. Eksperimente aprašyti sistemų grafinių vartotojo sąsajų sudarymo rezultatai. Darbo pabaigoje pateiktos išvados, naudota literatūra, terminai bei sutrumpinimai. Grafinės vartotojo sąsajos generavimo įrankio sugeneruoti grafinių sąsajų langai (aplikacijos ir internetinės sąsajos) ir eksperimento rezultatai pateikti prieduose nr. 1 ir nr. 2, sukurtos grafinės vartotojo sąsajos generavimo įrankio vadovas – priede nr. 3, sugeneruoto programinio kodo pavyzdžiai – priede nr. 4, darbų pasidalinimo lentelė - priede nr. 5. Straipsnis, pristatytas 15-oje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos (IVUS 2010)“ pateiktas priede nr. 6.

2. ANALIZĖS DALIS

2.1 Analizės tikslas

Analizės tikslas – pasirinkti patikimus ir informatyvius analizės metodus, išsiaiškinti informacijos sistemų kūrime dalyvaujančius vartotojus, jų tipus bei savybes, identifikuoti jų tikslus ir sprendžiamas problemas. Analizės dalyje taip pat bus išanalizuoti esami grafinės vartotojo sąsajos automatinio generavimo sprendimai. Surinkti trūkumai bei privalumai padės išspręsti esamą problemą sukuriant technologijomis pagrįstą sprendimą informacijos sistemų projektavimo proceso automatizavimo problemai spręsti.

Analizės dalyje bus atlikta tyrimo objekto analizė, aplinkos ir vartotojų analizė, esamų sprendimų analizė, vartotojo sąsajos elementų analizė, aptarti UML (angl. *Unified Modeling Language*) diagramų tipai ir jų sąsaja su grafine vartotojo sąsaja. Taip pat išskiriami nefunkciniai reikalavimai bei kokybės kriterijai.

Tiriamas objektas - informacijos sistemų projektavimo procesas ir jo metu kuriami būsimos sistemos prototipai.

Tyrimo objekto ir esamų sprendimų analizėms naudojami šie metodai:

- Mokslinės literatūros analizės metodas;

- Objektinis sistemų analizės metodas;
- Struktūrinis sistemų analizės metodas;
- Stebėjimas;
- Statistikos;
- Dokumentų turinio analizės metodas.

Esamiems sprendimams aprašyti taip pat bus naudojama lyginamoji analizė, kadangi struktūriškai aprašyti analizės rezultatai yra paprasčiau įvertinami ir suprantami.

Šie metodai pasirinkti dėl tinkamumo objektui analizuoti.

2.2 Tyrimo sritis, objektas ir problema

2.2.1 Tyrimo sritis

Informacijos sistemų projektavimo proceso automatizavimo galimybės.

2.2.2 Tyrimo objektas

Tyrimo objektas – informacijos sistemų projektavimo procesas ir jo metu kuriami būsimos sistemos prototipai.

2.2.3 Tyrimo problema ir numatomas sprendimas

Kuriant informacines sistemas ar paslaugų sistemas, dažnai siekiama kuo anksčiau pademonstruoti vartotojui būsimos sistemos prototipą. Toks prototipas kūrimo proceso pradžioje dažniausiai būna tik grafinė vartotojo sąsaja, tačiau jai sukurti reikalingos papildomos priemonės ir papildomas sistemos projektuotojų darbas. Sistemos projektuotojai tuo pačiu turi kurti ir sistemos reikalavimų specifikaciją bei projekto modelį, o jiems aprašyti dažnai naudojama UML kalba. UML pagalba sumodeliuojami pagrindiniai sistemos procesai, funkcionalumas, numatoma struktūra, klasės, jų atributai, operacijos ir ryšiai. Daugeliu atvejų, iš UML kalba aprašytos sistemos specifikacijos galima būtų automatiškai ar automatizuotai generuoti grafinę vartotojo sąsają – pradinį būsimos sistemos prototipą.

Egzistuojantys vartotojo sąsajos generavimo metodai dažniausiai:

- remiasi ne UML diagramomis, o jau su realizacija susietais būsimos sistemos aprašais (pvz. WSDL (angl. *Web Services Description Language*) schemomis [1]);
- naudoja UML diagramas, bet reikalauja daug papildomos informacijos (pvz. OCL (angl. *Object Constraint Language*) aprašų [2]), kurią kartais sudėtingiau aprašyti, negu sukurti pačią grafinę sąsają.

Šiame darbe numatomas sprendimas – vartotojo sąsajos generavimas naudojant informaciją, aprašytą UML diagramose.

2.2.4 Tyrimo tikslas ir uždaviniai

Tyrimo tikslas – pagreitinti ir palengvinti informacijos sistemų (IS) kūrimo procesą, sukuriant grafinės vartotojo sąsajos prototipo generavimo iš UML diagramų algoritmą.

Uždaviniai:

1. Ištirti egzistuojančius grafinės vartotojo sąsajos generavimo algoritmus;
2. Išanalizuoti UML kalbą;
3. Išanalizuoti grafinės vartotojo sąsajos sudarymo principus;
4. Ištirti UML modeliuose pateikiamos informacijos ryšį su grafine vartotojo sąsaja;
5. Pasirinkti UML CASE (angl. *Computer-Aided Software Engineering*) įrankį bei kitas siūlymo realizavimui tinkamas priemones;
6. Aprašyti UML diagramų, reikalingų grafinio sąsajos generavimui, kūrimo metodiką;
7. Sudaryti grafinės vartotojo sąsajos generavimo algoritmą;
8. Realizuoti sudarytą algoritmą pasirinktomis kūrimo priemonėmis;
9. Išbandyti algoritmo taikymą pasirinktai informacijos sistemai.

2.2.5 Tyrimo objekto analizė

Pagrindiniai programinės įrangos kūrimo etapai yra šie:

- tyrimo ir analizės etapas;
- uždavinio specifikavimas;
- projektavimas;
- kodavimas;
- testavimas;
- priežiūra;
- dokumentavimas.

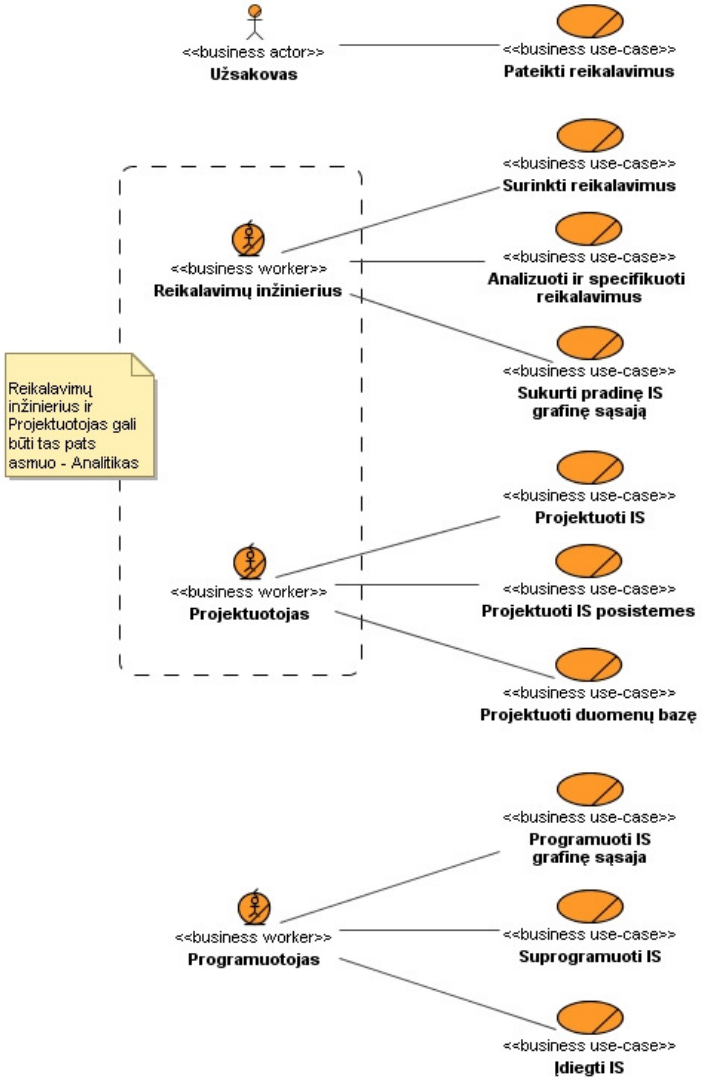
Šio darbo tyrimo objektas – informacijos sistemų projektavimo procesas ir jo metu kuriami būsimos sistemos prototipai. Projektavimo procesas pradedamas po užsakovo reikalavimų išsiaiškinimo, reikalavimų specifikacijos ir analizės.

Projektavimo proceso metu projektuotojai UML pagalba sudaro modelius, vaizduojančius sistemos procesus, struktūrą, klases, funkcionalumus.

2.3 Aplinkos analizė

Informacijos sistemų kūrime dalyvauja reikalavimų inžinierius, projektuotojai, programuotojai ir užsakovas.

Reikalavimų inžinierius yra atsakingas už reikalavimų surinkimą, specifikavimą ir analizę. Tai žmogus, kuris bendrauja su užsakovu ir reikalavimų sistemai surinkimo etape identifikuoja užsakovo poreikius. Reikalavimų inžinierius nustato pradinius panaudojimo atvejus bei sistemos vartotojus – aktorius, ir sudaro pagrindinius sistemos modelius. Projektuotojas pagal užsakovo pateiktus reikalavimus suprojektuoja informacinę sistemą, jos posistemes ir duomenų bazes nubraižydamas modelius, aprašančius kuriamą sistemą. Vartotojo sąsajos prototipo sudarymas apima du žingsnius: loginės ir fizinės sąsajos sukūrimą. Programuotojai kuria programinį kodą, todėl turėdami ne tik pateiktus reikalavimus sistemai, bet ir automatiškai sugeneruotą grafinės sąsajos prototipą galėtų geriau įvykdyti savo užduotis.



1 pav. Vartotojai, jų tipai ir veiklos sferos kuriant informacinę sistemą

Grafinės vartotojo sąsajos prototipo generavimas iš UML diagramų gali būti naudojamas šiuose etapuose:

- reikalavimų surinkimo;
- projektavimo;
- realizavimo.

2.4 Vartotojų analizė

2.4.1 Vartotojų aibė, tipai ir savybės

Identifikuoti pagrindiniai vartotojų tipai – reikalavimų inžinierius, projektuotojas bei programuotojas. Kuriant informacijos sistemas labai svarų indėlį į darbo procesą įdeda ir užsakovas, kuris vertina, peržiūri ir pateikia komentarus apie kuriamą sistemą iš vartotojo pusės. Vartotojų tipai bei jų savybes pateiktos 1 lentelėje.

1 lentelė. Vartotojų tipai ir savybės

Vartotojo tipas	**Savybės**
Reikalavimų inžinierius	Reikalavimų inžinierius yra atsakingas už reikalavimų iš užsakovo surinkimą, specifikavimą ir analizę. Tai aukštos kvalifikacijos specialistas, išmanantis reikalavimų inžineriją, reikalavimų surinkimo ir specifikavimo metodus. Norint išvengti subjektyvios reikalavimų interpretacijos, prie vieno projekto dažniausiai dirba vienas reikalavimų inžinierius.
Projektuotojas	Projektuotojo darbas – suprojektuoti informacinę sistemą pagal reikalavimų inžinieriaus pateiktus specifikuotus reikalavimus. Nubraižomi veiklos sąveikų, panaudojimo atvejų, tikslų, klasių, esybių ir kiti modeliai. Jie reikalingi programuotojams, realizuosiantiems informacinę sistemą. Projektuotojai – aukštos kvalifikacijos specialistai.
Programuotojas	Programuotojai atsakingi už korektiškai suprogramuotą informacinę sistemą. Tai aukštos kvalifikacijos specialistai.
Užsakovas	Užsakovu gali būti tiek fizinis, tiek juridinis asmuo, kurio veikloje reikalinga informacinė sistema, kompiuterizuojanti jo veiklą. Kadangi užsakovas retai išmano informacinių technologijas, sistemos projektavimo metu siekiama kuo anksčiau jam pademonstruoti būsimos sistemos prototipą.

2.4.2 Vartotojų tikslai ir problemos

Kiekvienas iš vartotojų tipų turi tikslus ir problemas, su kuriomis susiduria informacijos sistemų kūrimo procese. Sistemų analitikams bei programuotojams aktualu kuo greičiau bei kokybiškiau sukurti sistemą užsakovui, tuo tarpu užsakovas suinteresuotas efektyviu sistemos kūrėjų darbu bei rezultatu, padėsiančiu jam efektyviau vykdyti verslo procesus. Vartotojų tikslai ir problemos pateiktos 2 lentelėje.

2 lentelė. Vartotojų tipai, tikslai ir problemos

Vartotojo tipas	**Tikslai**	**Problema**
Reikalavimų inžinierius	Identifikuoti užsakovo pateiktus reikalavimus, juos specifikuoti ir išanalizuoti.	Sistemos projektavimo metu siekiama kuo anksčiau užsakovui pademonstruoti būsimos sistemos prototipą, tačiau jam sukurti reikalingos papildomos priemonės ir papildomas sistemos projektuotojų darbas. Sistemos projektuotojai tuo pačiu turi kurti ir sistemos reikalavimų specifikaciją bei projekto modelį. Dažniausiai projekto sąmatoje papildomi ištekliai (finansiniai bei laiko) prototipo kūrimui yra nenumatomi, todėl šį procesą palengvintų algoritmas, automatiškai generuojantis grafinę vartotojo sąsają.
Projektuotojai	Suprojektuoti informacinę sistemą pagal reikalavimų inžinieriaus pateiktą reikalavimų specifikaciją.	
Programuotojai	Suprogramuoti informacinę sistemą pagal užsakovo reikalavimus.	
Užsakovas	Naudotis kokybiškai sukurta informacine sistema ir efektyviai ją panaudoti verslo procesuose.	

2.5 Vartotojo sąsaja ir jos struktūra

Vartotojo sąsaja yra svarbi programinės įrangos dalis, kuri reikalinga norint perduoti informaciją iš vartotojo sistemai bei vykdyti jos funkcijas. Tai vienintelė sistemos dalis, kurią mato vartotojas ir kurios pagalba jis valdo sistemą. Dažnai grafinės vartotojo sąsajos programinis kodas sudaro apie 50% visos sistemos kodo [3].

Vartotojo sąsaja gali būti išskirta į keturias dalis [3]. Konceptualiame dizaine išskiriami aplikacijų konceptai objektų lygmenyje, taip pat nurodomi jų ryšiai. Funkciniame ir semantiniame sistemos dizaine nurodomos visos objektų operacijos su jų įvedimų ir išvedimų

informacija. Trečiojoje dalyje, sintaksiniame dizaine (angl. *syntactic*), apibrėžiamas įvedimų ir išvedimų eiliškumas. Techninės įrangos susiejimo dizainas nustato, kaip sintaksiniai elementai realizuojami techninėje dalyje.

Labai svarbu, kad programinės įrangos dizainas būtų suprantamas ir aiškus vartotojui. Kuriant dizainą, turi būti įvertinami sistemos naudojimo kriterijai, pavyzdžiui, kiek laiko vartotojas užtrunka išsiaiškindamas sistemos veikimą, kiek laiko vartotojas užtrunka įvykdydamas vieną ar kitą veiksmą sistemoje, kiek kartų jis suklysta ir t.t. Norėdami, kad sistema būtų patogu naudotis įvairiems vartotojams, programinės įrangos kūrėjai stengiasi išlaikyti tam tikrus bendrus grafinės vartotojo sąsajos kūrimo principus. Pavyzdžiui, programinėje įrangoje meniu dažniausiai kuriamas ne vertikalus, o horizontalus ir jo vieta – viršutinėje sistemos lango dalyje, o internetiniuose projektuose meniu dažniausiai kuriamas kairėje puslapio dalyje ir vertikalaus tipo. Tai leidžia labiau patyrusiam vartotojui intuityviai naudotis sistema ir palengvina jos funkcionalumo įsisavinimą bei taupo laiką.

Grafinę vartotojo sąsają sudaro langas ir lango elementai, kurie atvaizduojami tame lange. Vartotojo sąsajos elementai aprašyti 2.5.1 - 2.5.2 skiltyse.

Vartotojo sąsaja gali būti dviejų tipų – aplikacinė ir internetinė. Aplikacijos grafinė sąsaja kuriama programinei įrangai, o internetinė – sistemoms, kurios pasiekiamos per naršyklę, internetinėms svetainėms ir kt.

2.5.1 Aplikacijos grafinės sąsajos elementai

3 lentelėje pateiktas *Java* programavimo kalbos aplikacijos grafinių sąsajų elementų sąrašas [3].

3 lentelė. Aplikacijos grafinių sąsajų elementų sąrašas.

Tipas	**Elementai**
Meta klasės	*JApplet, JComponent, JDesktopPane, JLayeredPane, JRootPane, JViewport*
Formos	*JCheckBox, JCheckBoxMenuItem, JComboBox, JFormattedTextField, JFormattedTextField.AbstractFormatter, JFormattedTextField.AbstractFormatterFactory, JPasswordField, JRadioButton, JRadioButtonMenuItem, JSlider, JSpinner, JSpinner.DateEditor, JSpinner.DefaultEditor, JSpinner.ListEditor, JSpinner.NumberEditor, JTextArea, JTextField, JToggleButton, JToggleButton.ToggleButtonModel*
Redagavimas	*JEditorPane, JTextPane*

Mygtukai	*JButton*
Grupavimas ir maketavimas	*JPanel, JSeparator, JSplitPane JTabbedPane*
Langas ir dialogai	*JFrame, JInternalFrame, JInternalFrame.JDesktopIcon, Jwindow, JDialog, JFileChooser, JOptionPane*
Tekstai ir paveiksliukai	*JLabel*
Lentelė	*JList, JTable*
Meniu	*JMenu, JMenuBar, JMenuItem, JPopupMenu, JPopupMenu.Separator, JToolBar, JToolBar.Separator*
Medis	*JTree, JTree.DynamicUtilTreeNode, JTree.EmptySelectionModel*
Slinkties juosta	*JScrollBar, JScrollPane*
Progresinė juosta	*JProgressBar*
Patarimas	*JToolTip*

Dažniausiai aplikacijos grafinės vartotojo sąsajos sudarymui naudojami šie elementai – *JCheckBox, JComboBox, JRadioButton, JTextArea, JTextField, Jbutton, JPanel, Jseparator, JFrame, JWindow, JDialog, Jlabel, JList, Jtable, JMenu, JMenuBar, JmenuItem, JscrollBar.*

2.5.2 Internetinės grafinės sąsajos elementai

4 lentelėje pateiktas HTML grafinių vartotojo sąsajų elementų sąrašas [3].

4 lentelė. HTML grafinių sąsajų elementų sąrašas.

Tipas	**Elementai**
Meta informacija	*html, head, title, base, meta, link, noframes, style, script, noscript, isindex*
Šrifto formatavimas	*center, br, em, strong, q, sub, sup, tt, i, b, big, small, u, s, strike, basefont, font, pre, blockquote, span, address, ins, del, bdo, dfn, code, samp, kbd, var, cite, abbr, acronym*
Pagrindinė dalis	*body*
Pavadinimai	*h1, h2, h3, h4, h5, h6*
Sąrašai	*ul, ol, li, dl, dt, dd, dir, menu*
Teksto struktūra	*p, div*
Lentelė	*table, caption, thead, tfoot, tbody, colgroup, col, tr, th, td*
Nuorodos	*a, map, area*

Formos	*form, label, input, select, optgroup, option, textarea, button*
Mygtukai	*button, input type="button"*
Paveiksliukai	*img*
Objektai	*object, param, applet*
Karkasas	*frame, iframe*

Dažniausiai internetinės grafinės vartotojo sąsajos sudarymui naudojami šie elementai - *html, head, title, meta, style, body, h1, h2, ul, l, i, p, div, table, thead, tbody, tr, th, td, a, form, label, input, select, textarea, button, button, input type="button".*

2.5.3 Vartotojo sąsajos prototipo sudarymas

Vartotojo sąsajos kūrimas remiasi reikalavimų surinkimo metu identifikuotais panaudojimo atvejais ir prototipinių langų kūrimu kiekvienam panaudojimo atvejui. Pirmiausiai vartotojo sąsajos dizaineris išsiaiškina užsakovo poreikius bei sistemos vartotojo viziją. Yra du pagrindiniai būdai kurti sistemos vartotojo sąsajos prototipą [3]. Vienas iš jų, paprastesnis, dar kitaip vadinamas „mažo tikslumo“ (angl. *low-fidelity*), yra sudaryti prototipinius sistemos langus ant popieriaus arba kompiuterio ekrane. Sudaromas ne toks tikslus prototipas, tačiau vartotojas gali suprasti sistemos veikimo principą ir pabandyti įsivaizduoti, kaip veiks galutinė sistema. Ypač tai aktualu didesnėms sistemoms, kurių grafinę sąsają įsivaizduoti užsakovui, neturinčiam informacinių technologijų žinių, sunku. Sudėtingesnis ir daugiau pastangų reikalaujantis variantas – sudaryti veikiantį sistemos prototipą ir leisti vartotojui jį išbandyti. Tiek pirmuoju, tiek antruoju atveju sudaryto prototipo panaudojimą stebi kūrėjai, identifikuoja trūkumus bei juos šalina. Pagrindiniai prototipų sudarymo būdų privalumai ir trūkumai pateikti 5 lentelėje.

5 lentelė. Prototipų sudarymo būdų lyginamoji analizė

	Mažo tikslumo prototipo sudarymas	**Tikslaus sistemos prototipo sudarymas**
Sudarymo kaštai	Nedideli kaštai.	Brangus realizavimas.
Laiko kaštai	Greitai sudaromas ir keičiamas.	Prototipo sudarymas atima nemažai laiko ir pastangų.
Testavimo galimybė	Nėra.	Nesudėtingai testuojama. Neefektyvus reikalavimų surinkimui.
Tobulinimo galimybė	Greitas ir patogus	Sudėtingesnis tobulinimas radus

	tobulinimas radus klaidas.	klaidas.
Sistemos koncepcijos demonstravimas	Nesudėtinga pademonstruoti vartotojo sąsajos koncepciją.	Neefektyvus norint pademonstruoti tik koncepcinę sąsają.
Funkcionalumo realizavimas	Funkcionalumas realizuojamas tik koncepciniame lygmenyje.	Pilnas sistemos funkcionalumas.
Interaktyvumas	Nėra.	Interaktyvus prototipo variantas.
Navigacijos aiškumo lygmuo	Navigacija nėra aiški.	Realus navigavimas tarp sistemos langų.
Prototipų variantų skaičius	Galimybė nesunkiai sudaryti keletą sąsajos prototipo variantų.	Sudaromas vienas variantas.
Panašumas į galutinį sistemos variantą	Galutinis sistemos variantas dažnai skiriasi nuo koncepcinio.	Galimybė nesunkiai įsivaizduoti galutinį produktą.
Specialistų indėlis	Gali sudaryti bet kuris komandos narys.	Darbą gali atlikti tik projektuotojai ir programuotojai.

2.6 UML modeliavimo kalba

UML – vieninga modeliavimo kalba – modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektinių programų dokumentus [4]. Ji aprūpina sintaksiniais ženklais, kurie apibūdina visus pagrindinius sistemos vaizdus, naudodama įvairias diagramų rūšis [4]. Tai standartinė modeliavimo kalba, skirta verslo, informacinių ir programų sistemų vaizdavimui, specifikavimui, projektavimui, dokumentavimui.

UML buvo sukurta 1997 m. J.Rumbaugh, I.Jacobson ir G.Booch objektinio projektavimo metodų pagrindu (*Rational Software* korporacijoje). Šiuo metu ją prižiūri OMG (*Object Management Group*) – Objektinių standartų organizacija.

Galima sudaryti keliolika diagramų. UML 2.0 apibrėžiama trylika diagramų, kurios drauge sudaro bendrą sistemos vaizdą su reikalingais aspektais. Diagramose sistemą galima projektuoti skirtingais lygmenimis – nuo to, kaip vartotojas suvokia kuriamą sistemą iki to, kaip pati sistema bus kuriama, detalizuojant jos kritinius procesus, struktūrą ir t.t. [5]

UML diagramas galima suskaidyti į dvi didesnes šakas: vaizduojančias sistemos elgseną ir struktūrą. Sistemos elgsena vaizduojama panaudojimo atvejų, veiklos,

bendradarbiavimo, sekos, būsenų, laiko diagramomis. Struktūrą galima pavaizduoti klasių, komponentų, įdiegimo, objektų diagramomis.

2.6.1 UML diagramų tipai

Panaudojimo atvejų diagrama (angl. *use case diagram*) – UML diagrama, aprašanti, ką projektuojama sistema gali atlikti, kartu įtraukiant ir išorinius sistemos veikėjus (aktorius). Didelėms sistemoms ši diagrama skirstoma į posistemes. Pagrindinis šios diagramos elementas – panaudojimo atvejis, aprašantis aibę panašių sąveikos scenarijų. Kiekvienas panaudojimo atvejis paprastai turi vieną pagrindinį ir keletą šalutinių scenarijų, kurie aprašomi dinaminėmis UML diagramomis (sekų, bendradarbiavimo, veiklos) [4].

Panaudojimo atvejų diagramos yra susijusios su sąveika tarp sistemos ir aktorių (objektų, kurie tiesiogiai sąveikauja su sistema). Šios diagramos yra naudingos įsivaizduoti bendrą sistemos kontekstą ir sistemos ribas. Panaudojimo atvejų diagrama sudaroma visoms projektuojamoms sistemoms, nes yra lengviausiai suprantama net vartotojui, neturinčiam specialių projektavimo ar programavimo žinių.

Pagrindiniai diagramos elementai – panaudojimo atvejis ir aktorius. Kiekvienas aktorius gali būti susijęs su keletu panaudojimo atvejų, kurie nurodo, ką susijęs vartotojas gali atlikti su sistema.

Generuojant vartotojo sąsajos prototipą panaudojimo atvejų diagrama būtų naudinga apibrėžiant sistemos langų skaičių. Pavyzdžiui, kiekvienam panaudojimo atvejui gali būti sukurtas atskiras sąsajos langas.

Kiekviename lange turi būti atitinkami grafiniai elementai informacijos įvedimui, sąrašo atvaizdavimui ir t.t. Šiuos laukus galima aprašyti klasių diagrama.

Klasių diagrama (angl. *class diagram*) – tai statinės struktūros diagrama, aprašanti sistemos klases bei ryšius tarp jų. Ši diagrama identifikuoja visas sistemos klases ir apibrėžia kiekvienai klasei jos atributus, operacijas ir ryšius į kitas klases. Ryšiai apima paveldėjimą, asociaciją ir agregavimą. Klasių diagrama yra centrinė UML modelio diagrama.

Veiklos diagrama (angl. *activity diagram*) – modeliuoja dinaminę sistemos elgseną (vaizduojami veiksmai) [4]. Veiklos šioje diagramoje nurodo, kas atsitinka įvykdžius tam tikrą operaciją, aprašytą klasėse klasių diagramoje.

Veiklos diagrama, generuojant vartotojo sąsajos prototipą, galėtų nustatyti tvarką, pagal kurią būtų iškviečiami sugeneruoti langai. Kiekviena veiklos diagramos veikla atitiktų panaudojimo atvejį.

Bendradarbiavimo diagrama (angl. *communication diagram*) – iliustruoja ryšius ir sąveikas tarp programinės įrangos objektų.

Sekų diagrama (angl. *sequency diagram*) – apibūdina dinaminę veikėjų (aktorių), sistemos objektų ir sistemos sąveiką.

Būsenų diagrama (angl. *state machine diagrama*) – apibūdina vieno sistemos objekto dinaminį elgesį kaip būsenų kaitą. Vaizduojamos svarbiausios verslo ar veiklos sistemos būsenos bei tų būsenos kitimas toje pačioje sistemoje. Diagrama nusako objektų būsenas ir jų pasikeitimus laike. Būsenos diagrama žingsnis po žingsnio atvaizduoja verslo ir operatyvinius sistemos komponentų srautus [4].

Komponentų diagrama (angl. *component structure diagram*) – aprašomi sistemoje naudojami komponentai ir jų bendradarbiavimas.

Veiklos procesų diagrama (angl. *bussiness process diagram*) – atvaizduojami analizuojamos veiklos procesai.

Kiti UML diagramų tipai - realizavimo (angl. *implementation*), bendrinė DDL (angl. *Data Definition Language*), tinklinio sujungimo diagrama (angl. *networking*), Oracle DDL, WSDL diagrama, XML (angl. *Extensible Markup Language*) schemos diagramos.

Generuojant vaizdinį sistemos prototipą gali būti naudojamos klasių, būsenų, panaudojimo atvejų, sekų diagramos.

2.6.2 UML diagramų ryšys su grafine vartotojo sąsaja

Sudarant UML diagramas taikomi tam tikri apribojimai.

Pirmiausia sudaroma Panaudojimo atvejų diagrama. Šios diagramos elementai – panaudojimo atvejai – atspindi grafinės sąsajos langus. Klasių diagrama suprantama kaip kuriamos sistemos grafinės sąsajos elementų visuma. Klasės turi turėti sąryšius su panaudojimo atvejais, kadangi sudarant langą pagal panaudojimo atvejį, reikia žinoti, kurie atributai kuriam langui turi būti priskiriami. Veiklos diagrama atspindi navigaciją tarp grafinės sąsajos langų. Šios veiklos languose galėtų būti sukurtos kaip mygtukai su veiklos pavadinimu.

2.7 Problemos sprendimo metodų literatūros šaltiniuose analizė

Šioje dalyje bus analizuojami siūlomi sprendimai analizuojamai problemai spręsti.

Autorių Rosado da Cruz A. M., Pascoal de Faria J. siūlomas sprendimas sistemos struktūrą aprašyti klasių diagramomis, o sistemos funkcionalumą – panaudojimo atvejų diagrama. Formalia OCL kalba aprašomos sistemos būsenos ir elgsena. Kiekvienas panaudojimo atvejis yra susiejamas su tam tikra sistemos būsena. Šio metodo trūkumas – aprašomas sistemos funkcionalumas, tačiau neaprašoma, kaip tai sistema turi atlikti ar įvykdyti [2].

Internetinės paslaugos (angl. *web services*) gali būti iškviečiamos viena kitos arba vartotojo. Šių iškvietimų pobūdis yra skirtingas. Tam, kad vartotojas galėtų iškviesti internetines paslaugas, jis turi matyti grafinę vartotojo sąsają. Jai generuoti autoriai siūlo naudoti WSDL aprašymo kalbą kartu su papildomais grafinės sąsajos aprašymais [1].

Grafinę vartotojo sąsają generuoti galima ir naudojant XUL, XML ir Gecko biblioteką [6]. Sudarius XUL ir XML failus, jie yra nuskaitomi ir įrašomi į atmintį. Kiekvienam XUL elementui yra priskiriamas tam tikras dizaino elementas.

Vienas iš siūlomų sprendimų - grafinę vartotojo generuoti iš šių UML diagramų: panaudojimo atvejų (angl. use-case), klasių (angl. *class*), bendradarbiavimo (angl. *collaboration*) ir būsenų (angl. *state machine*). Papildomi apribojimai kuriamai sistemai aprašomi deklaratyvia apribojimų kalba (OCL) kalba [7]. Ši kalba skirta aprašyti taisykles, taikomas UML modeliams.

Klasių diagramą autorius siūlo naudoti sistemos struktūros aprašymui ir ryšių tarp sistemos klasių identifikavimui. Panaudojimo atvejų diagrama yra susijusi su sistemos vartotojų ir sistemos sąveika, todėl šia diagrama aprašomas sistemos kontekstas, funkcionalumas ir elgsena. Bendradarbiavimo diagrama naudojama identifikuoti kiekvieno panaudojimo atvejo vykdymo scenarijui. Šią funkciją taip pat gali atlikti ir sekų diagrama. Būsenų diagrama parodo objektų būsenų seką įvairiuose vykdymo etapuose. Ši diagrama gali būti sujungta su objektų klasėmis, kurių būsenos kinta dinamiškai.

Polak G., Jarosz J. siūlomas metodas yra generuoti grafinę vartotojo sąsają naudojant aprašytus šablonus. Išsirinkus atitinkamus parametrus yra sugeneruojama standartinio stiliaus grafinė sąsaja [8].

6 lentelė. Siūlomi sprendimai problemai spręsti

Siūlomo sprendimo šaltinis	**Sprendimas**	**Trūkumai**
A. Miguel Rosado da Cruz, J. Pascoal de Faria [2]	Panaudojimo atvejų ir klasių diagramos. Papildomi apribojimai – OCL kalba.	Aprašomas sistemos funkcionalumas, tačiau neaprašoma, kaip tai sistema turi atlikti ar įvykdyti.
Jiang He, I-Ling Yen [1]	Vartotojo sąsajai generuoti autoriai siūlo naudoti WSDL aprašymo kalbą kartu su papildomais grafinės sąsajos aprašymais.	Papildomas grafinės sąsajos specifikavimas, kuris turi būti pritaikytas kiekvienam vartotojui.

N. Aloia, C. Concordia, M. T. Paratore [6]	XUL, XML ir Gecko biblioteka	Reikalingos papildomos XUL kalbos žinios.
M. Elkoutbi, I. Khriss ir R. Keller [7]	Panaudojimo atvejų, klasių, bendradarbiavimo ir būsenų diagramos. Papildomi apribojimai – OCL kalba.	Reikia papildomų OCL žinių.
Polak G., Jarosz J. [8]	Siūlomas metodas generuoti grafinę vartotojo sąsają naudojant aprašytus šablonus.	Formų generavimas neprisirišantis prie duomenų bazės.

2.8 Grafinę vartotojo sąsają generuojančių įrankių analizė

Šiuo metu yra sukurta keletas įrankių, kurie palengvina grafinės vartotojo sąsajos generavimą, todėl verta išanalizuoti jų privalumus, trūkumus bei veikimo principus.

Nuspręsta išanalizuoti *MagicDraw UML*, *Oracle Forms Builder* ir *ProVision Workbench* įrankius ir juos palyginti tarpusavyje.

2.8.1 MagicDraw UML

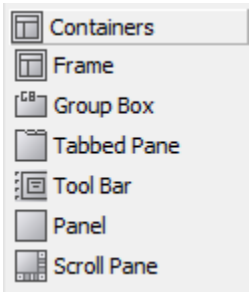
Keletas iš diagramų braižymo įrankių turi galimybę nubraižyti ar sugeneruoti vartotojo sąsajos prototipą. Vienas iš tokių įrankių – *MagicDraw UML. 16.6 MagicDraw UML* versijoje atsirado naujas diagramų tipas – Vartotojo sąsajos modeliavimo diagrama. Pagrindiniai šios diagramos sudarymo tikslai:

- palengvinti bei sutrumpinti darbą vartotojo sąsajos dizaineriams bei programuotojams;
- galimybė greičiau gauti užsakovų komentarus ir pastabas dėl grafinės sąsajos;
- galimybė projekto eigoje kuo anksčiau ištestuoti vartotojo sąsają bei jos panaudojamumo galimybes ir ją suderinus gauti kuo geresnį galutinį variantą;
- nesudėtingai išbandyti keletą vartotojo sąsajos variantų ir pasirinkti tinkamiausią, bei kiti.

Šios diagramos [9] pagalba galima sudaryti vartotojo sąsajos prototipą naudojantis pagrindiniais sąsajų elementais: mygtukais, meniu, sąrašais, tekstiniais laukais, lentelėmis, pasirinkimo elementais, nuorodų, slaptažodžių laukais ir t.t. bei susieti ją su atitinkamomis klasėmis tame pačiame projekte.

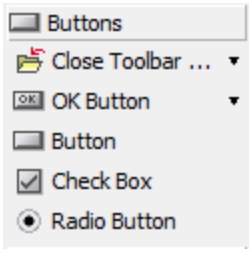
MagicDraw UML vartotojo sąsajos diagramos elementai yra suskirstyti keturias didesnes grupes: blokus, mygtukus, tekstinius laukus ir kitus. Blokai apima įvairius langų

elementus: pagrindinį langą su jo išjungimo, sumažinimo ir padidinimo mygtukais, kelių pakopų langų elementą, lango bloko su slankiosiomis juostomis elementą ir pan.

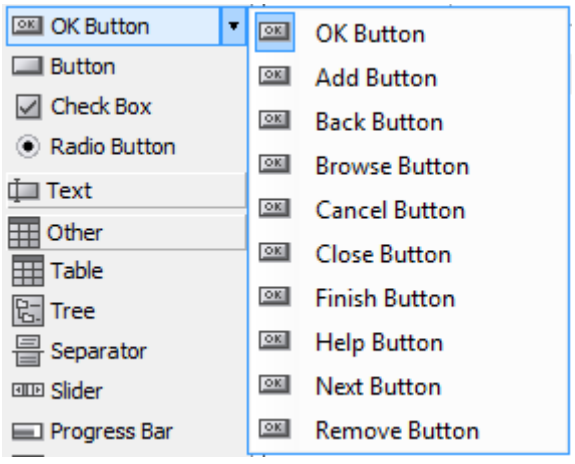


2 pav. Blokų elementai

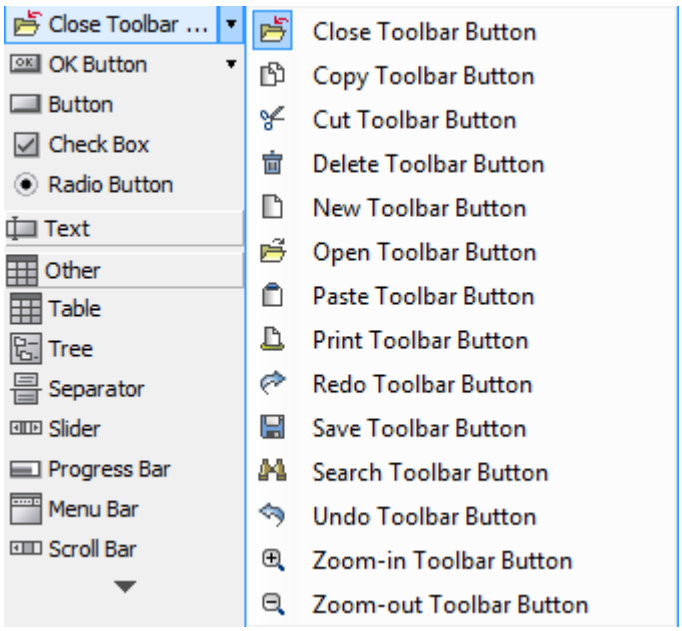
Siūlomi mygtukai (3 pav.): universalus mygtukas, specialieji mygtukai (4 pav.) – „OK“, „Atšaukti“, „Pridėti“, „Ieškoti“, „Uždaryti“, Pagalba“ ir kt., visi pagrindiniai mygtukai su įprastais *Windows* operacinės sistemos paveikslėliais (5 pav.), reiškiančiais išsaugojimo, atspausdinimo, langą uždarymo, lango atidarymo, naujo elemento kūrimo, sekančio ir praeito elemento pasirinkimo, šalinimo, patikrinimo ir kt. funkcijas, taip pat prie šios grupės priskiriamas patikrinimo elementas su varnele bei taškeliu.



3 pav. Siūlomi *MagicDraw UML* mygtukai

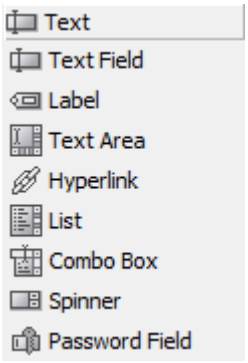


4 pav. Specialieji mygtukai



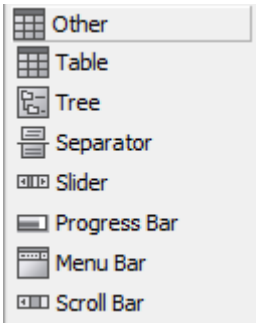
5 pav. Mygtukai su įprastais *Windows* operacinės sistemos paveikslėliais

Tekstinių elementų grupė apima elementarų vienos bei keleto eilučių tekstinį lauką, pavadinimą, iškrentantį sąrašą, tekstinį lauką su slankiojančia vertikaliąja juosta, slaptažodžio laukelį ir kt.



6 pav. Tekstinių laukų elementai

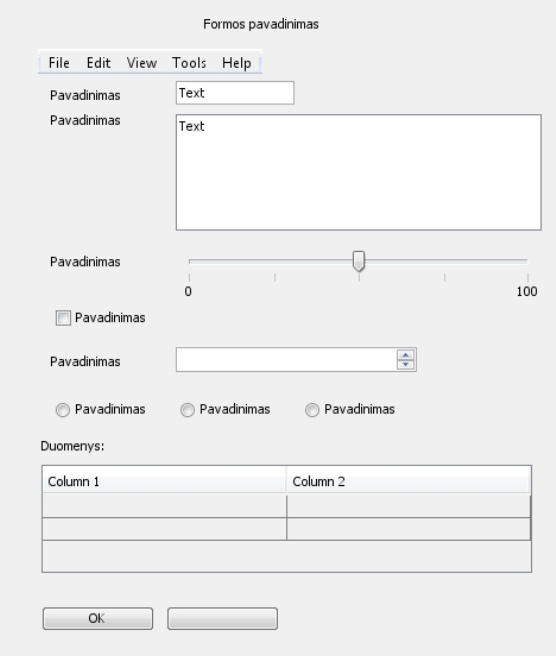
Prie likusiųjų kitų elementų grupės priskiriami rečiau vartotojo grafinės sąsajos kūrimui naudojami elementai: lentelės, slankiojanti juostelė, medis, skirtukas, meniu ir pan.



7 pav. Kiti grafiniai elementai, nepriskirti jokiai kitai grupei

Pagrindinis šios diagramos trūkumas – iš atskirų elementų suprojektuota vartotojo sąsajos diagrama gali būti išsaugoma ir peržiūrima tik kaip paveikslėlis arba interneto sąsajos

prezentacija. Be to, ji nėra generuojama iš kitų UML diagramų, o yra sudaroma kaip atskira diagrama.



8 pav. *MagicDraw* įrankiu sudaryta pavyzdinė grafinė vartotojo sąsaja

7 lentelė. MagicDraw įrankio privalumai ir trūkumai

Privalumai	**Trūkumai**
Palengvina darbą dizaineriams bei programuotojams, dirbantiems su vartotojo sąsajos projektavimu bei realizavimu.	Vartotojo sąsajos diagrama sudaroma atskirai, o ne generuojama iš jau nubraižytų UML diagramų.
Sudarant grafinę vartotojo sąsają galima naudotis eile sąsajos elementų, keisti jų spalvas, dydžius, šriftą.	Gali būti peržiūrima kaip paveikslėlis, o ne reali sugeneruota vartotojo sąsaja su aktyviais mygtukais, sąrašais bei kitais elementais.
Palaikoma *Windows XP*, *MAC*, etc.	
Galima išsaugoti keletu formatų: *JPG*, *PNG*, *TIFF*, *XML*.	

2.8.2 Oracle Forms Builder

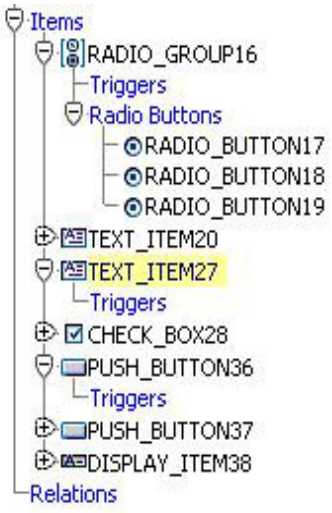
Oracle Forms Builder [10] yra įrankis, leidžiantis generuoti vartotojo sąsają. *Oracle Forms Builder* siejasi su sukurta *Oracle* (ar kita) duomenų baze ir leidžia sukurti formas, kurios parodo, įveda ar redaguoja duomenis toje duomenų bazėje. Formų sugeneravimui reikalingas ir *The Forms Runtime* įrankis. Kuriant vartotojo sąsają, gali būti naudojami įvairūs elementai: meniu, mygtukai, pavadinimai, iškrentantys sąrašai ir kt. (9pav.)

9 pav. Vartotojo sąsajai kurti naudojami elementai

Horizontaliame meniu (10 pav.) galima redaguoti teksto šriftą, dydį, lygiuoti pažymėtus elementus ir t.t.

10 pav. Horizontalus vartotojo sąsajos redagavimo meniu

Vartotojo sąsajos formos yra kuriamos trigerių pagrindu (11 pav.). Skirtingi trigeriai iškviečiami kiekviename kritiniame žingsnyje įvedant, redaguojant ar darant kitokius veiksmus formoje. Formų sudarymas remiasi šių trigerių modifikavimu. Tokiu būdu įmanoma sukurti visas reikalingas formas ar jų šablonus, reikalingus dirbant su numatyta duomenų baze.



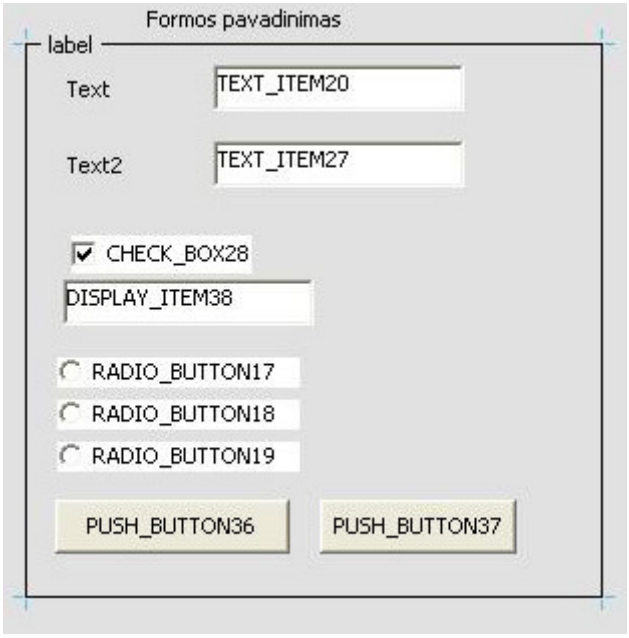
11 pav. Trigeriai

Norint sukurti formą naudojantis *Oracle Forms Builder* programa, pirmiausia reikia sukurti funkcijų diagramą ir duomenų bazę. *The Application Design Transformer* įrankis konvertuoja sukurtas funkcijas į modulius, kurie gali būti realizuojami kaip atskiros formos arba ataskaitos. *The Application Design Transformer* taip pat konvertuoja funkcijose naudojamus duomenis į duomenis, kurie bus naudojami moduliuose, bei apibrėžia, kaip duomenys bus naudojami aplikacijų generavimui.

Pirmuoju formų kūrimo etapu sudaromi automatizuotų funkcijų modulių apibrėžimai. Naudojantis *Function Hierarchy Diagrammer* sukuriama nauja diagrama, grindžiama AUTO 1 "*Handle automated functions*" funkcija. Atlikus reikalingus nustatymus, vykdomas generavimo procesas. Šio proceso rezultatas – modulio apibrėžimas. Jis ir bus naudojamas veikiančios formos generavimui. *Oracle Forms Builder* įrankyje yra nemažai nustatymų, kurie keičia formos elementų išdėstymą, vaizdinį sprendimą ir t.t.

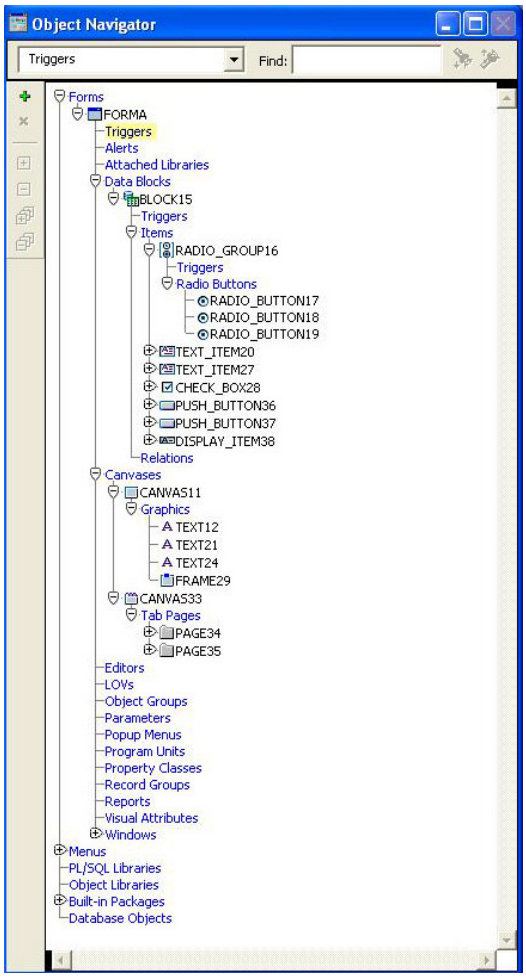
Naudojantis *Oracle Forms Builder* įrankiu taip pat galima kurti internetines aplikacijas, kurių pagalba galima vykdyti užklausas, duomenų bazės įrašų atnaujinimą, trynimą ir įvedimą. Kūrimo procesas panašus į jau aprašytą procesą. Internetinių aplikacijų peržiūra vykdoma naršyklės pagalba.

Sukurta nesudėtinga pavyzdinė forma pavaizduota 12 paveiksle.



12 pav. Pavyzdinė forma

Pavyzdinės formos objektų navigatorius programoje:



13 pav. Pavyzdinės formos objektų navigatorius programoje

8 lentelė. Oracle Forms Builder įrankio privalumai ir trūkumai

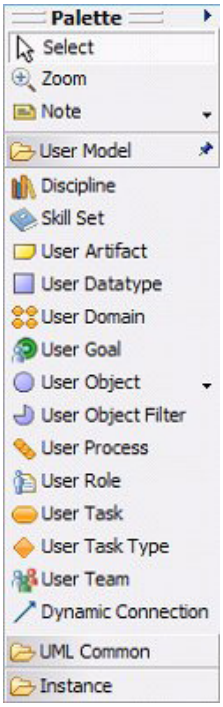
Privalumai	**Trūkumai**
Sugeneruojama forma su aktyviais mygtukais, kuri vykdo veiksmus su sukurta duomenų baze.	Darbas su *Oracle Forms Builder* reikalauja ne tik projektavimo, bet ir programavimo žinių.
Sudarant grafinę vartotojo sąsają galima naudotis įvairiais sąsajos elementais.	Kuriant formas, reikia ne tik sudaryti duomenų bazę, bet ir numatyti daug papildomų nustatymų.
Gali būti kuriamos internetinės aplikacijos.	

2.8.3 Provision Workbench

The User Interface Generator yra *IBM® InfoSphere Master Data Management (MDM) Workbench* komponentas [11], kurio dėka galima sudaryti vartotojo modelį (angl. *User model*). Vartotojo modelis, sukurtas minėtu įrankiu, yra reikalingas norint sugeneruoti internetinę aplikaciją. *The User Interface Generator* įrankiu galima naudotis ir turint minimalius UML modeliavimo pagrindus.

Pirmiausia reikia įdiegti *User Modeling Profile*, nes jame aprašyti stereotipai bus naudojami kuriant vartotojo modelį. Lengviausia vartotojo modelį kurti naudojantis šablonais, taigi vedlio pagalba pasirenkamas standartinis šablonas ir jį atitinkantis vartotojo modelis. Jį sukūrus, atsiranda *User Roles* paketas, kuris naudojamas stereotipams ir rolėms apibrėžti.

Internetinei sąsajai sugeneruoti taip pat reikalingi ir tikslų bei užduočių modeliai (klasių diagramos), parodantys, kaip rolės yra susiję tarpusavyje. Šis modelis kuriamas *User Roles* pakete. Paletėje galima išsirinkti nemažai reikalingų objektų. Jiems priskiriami pavadinimai, aprašymai, stereotipai ir kt. Objektai sujungiami standartiniais ryšiais.



14 pav. Objektų paletė

Žodyno pakete yra sukuriama objektų aibė, naudojama organizacijoje. Objektams naudojamos pagrindinės užduotys: sukurti, skaityti, atnaujinti ir trinti (angl. *CRUD – Create*, *Read*, *Update*, *Delete*). Sukūrus šiuos modelius, juos galima panaudoti internetinei sąsajai sukurti naudojantis *The User Interface Generator* paketu.

9 lentelė. Įrankių lyginamoji analizė

	MagicDraw	***Oracle Forms Builder***	***Provision WorkBench***
Ar vartotojo sąsaja generuojama iš diagramų informacijos?	+	+	+
Ar reikia nurodyti papildomų nustatymų?	+	+	+
Ar naudojamos UML diagramos?	+	-	-
Ar rezultatas yra veikianti vartotojo sąsaja?	-	+	-
Ar galima rinktis įprastus elementus iš sąrašo?	+	+	+
Ar galima sugeneruoti internetinę vartotojo sąsają?	-	+	+

Ar reikia programuoti?	-	+	+

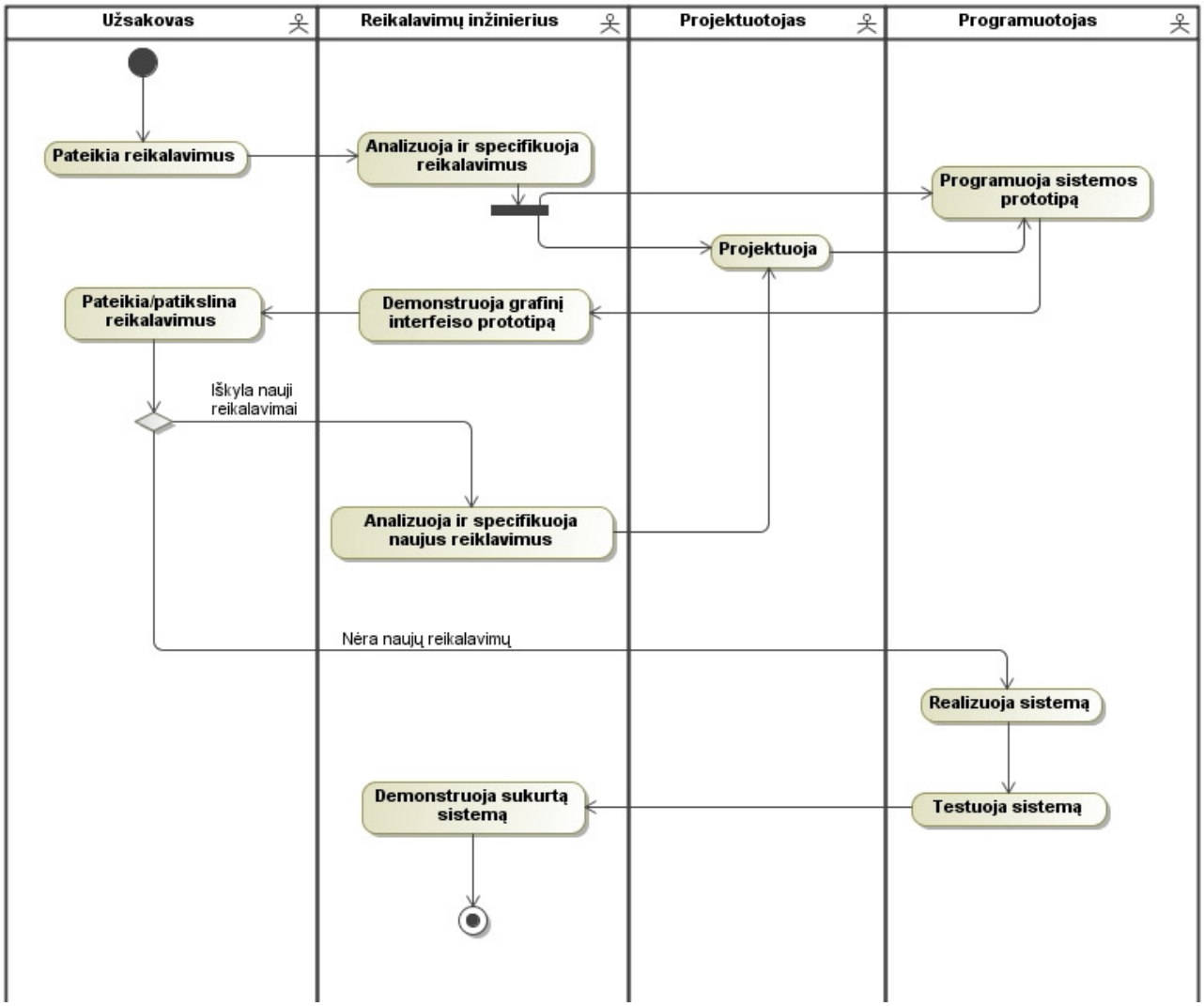
Atlikus lyginamąją *MagicDraw UML*, *Oracle Forms Builder* ir *ProVision Workbench* įrankių analizę nuspręsta kurti įrankį, leidžiantį automatiškai generuoti grafinę vartotojo sąsają iš UML diagramų neturint programavimo įgūdžių bei nurodant kuo mažiau papildomų nustatymų. Pažymėtina, kad visi išanalizuoti įrankiai remiasi ne projektavimo etape projektuotojų bet kuriuo atveju sudaromomis UML diagramomis, be to, turi kitų trūkumų. Pavyzdžiui, *MagicDraw UML* įrankio pagalba galima sudaryti tik paveikslėlį, kuris parodo, kaip galėtų atrodyti grafinė vartotojo sąsaja, be to, grafinė sąsaja nėra sugeneruojama automatiškai. Kiti įrankiai remiasi specialiai sudaromomis diagramomis vartotojo grafinei sąsajai generuoti pagal konkrečius reikalavimus, *Oracle Forms Builder* reikalauja ir papildomų programavimo žinių.

2.9 Siekiamos sistemos apibrėžimas

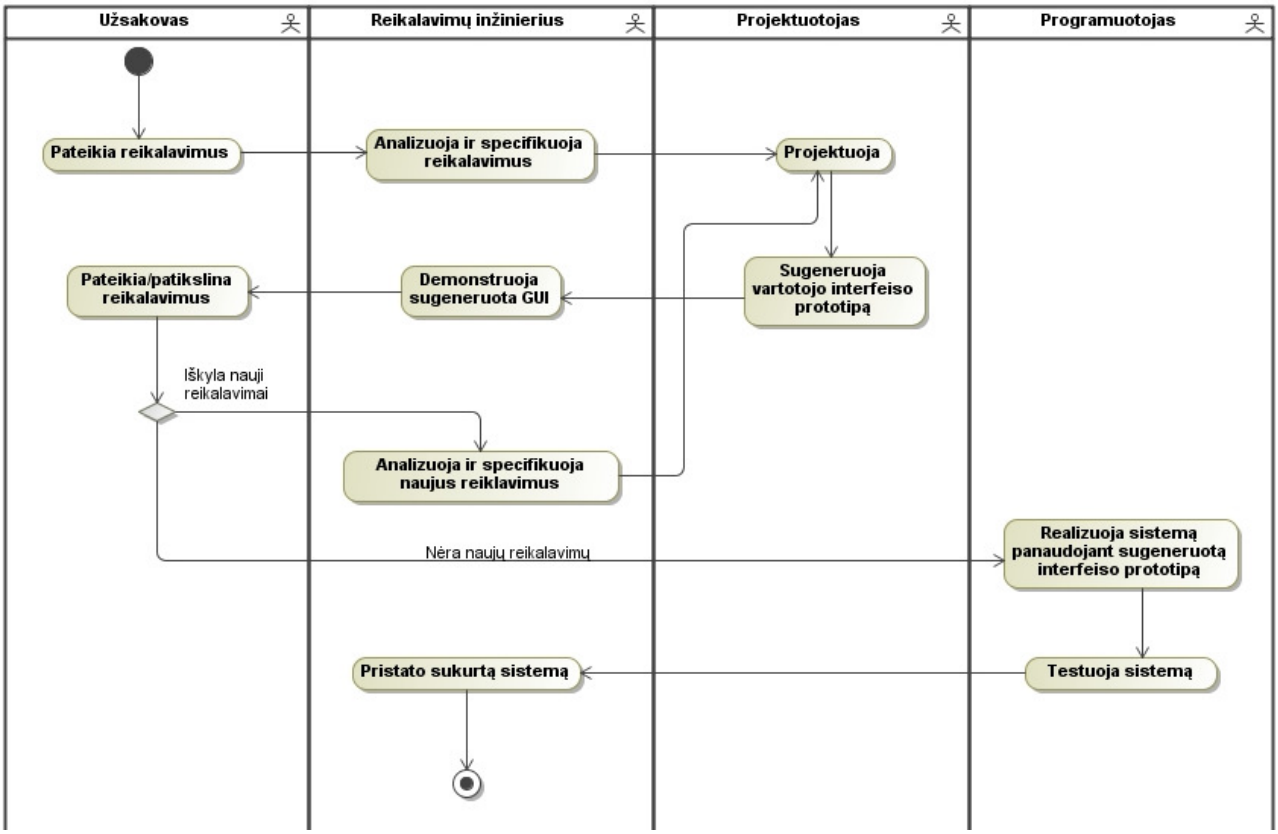
Siekiama kuo anksčiau pademonstruoti užsakovui būsimos sistemos prototipą kuo mažesnėmis darbo sąnaudomis. Automatiškai sugeneruota vartotojo grafinė sąsaja turi būti naudinga tiek reikalavimų inžinerijos procese, tiek kuriamos informacijos sistemos realizavime. Siekiama, kad automatiškai sugeneruotas informacijos sistemos prototipas būtų kuo panašesnis į galutinį rezultatą ir nereiktų iš naujo programuoti grafinės vartotojo sąsajos realizacijos etape.

Užsakovui pademonstravus pradinį sistemos prototipą gali anksčiau išryškėti sistemos trūkumai ir privalumai, į kuriuos laiku atsižvelgus būtų išvengta nereikalingų nuostolių.

Šiuo metu naudojamas kūrimo procesas pavaizduotas 15 paveiksle, o siekiamas – 16 paveiksle.



15 pav. Šiuo metu naudojamas kūrimo procesas



16 pav. Siekiamas naudoti kūrimo procesas

Išanalizavus siūlomus sprendimus vartotojo sąsajos generavimui, bus siekiama grafinę vartotojo sąsają generuoti iš panaudojimo atvejų, klasių ir veiklos diagramų.

Generuojant vartotojo sąsajos prototipą panaudojimo atvejų diagrama būtų naudinga apibrėžiant sistemos langų skaičių. Kiekvienam panaudojimo atvejui reikėtų nustatyti, ar bus reikalingas atskiras sąsajos langas.

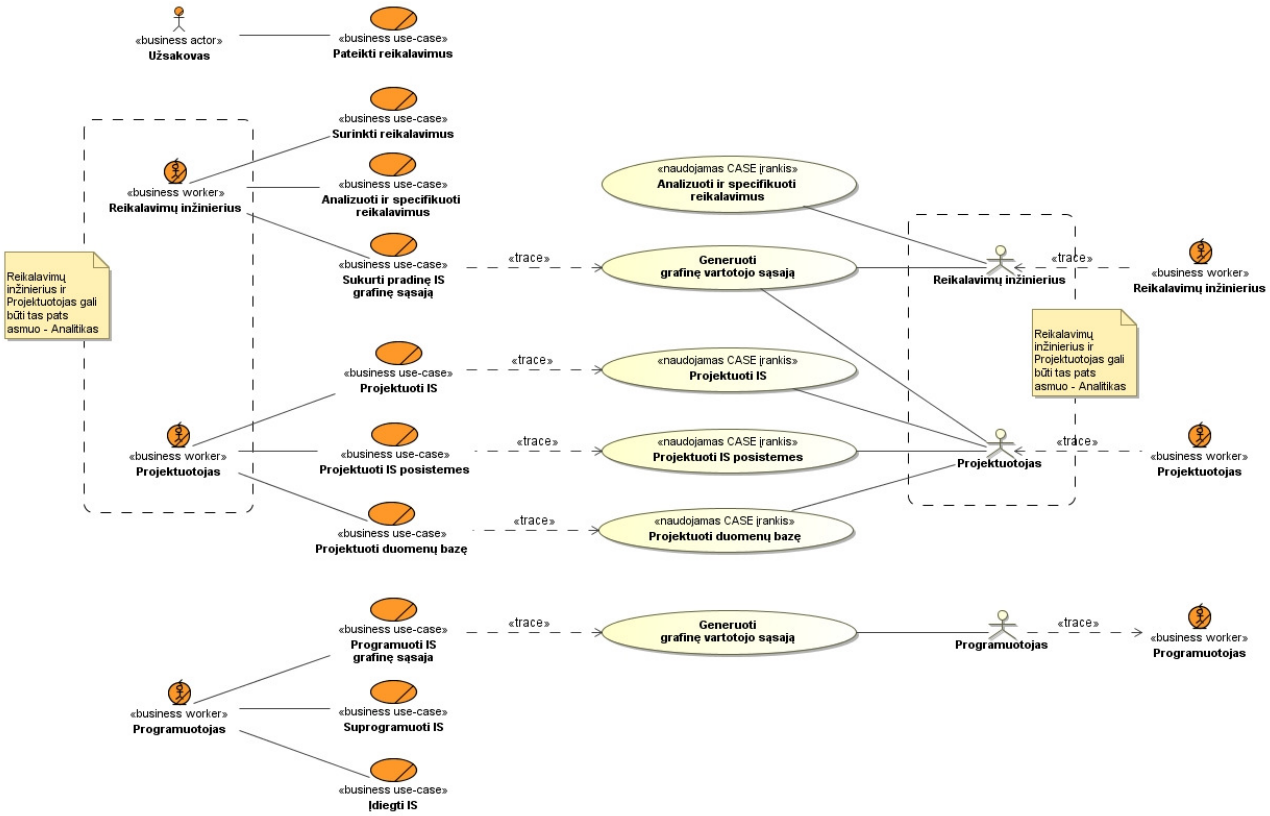
Klasių diagramoje nurodomos klasės, jų ryšiai ir atributai. Šie duomenys bus panaudoti grafiniams elementams sukurti pagal klasių atributų tipus.

Veiklos diagrama generuojant vartotojo sąsajos prototipą, nustatys tvarką, pagal kurią iškviečiami sugeneruoti langai. Kiekviena veiklos diagramos veikla atitiktų panaudojimo atvejį.

2.10 Darbo tikslas ir siekiami privalumai

Siūlomas sprendimas iš UML diagramų generuoti grafinę vartotojo sąsają būtų naudojamas informacinių sistemų kūrime. Programinę įrangą galėtų naudoti tiek mažos, tiek ir didelės įmonės, siūlančios informacijos sistemų kūrimo paslaugas. Šis sprendimas bus pritaikytas naudotis tiek projektuotojams, tiek programuotojams, į kurių užduotis įeina informacinės sistemos grafinės sąsajos ar jos prototipo kūrimas. 17 paveiksle pavaizduotoje diagramoje matoma, kad panaudojimo atvejai „Sukurti pradinę IS grafinę sąsają“ bei „Programuoti IS grafinę sąsają“ naudojant automatinį grafinės vartotojo sąsajos iš UML

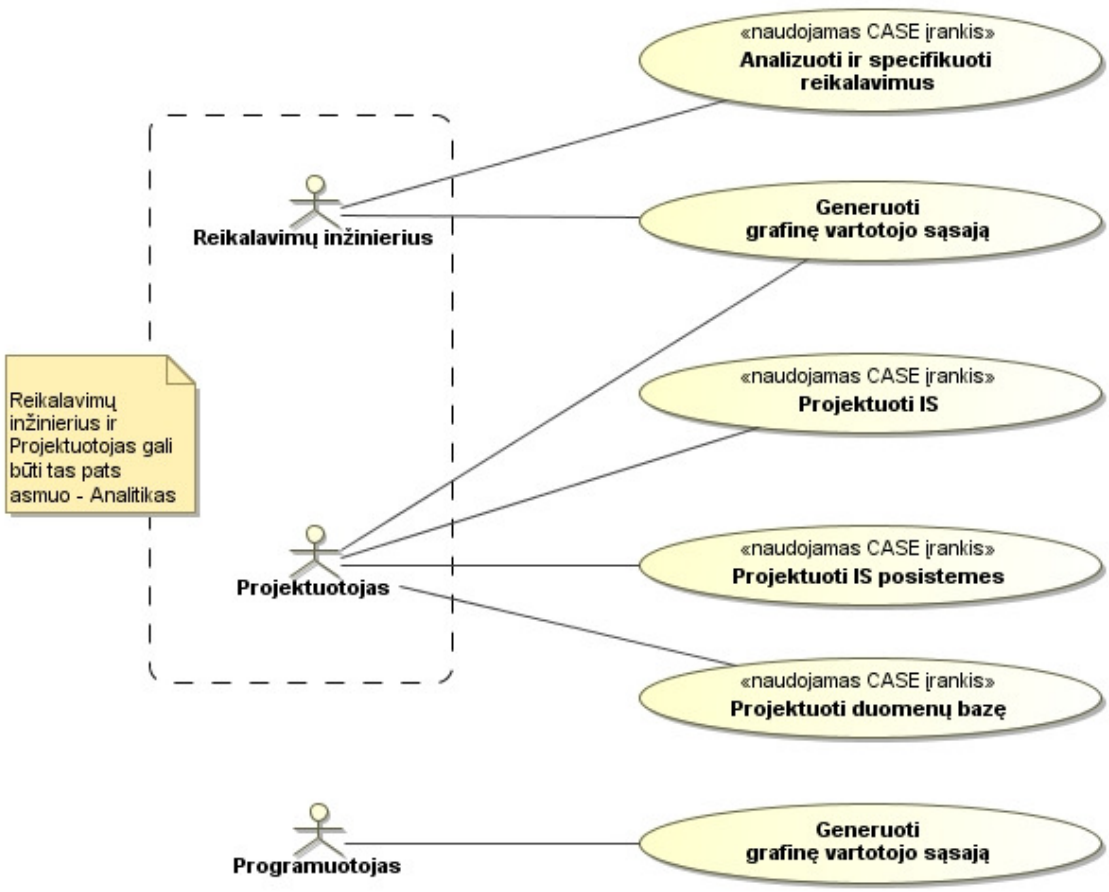
diagramų generavimą keičiami vieninteliu panaudojimo atveju „Generuoti grafinę vartotojo sąsają“.



17 pav. Veiklos ir kompiuterizuojamų panaudojimo atvejų priklausomybės

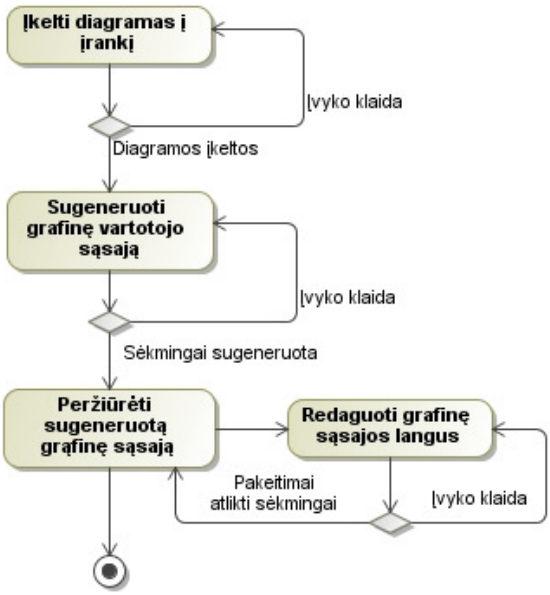
2.11 Kompiuterizuojamos funkcijos

Kompiuterizuojamų panaudojimo atvejų diagrama pavaizduota 18 paveiksle. Pagrindinis kompiuterizuojamas panaudojimo atvejis – „Generuoti grafinę vartotojo sąsają“.



18 pav. Kompiuterizuojami panaudojimo atvejai

Į šio kompiuterizuojamo panaudojimo atvejo veiklą (19 pav.) įeina šios pagrindinės veiklos: įkelti diagramas, atlikti generavimą ir peržiūrėti grafinę sąsają. Sugeneruotos grafinės sąsajos langus turėtų būti galima redaguoti, įtraukti naujus bei pašalinti nereikalingus. Diagramas siūloma braižyti su CASE įrankiu, pavyzdžiui, *MagicDraw UML.*



19 pav. Panaudojimo atvejo „Generuoti grafinę vartotojo sąsają“ veiklos diagrama

2.12 Reikalavimai duomenims

Numatoma, kad automatizuotam grafinės vartotojo sąsajos generavimui bus naudojami tokie duomenys:

- iš įkeltos į sistemą panaudojimo atvejų diagramos bus nuskaitomi panaudojimo atvejai;
- iš įkeltos į sistemą klasių diagramos bus nuskaitomos klasės ir jų atributai;
- iš veiklos diagramos bus nuskaitomos veiklos ir jų ryšiai.

Panaudojimo atvejų diagramoje būtinai turi būti:

- bent vienas panaudojimo atvejis.

Klasių diagramoje turi būti:

- bent viena klasė su bent vienu atributu.
- klasė, kuri bus naudojama grafiniams elementams sukurti, turi būti susieta su bent vienu panaudojimo atveju.

Veiklos diagramos apribojimai:

- kiekviena veikla, reiškianti navigaciją tarp langų, turi sietis su bent vienu panaudojimo atveju;
- veiklos diagrama turi būti sudaryta bent iš vienos veiklos.

2.13 Nefunkciniai reikalavimai ir apribojimai

2.13.1 Reikalavimai standartams

Diagramos, kurios naudojamos sąsajos generavimui turi būti sudaromos pagal UML notaciją.

2.13.2 Reikalavimai suderinamumui su kitomis sistemomis

UML diagramos, kurios įkeliamos į sistemą turi būti braižomos *MagicDraw UML* įrankiu. Į sistemą bus įkeliamas šio įrankio sukurtas XML failas.

2.13.3 Kiti reikalavimai

Pagrindiniai nefunkciniai sistemos reikalavimai:

1) Patogumui:

Sistema bus patogi naudoti tuomet, kai bus išpildyti žemiau nurodyti reikalavimai sistemos dizainui bei įvedamiems duomenims.

1.1) Reikalavimai sistemos dizainui:

- Sistemos funkcijos turi būti vykdomos meniu ir mygtukų pagalba. Turi būti realizuota lengvai suprantama navigacija.

1.2) Reikalavimai vartotojo duomenims:

- Į sistemą įkeliamas *MagicDraw XML* projekto byla.
- Tikrinamas įvestų duomenų korektiškumas.
- Visi įvedami duomenys saugomi sukurtoje *MySQL* duomenų bazėje.

2) Sistemos modeliavimui ir realizavimui:

3.1) Sistema turi būti suprojektuota *MagicDraw UML* įrankiu.

3.2) Duomenų saugojimui turi būti sukurta duomenų bazė.

2.14 Rezultato kokybės kriterijai

Grafinės vartotojo sąsajos prototipo sudarymui apibrėžiami tam tikri kokybės kriterijai. Visų pirma, grafinės sąsajos generavime turi būti naudojami pagrindiniai grafiniai elementai (sąrašai, tekstiniai laukai, pavadinimai, pasirinkimai ir t.t.), kurie turi būti parenkami automatiškai arba redaguojant sąsajos langus. Analizės metu nustatyti pagrindiniai grafiniai elementai pateikti 2.5.1 - 2.5.2 skyriuose.

Grafinei vartotojo sąsajai sugeneruoti naudojamos projektavimo etape bet kuriuo atveju sudaromos panaudojimo atvejų, klasių ir veiklos diagramos. Grafinę sąsają generuojantis įrankis naudoja būtent šių diagramų informaciją, taigi turi būti sutaupomas laikas, kuris skirtas sudaryti grafinės sąsajos prototipui. Dažniausiai toks prototipas sudaromas rankiniu būdu ir užima nemažai laiko, taigi įrankis turėtų efektyviai išnaudoti diagramose jau atvaizduotą informaciją.

Vartotojo sąsaja turi būti generuojama neatsižvelgiant į panaudojimo atvejų, klasių ar veiklų kiekį diagramose.

2.15 Analizės išvados

1. Vartotojo sąsajos prototipas yra naudingas:
   - reikalavimų analizės ir specifikavimo etape, nes pademonstruoja būsimos sistemos veikimo principus pagal vartotojo pateiktus reikalavimus;
   - realizuojant sistemą taip pat gali būti naudojamas jau sukurtas grafinės sąsajos prototipas.
2. Analizuotuose egzistuojančiuose sprendimuose vartotojo sąsajos generavimas yra sudėtingas ir užima daug laiko aprašant diagramas. Be to, metodai dažniausiai remiasi ne UML diagramomis, o jau su realizacija susietais būsimos sistemos aprašais, o tie kurie naudoja UML diagramas, reikalauja daug papildomos informacijos.

3. Pastebėta, kad reikėtų tokios generavimo priemonės, kuri padėtų sugeneruoti grafinės vartotojo sąsajos prototipą, naudingą įvairiuose kūrimo etapuose – reikalavimų analizės ir specifikavimo, projektavimo, realizavimo.
4. Būtų naudinga sukurti vartotojo sąsajos generavimo algoritmą, kuris naudotų minimalų reikalingų diagramų kiekį ir neperkrautų vartotojo reikalavimais įvesti daug papildomos informacijos. Taip pat svarbu, kad papildomos informacijos įvedimas nebūtų labai sudėtingas, priešingu atveju, vartotojui bus paprasčiau kurti vartotojo sąsają rankiniu būdu, negu naudoti pasiūlytą algoritmą.
5. Sistema, generuojanti grafinę vartotojo sąsają iš bet kuriuo atveju projektavimo metu sudaromų UML diagramų sutaupytų laiko, skirto sąsajos prototipo kūrimui, o esant net ir dideliems pakeitimams, sąsają tereikėtų tik redaguoti, o ne sudaryti iš naujo.
6. Siūlomam generavimo metodui būtų reikalinga panaudojimų atvejų diagrama, nurodanti sistemos langus, klasių diagrama ir veiklos diagrama.
7. Nustatyti pagrindiniai aplikacijos ir internetinės grafinės vartotojo sąsajos elementai bei pačio prototipo sudarymo būdai.

3. REIKALAVIMŲ ANALIZĖ IR SPECIFIKACIJA

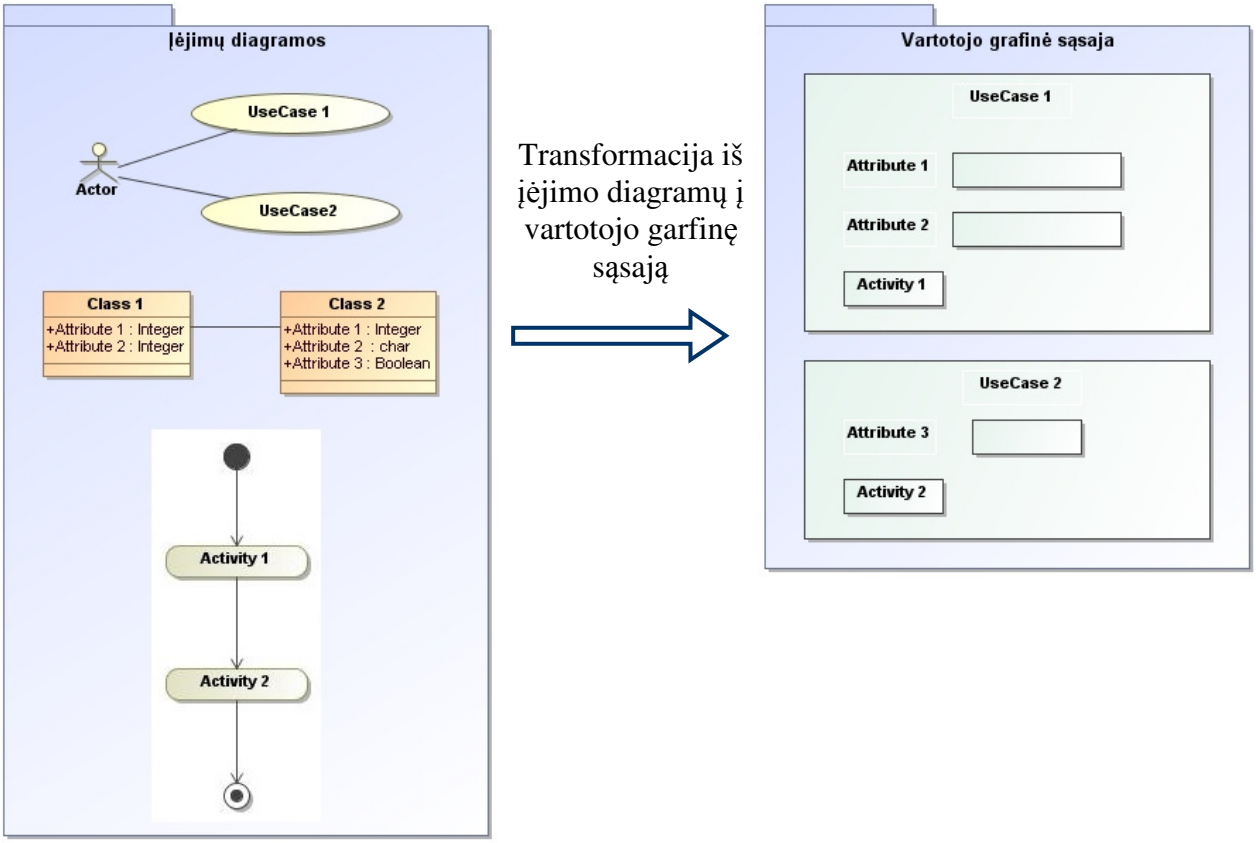
3.1 Reikalavimų specifikacija

Pagrindinio panaudojimo atvejo „Generuoti grafinę vartotojo sąsają“ veiklos diagrama pavaizduota 20 paveiksle.

20 pav. Vartotojo grafinės sąsajos generavimo veiklos etapų diagrama

3.1.1 Taikymo proceso schema ir taikymo situacijos

Kurdami sistemą su dviem panaudojimo atvejais, dviem klasėm ir dvejomis veiklomis gausime sistemos grafinę vartotojo sąsają, pavaizduotą 21 paveikslo dešinėje pusėje.

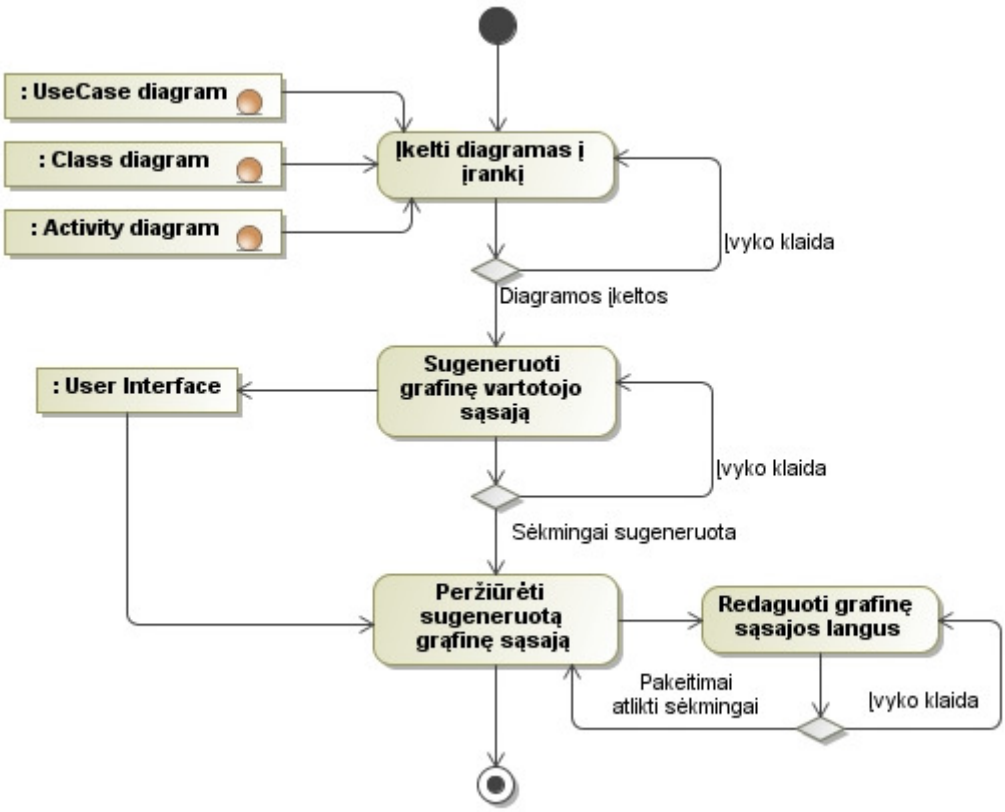


21 pav. Transformacija iš diagramų į programos langus

Kadangi veiklos diagrama nurodo grafinės vartotojo sąsajos langų navigaciją, o kai, pavyzdžiui, panaudojimo atvejų diagramoje yra tik vienas panaudojimo atvejis, ji nėra būtina. Visais kitais atvejais, kai yra keletas panaudojimo atvejų ir grafinėje vartotojo sąsajoje reikia sukurti keletą langų, veiklos diagramoje būtina atvaizduoti langų atidarymo eiliškumą.

Sistema pagal įvestas diagramas sugeneruos grafinę vartotojo sąsają. Ją galima redaguoti įkeliant naujus langus, elementus, juos redaguojant ar pašalinant. Sistema taip pat pateiks programinį grafinės sąsajos kodą.

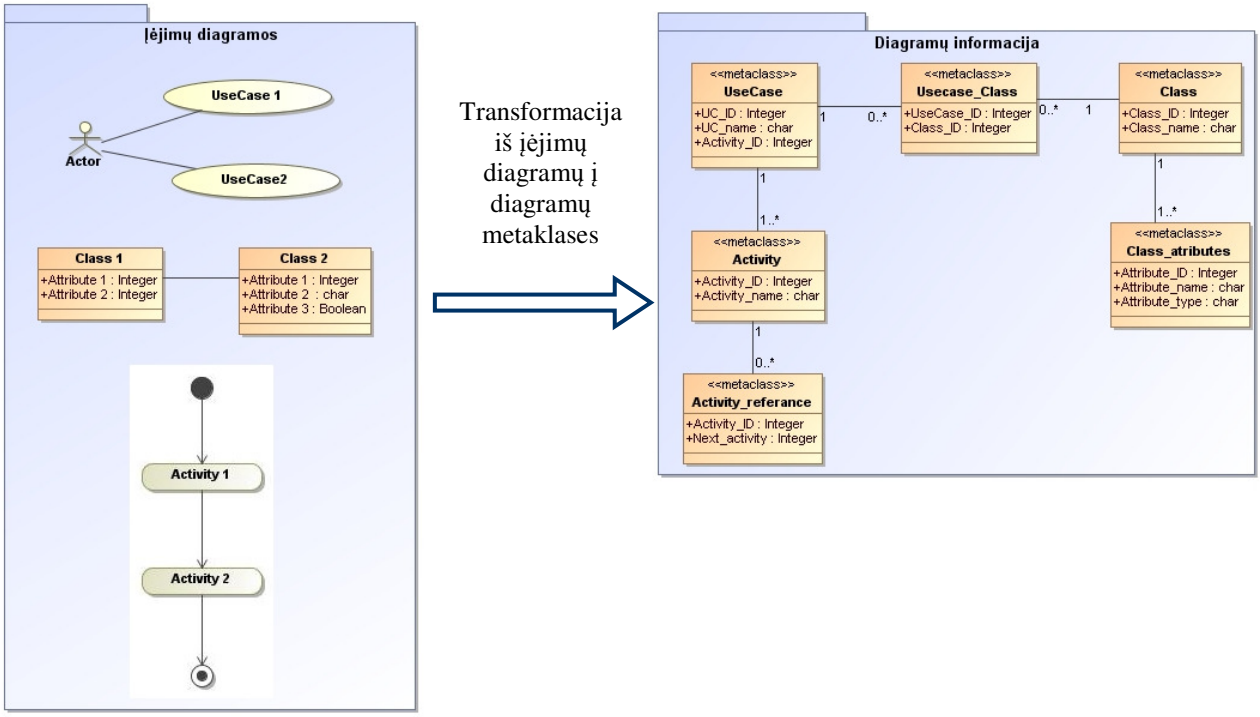
Detali panaudojimo atvejo „Generuoti grafinę vartotojo sąsają“ veiklos diagrama pavaizduota 22 paveiksle.



22 pav. Detali panaudojimo atvejo „Generuoti grafinę sąsają“ veiklos diagrama

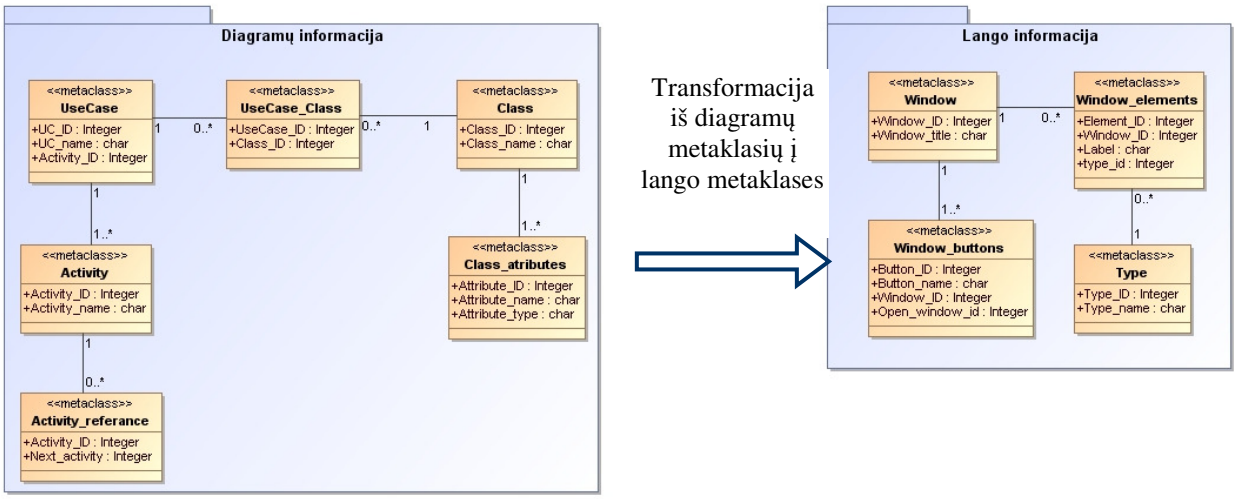
3.1.2 Detalizuotas taikymo procesas

23 paveiksle pavaizduotoje diagramoje matyti, kaip iš nubraižytų panaudojimo atvejų, klasių bei veiklos diagramų informacija saugoma sistemoje. Kiekvienas panaudojimo atvejis ir jo duomenys saugomi lentelėje *UseCase*. Kadangi panaudojimo atvejai susiję su veiklomis, lentelėje saugomas panaudojimo atvejo *pavadinimas*, jo *ID* bei veiklos, su kuria jis siejasi, *ID*. Veiklos diagramos duomenys (*ID* ir *pavadinimas*) saugomi *Activity* lentelėje. Kadangi veiklos nusako sistemos langų navigaciją, *Activity_referance* lentelėje papildomai išsaugomas sekančios veiklos pavadinimas. Informacija apie klases saugoma *Class* lentelėje, klasių atributai – *Class_attributes* lentelėje.



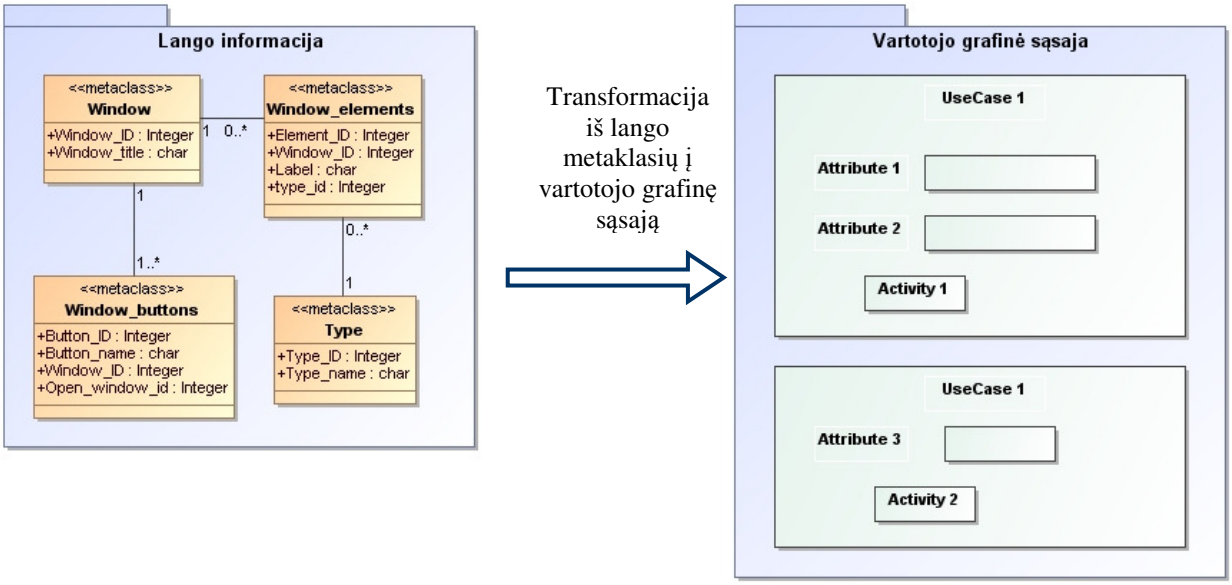
23 pav. Transformacija iš UML diagramų į diagramų metaklases

Kiekvienas panaudojimo atvejis simbolizuoja naują grafinės vartotojo sąsajos langą. Klasių atributų pavadinimai įrašomi į *Window_elements* lentelės *Label* poziciją. Lango elementui parenkamas išsaugotas jo tipas. Veiklos pavadinimai atspindi mygtukų pavadinimus, taigi veiklos pavadinimas įrašomas į *Button_name* poziciją *Window_buttons* lentelėje.



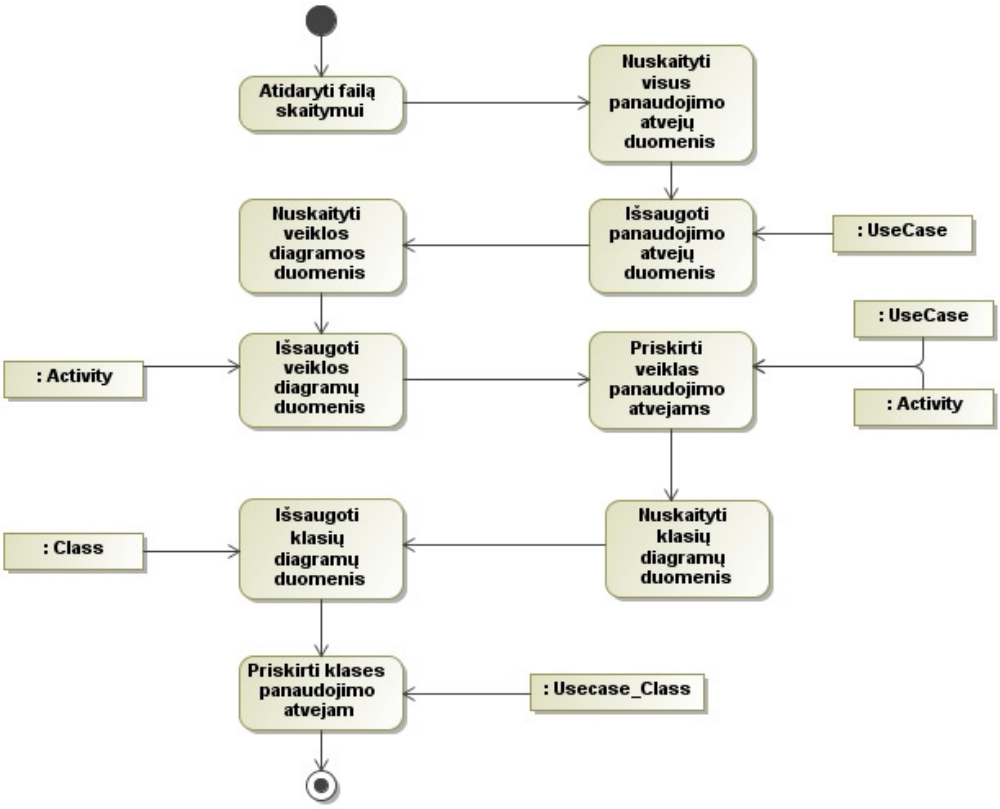
24 pav. Transformacija iš diagramų metaklasių į langų metaklases

25 paveiksle pavaizduota, kaip iš turimos išsaugotos informacijos sistemoje sugeneruojama grafinė vartotojo sąsaja. Kiekvienas lentelėje *Window* aprašytas langas paverčiamas į tikrą sąsajos langą su nurodytu pavadinimu. Langai gali turėti keletą įvairių elementų, kurie aprašyti *Window_elements* lentelėje. Jie atvaizduojami sugeneruotame lange. Taip pat atvaizduojami ir mygtukai.



25 pav. Langų metaklasių transformaciją į grafinės vartotojo sąsajos langus

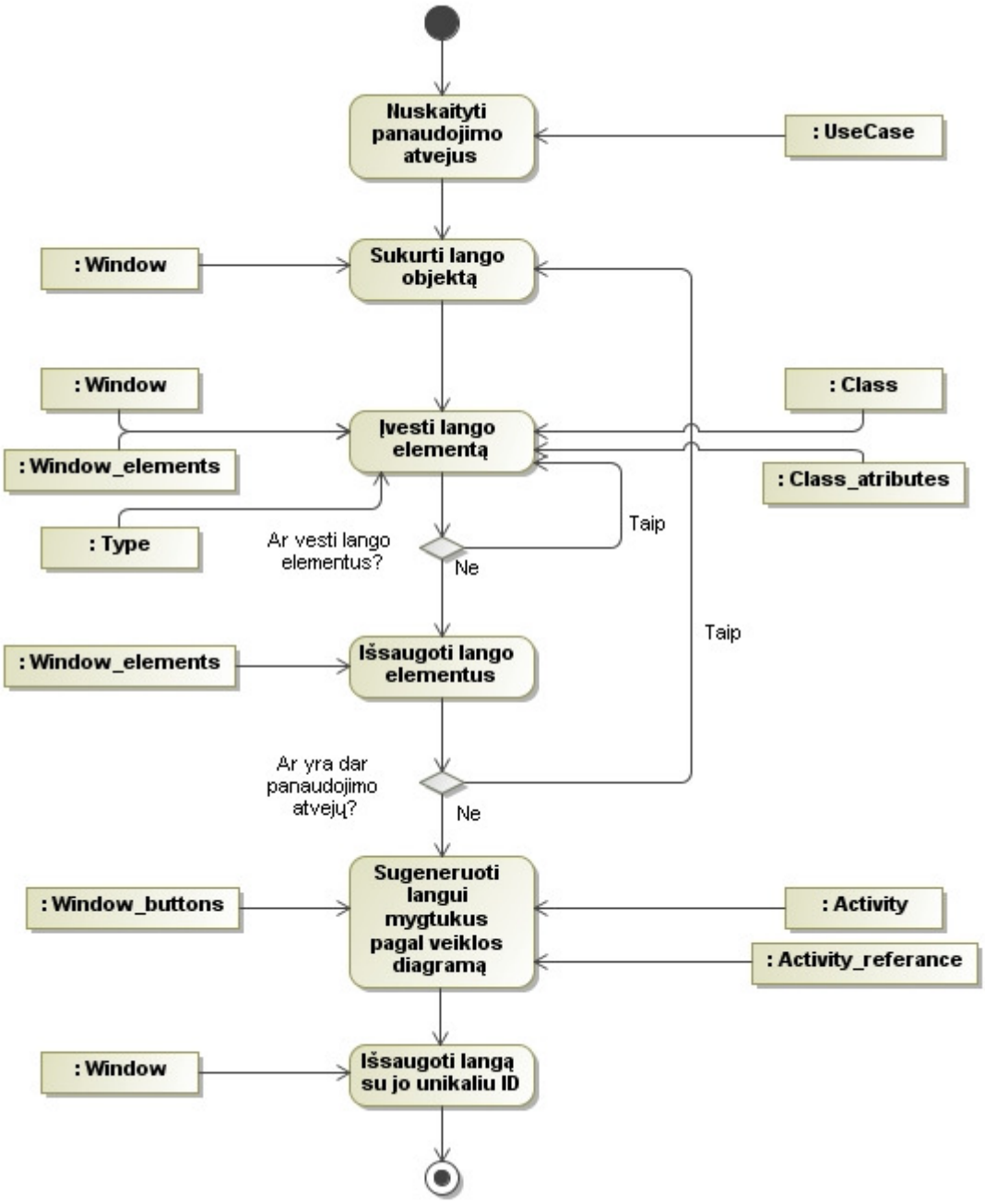
Informacijos įvedimo iš UML diagramų procesas pavaizduotas 26 paveiksle. Įkėlus į sistemą diagramas, pirmiausia nuskaitomi ir išsaugomi panaudojimo atvejų duomenys, vėliau – veiklos diagramos duomenys. Jie tarpusavyje susiejami, t.y. panaudojimo atvejai „surišami“ su veiklomis. Galiausiai nuskaitomi klasių diagramos duomenys ir jie susiejami su panaudojimo atvejais.



26 pav. Diagramų įvedimo į sistemą veiklos procesas

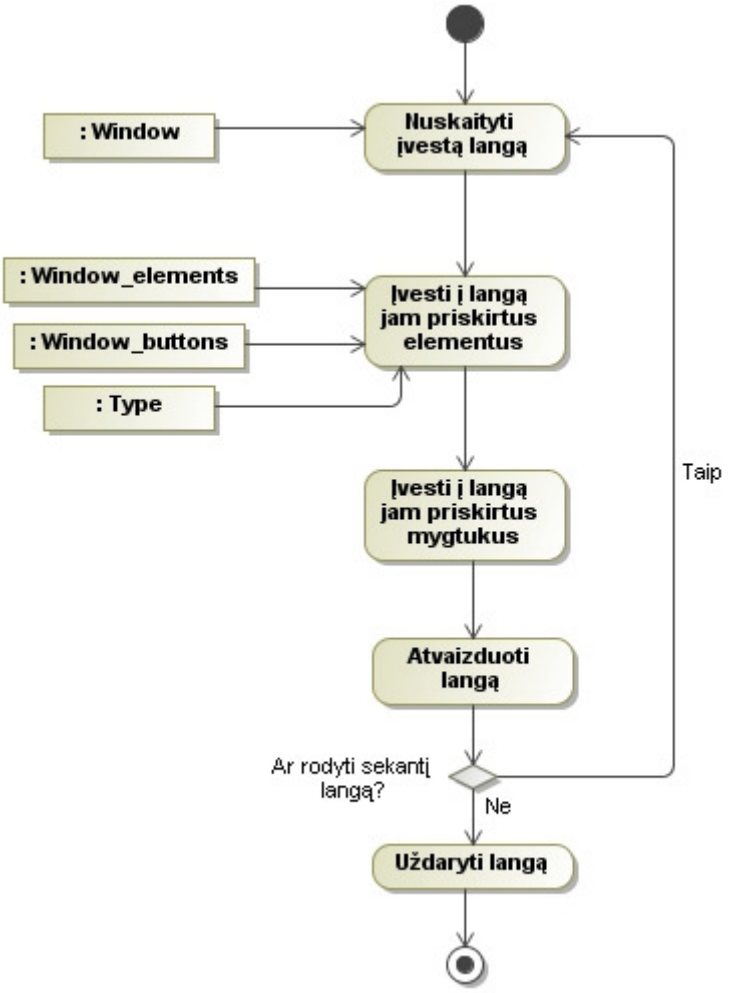
Nuskaičius panaudojimo atvejų duomenis, kiekvienam panaudojimo atvejui sukuriamas grafinės vartotojo sąsajos langas. Sukeliami nurodyti elementai ir jie išsaugomi.

Šis procesas vykdomas tol, kol yra panaudojimo atvejų. Sukurtuose languose įkeliami nurodyti mygtukai, kurių pavadinimus atspindi susietų veiklų pavadinimai. Langai bei informacija apie juos išsaugoma duomenų bazėje.



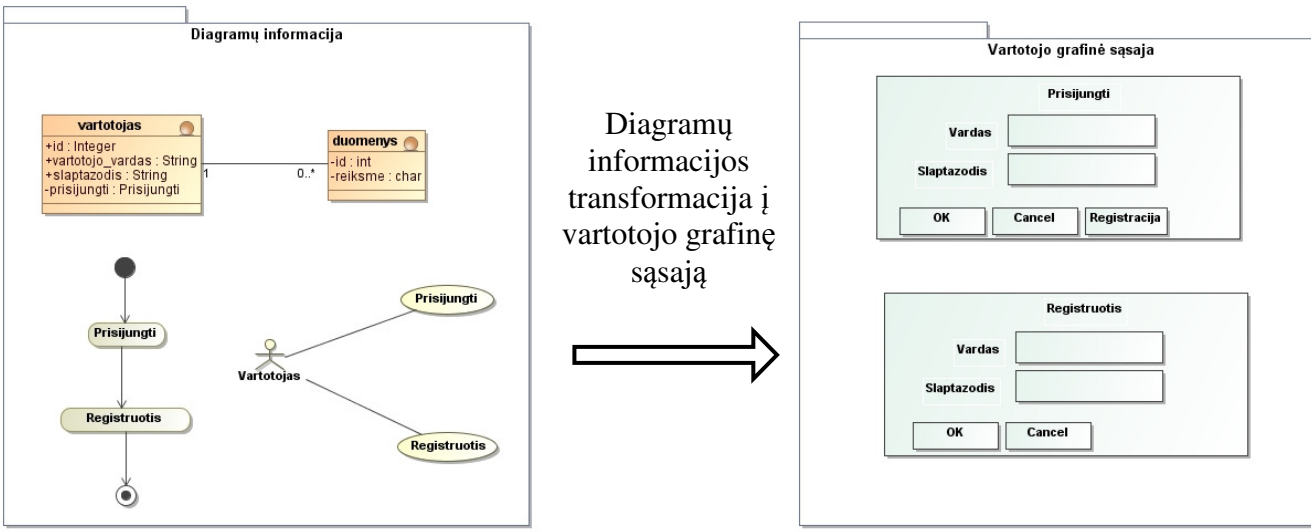
27 pav. Transformacijos iš diagramų elementų į lango elementus procesas

Išėjimo klasių transformacija į grafinę sąsają pavaizduota 28 paveiksle. Nuskaičius įvestus langus, į juos įvedami priskirti elementai bei mygtukai. Langai atvaizduojami. Procesas vyksta tol, kol yra neatvaizduotų langų.



28 pav. Langų atvaizdavimo sistemoje veiklos procesas

29 paveiksle pavaizduota pavyzdinė transformacija iš UML diagramų į grafinę vartotojo sąsają.

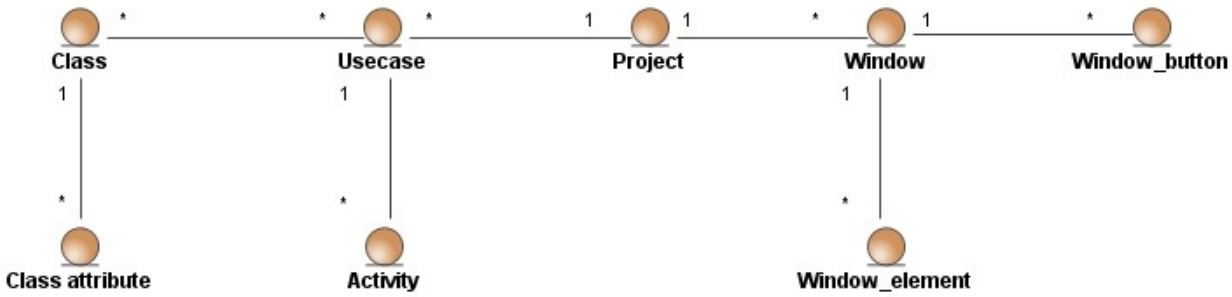


29 pav. Pavyzdinė transformacija iš UML diagramų į grafinę sąsają

3.2 Dalykinės srities modelis

Pagrindinės sistemos esybės yra UML diagramų elementai – klasė bei jos atributas, panaudojimo atvejis, veikla. Langų esybės – langas, lango elementas ir lango mygtukas, kurie priskiriami projektui.

Dalykinės srities esybių modelis pavaizduotas 30 paveiksle.

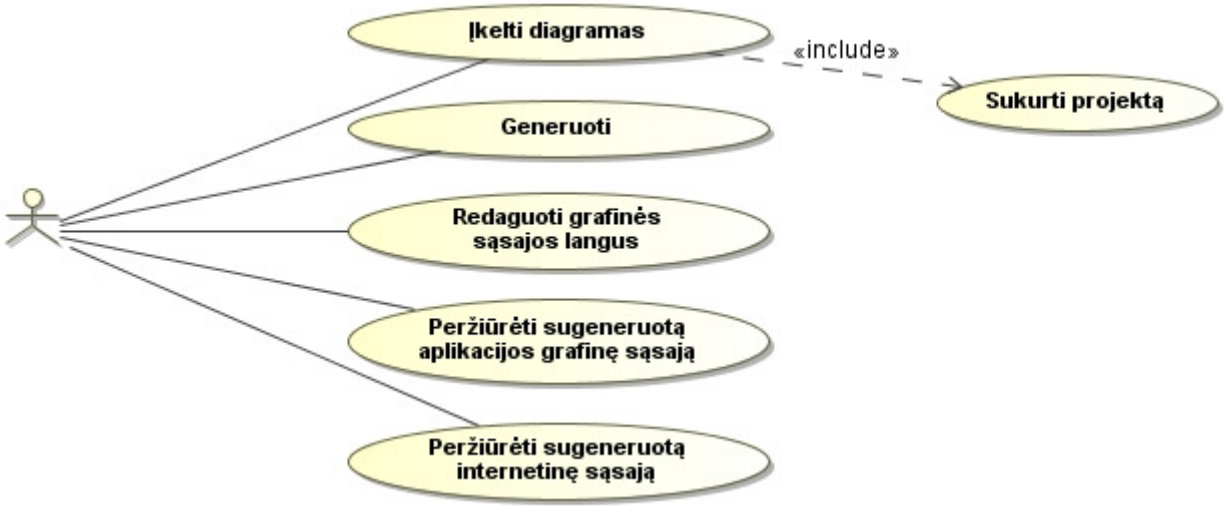


30 pav. Dalykinės srities esybių modelis

3.3 Reikalavimų analizė

Norint sugeneruoti grafinę vartotojo sąsają iš UML diagramų, pirmiausia reikia nubraižyti panaudojimo atvejų, klasių bei veiklos diagramas. Šios diagramos turi būti nubraižytos UML CASE įrankiu. Nubraižytos diagramos įkeliamos į grafinės vartotojo sąsajos generavimo įrankį ir patikrinamas jų korektiškumas.

Grafinės vartotojo sąsajos smulkesni etapai pavaizduoti 31 paveiksle.



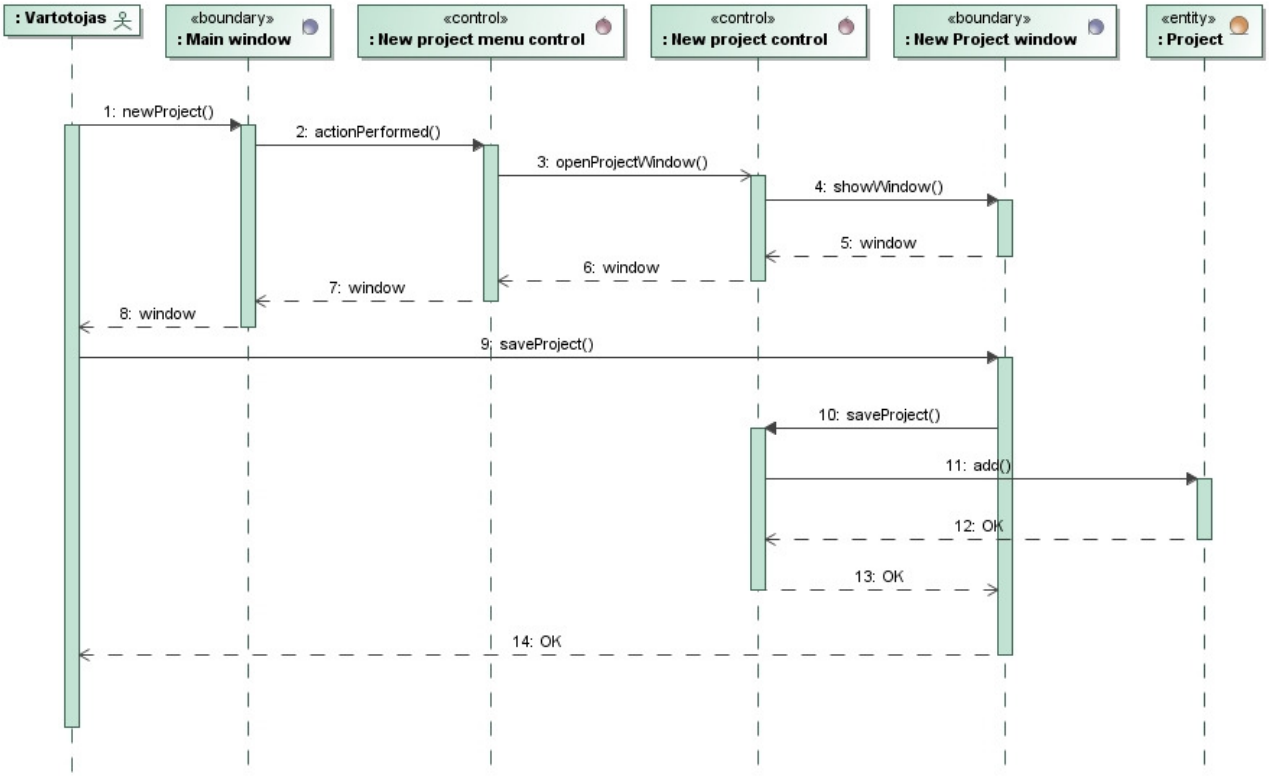
31 pav. Kompiuterizuojamo panaudojimo atvejo smulkesni etapai

Panaudojimo atvejo „Sukurti projektą“ specifikacija pateikiama 10 lentelėje.

10 lentelė. PA „Sukurti projektą“ specifikacija

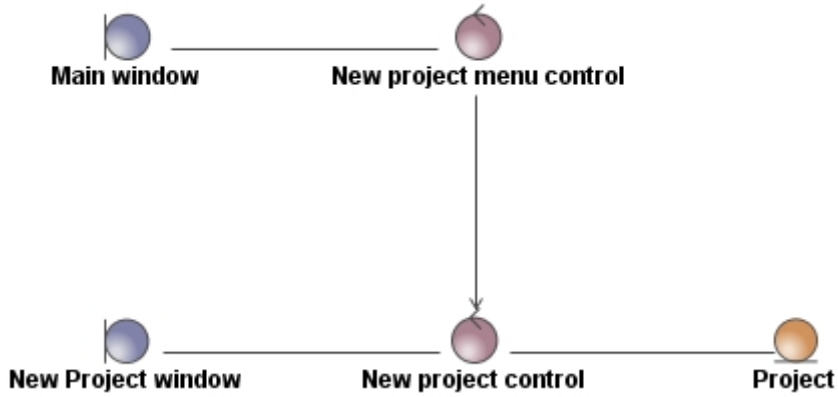
Panaudojimo atvejis	Sukurti projektą
Tikslas	Pradėti darbą su sistema
Aprašymas	Kadangi su sistema galima įgyvendinti keletą projektų, dėl patogumo kiekvienam iš jų sukuriama atskira byla sistemoje
Prieš sąlyga	Vartotojas meniu pasirenka naujo projekto sukūrimo funkciją
Aktorius	Vartotojas
Sistema	Grafinės vartotojo sąsajos generavimas naudojant UML diagramas
Sužadinimo sąlyga	1.Vartotojas pasirenka naujo projekto sukūrimo funkciją
Veiklos taisyklės	-
Ryšiai su kitais PA	Apima PA „Įkelti diagramas“
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda naujo projekto pavadinimą	1.1. Sukuriamas naujas projektas su nurodytu pavadinimu
Po sąlyga	Sukurtas naujas projektas
Alternatyvos (nesėkmės atvejai)	1.1.a. Vartotojas neįveda naujo projekto pavadinimo
Specialūs (nefunkciniai) reikalavimai	-

Panaudojimo atvejo „Sukurti projektą“ sekų diagrama pateikiama 32 paveiksle.



32 pav. PA „Sukurti projektą“ sekų diagrama

Panaudojimo atvejo „Sukurti projektą“ analizės klasių diagrama pateikiama 33 paveiksle.



33 pav. PA „Sukurti projektą“ analizės klasių diagrama

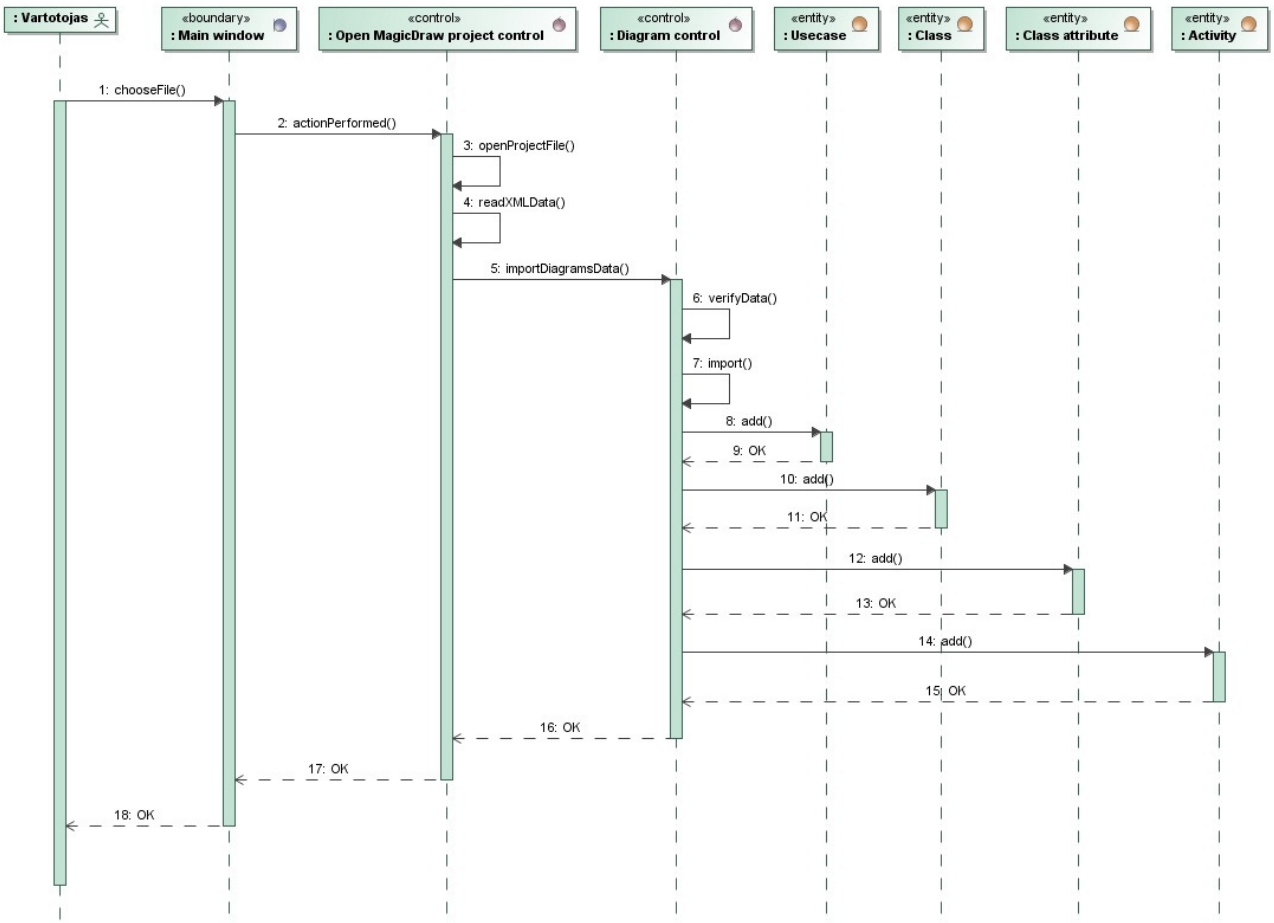
Panaudojimo atvejo „Įkelti diagramas“ specifikacija pateikiama 11 lentelėje.

11 lentelė. PA „Įkelti diagramas“ specifikacija

Panaudojimo atvejis	Įkelti diagramas
Tikslas	Įkelti nubraižytas UML diagramas į sistemą
Aprašymas	Grafinės vartotojo sąsajos generavimui naudojamos UML diagramos. Jas reikia įkelti į sistemą.
Prieš sąlyga	Vartotojas meniu pasirenka diagramų įkėlimo funkciją
Aktorius	Vartotojas
Sistema	Grafinės vartotojo sąsajos generavimas naudojant UML

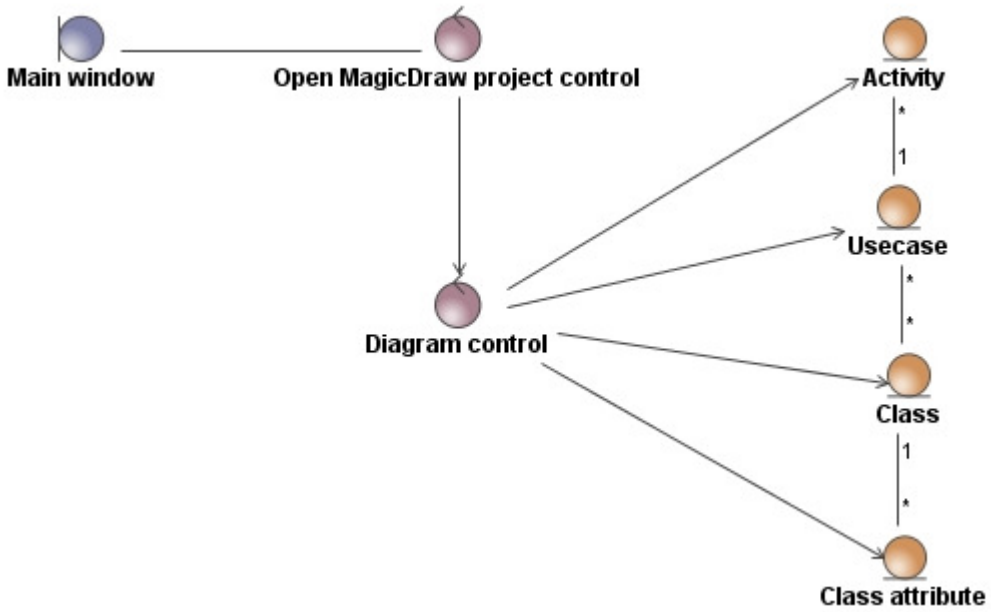
	diagramas
Sužadinimo sąlyga	1.Vartotojas pasirenka diagramų įkėlimo į sistemą funkciją.
Veiklos taisyklės	-
Ryšiai su kitais PA	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka UML diagramas aprašantį failą savo kompiuteryje	1.1. Nurodyta byla su UML diagramų informacija yra įkeliama į sistemą
Po sąlyga	Diagramos įkeltos į sistemą
Alternatyvos (nesėkmės atvejai)	1.1.a. Vartotojas neįkelia nubraižytas UML diagramas aprašančios XML bylos
Specialūs (nefunkciniai) reikalavimai	-

Panaudojimo atvejo „Įkelti diagramas“ sekų diagrama pateikiama 34 paveiksle.



34 pav. PA „Įkelti diagramas“ sekų diagrama

Panaudojimo atvejo „Įkelti diagramas“ analizės klasių diagrama pateikiama 35 paveiksle.



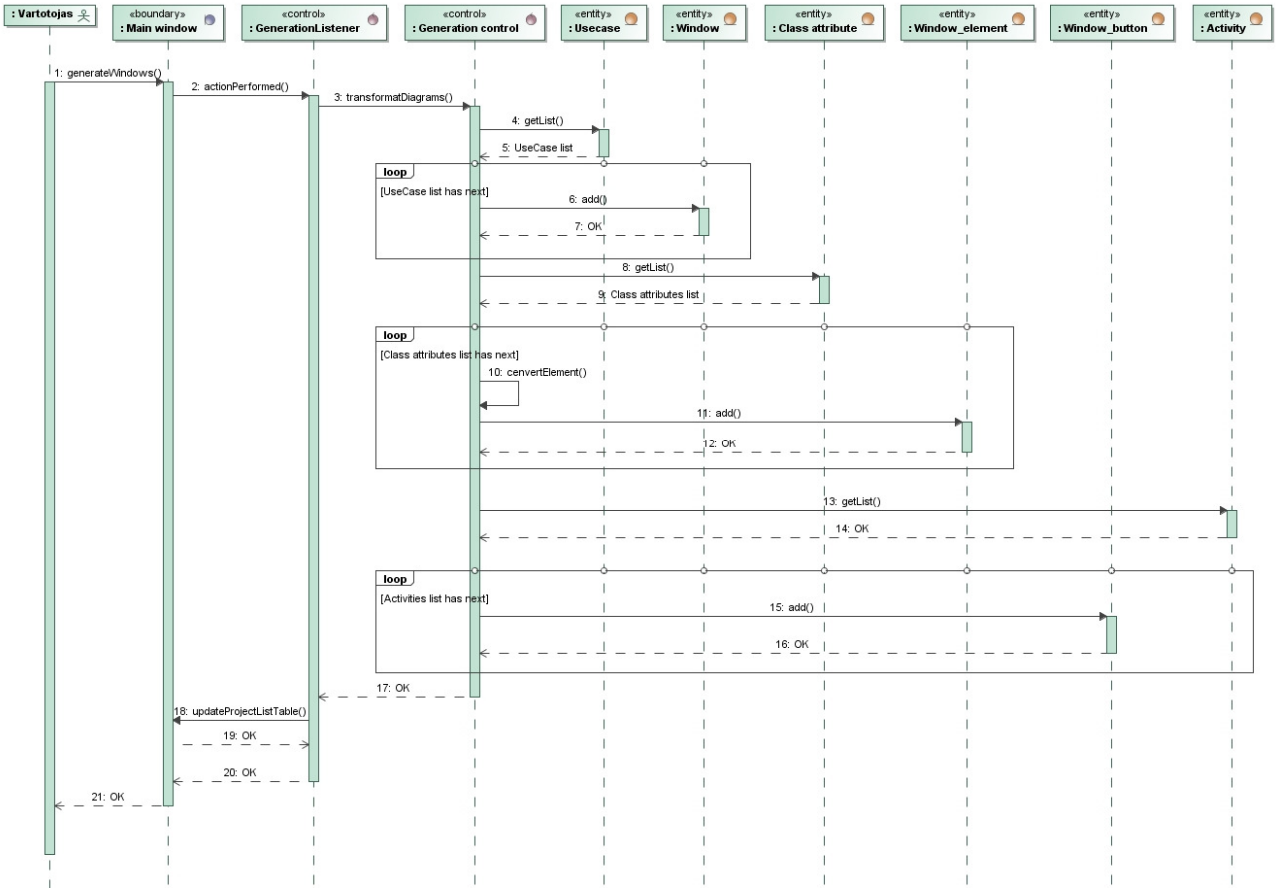
35 pav. PA „Įkelti diagramas“ analizės klasių diagrama

Panaudojimo atvejo „Generuoti“ specifikacija pateikiama 12 lentelėje.

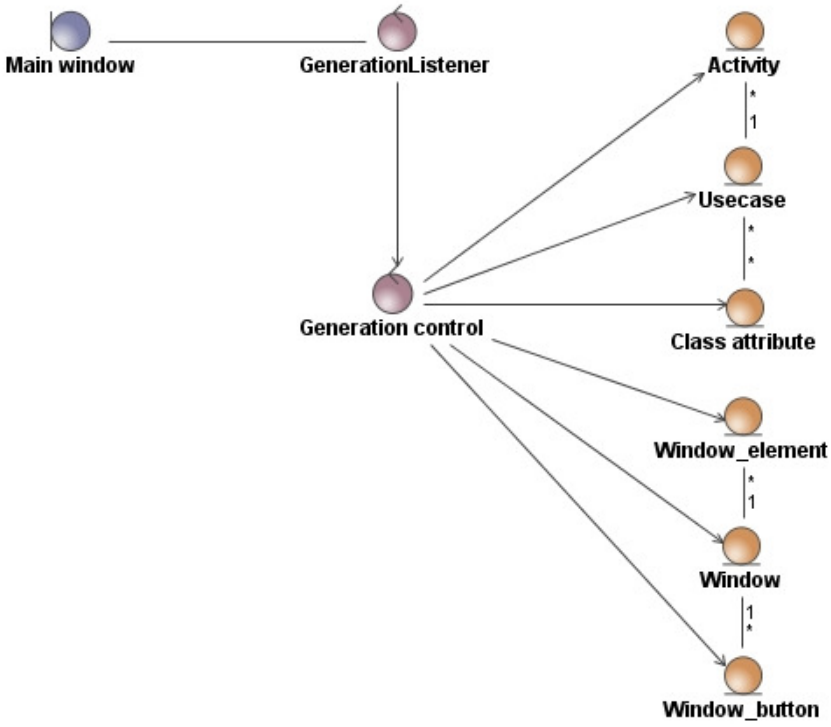
12 lentelė. PA „Generuoti“ specifikacija

Panaudojimo atvejis	Generuoti
Tikslas	Sugeneruoti grafinę vartotojo sąsają
Aprašymas	Grafinė vartotojos sąsaja generuojama naudojantis įkeltais į įrankį duomenimis.
Prieš sąlyga	Vartotojas meniu pasirenka generavimo funkciją
Aktorius	Vartotojas
Sistema	Grafinės vartotojo sąsajos generavimas naudojant UML diagramas
Sužadinimo sąlyga	1.Vartotojas pasirenka generavimo funkciją
Veiklos taisyklės	-
Ryšiai su kitais PA	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1.Vartotojas pasirenka generavimo funkciją	1. Sugeneruojama grafinė sąsaja, vartotojas apie tai informuojamas žinute.
Po sąlyga	Sugeneruota grafinė vartotojo sąsaja
Alternatyvos (nesėkmės atvejai)	
Specialūs (nefunkciniai) reikalavimai	-

Panaudojimo atvejo „Generuoti“ sekų diagrama pateikiama 36 paveiksle.



36 pav. PA „Generuoti“ sekų diagrama



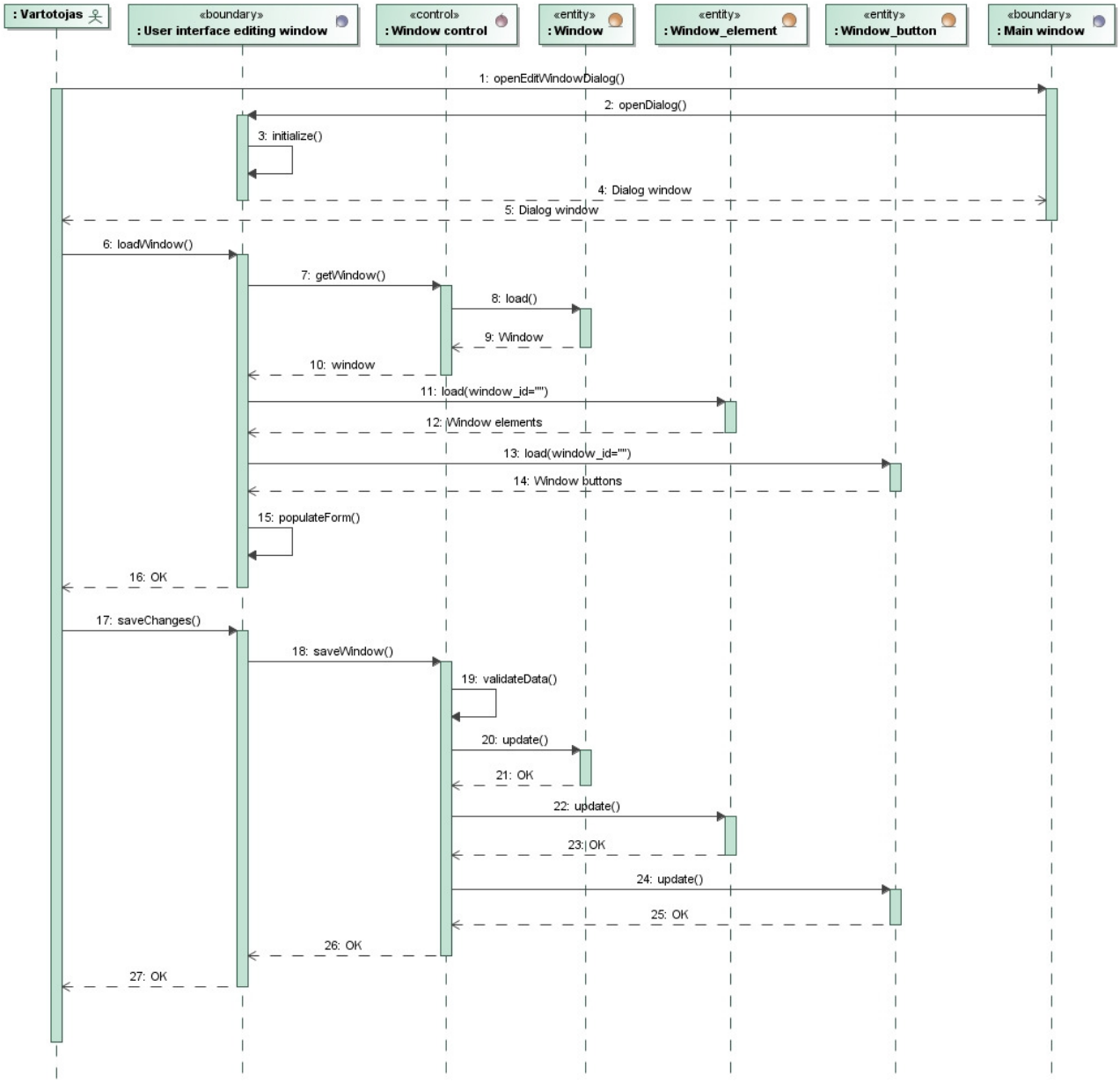
37 pav. PA „Generuoti“ analizės klasių diagrama

Panaudojimo atvejo „Redaguoti grafinės sąsajos langus“ specifikacija pateikiama 13 lentelėje.

13 lentelė. PA „Redaguoti grafinės sąsajos langus“ specifikacija

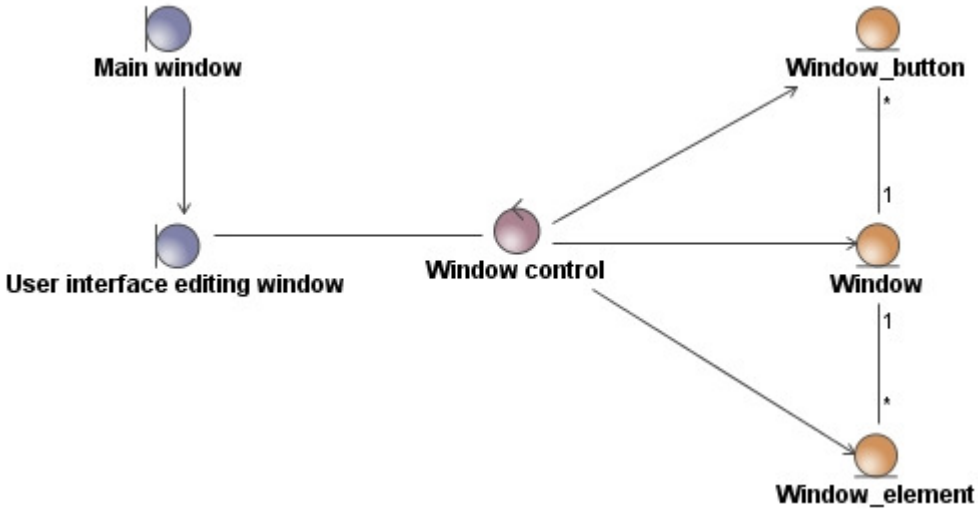
Panaudojimo atvejis	Redaguoti grafinės sąsajos langus
Tikslas	Redaguoti sugeneruotus rafinės sąsajos langus
Aprašymas	Suteikiama galimybė sukurti ar redaguoti naujus elementus, mygtukus
Prieš sąlyga	Vartotojas meniu pasirenka langų redagavimo funkciją
Aktorius	Vartotojas
Sistema	Grafinės vartotojo sąsajos generavimas naudojant UML diagramas
Sužadinimo sąlyga	1.Vartotojas pasirenka langų redagavimo funkciją
Veiklos taisyklės	-
Ryšiai su kitais PA	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1.Vartotojas pasirenka langą iš pateiktų sugeneruotų sistemos langų	1. Pateikiami lango elementai, jo parametrai bei lango mygtukai, kuriuos galima redaguoti
Po sąlyga	Išsaugoti pasirinkti papildomi nustatymai
Alternatyvos (nesėkmės atvejai)	1. a. Vartotojas neišsaugo nurodytų duomenų arba ištrina elemento/mygtuko pavadinimą
Specialūs (nefunkciniai) reikalavimai	-

Panaudojimo atvejo „Redaguoti grafinės sąsajos langus“ sekų diagrama pateikiama 38 paveiksle.



38 pav. PA „Redaguoti grafinės sąsajos langus“ sekų diagrama

Panaudojimo atvejo „Redaguoti grafinės sąsajos langus“ analizės klasių diagrama pateikiama 39 paveiksle.



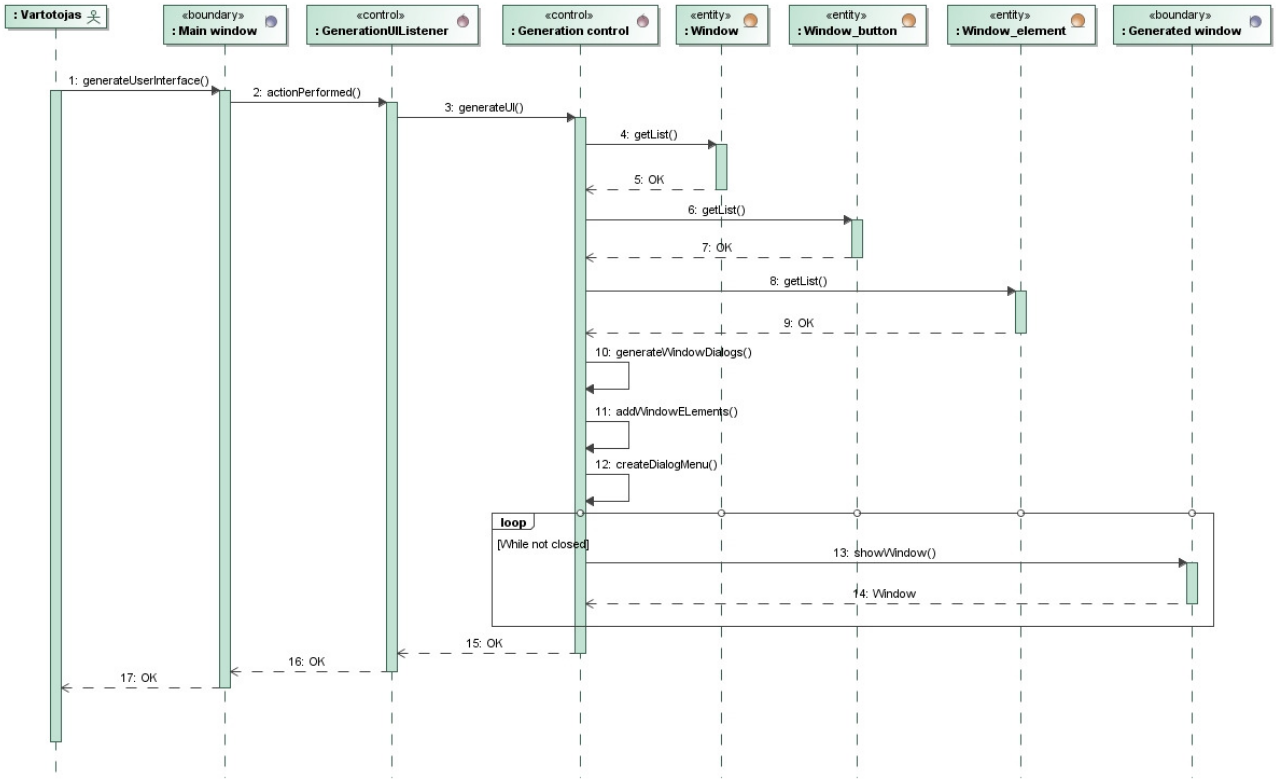
39 pav. PA „Redaguoti grafinės sąsajos langus“ analizės klasių diagrama

Panaudojimo atvejo „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ specifikacija pateikiama 14 lentelėje.

14 lentelė. PA „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ specifikacija

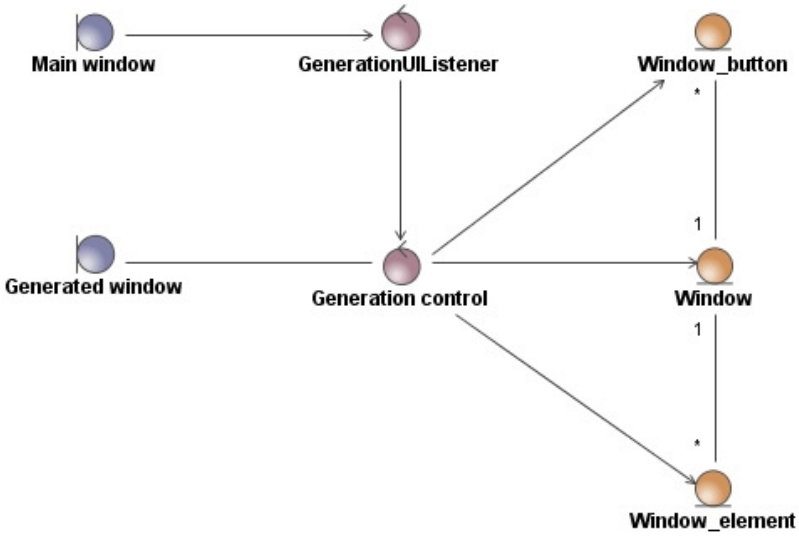
Panaudojimo atvejis	Peržiūrėti aplikacijos grafinę vartotojo sąsają
Tikslas	Peržiūrėti sugeneruotą aplikacijos grafinę vartotojo sąsają
Aprašymas	Įkėlus diagramas, galima peržiūrėti sugeneruotą grafinę vartotojo sąsają pagal įkeltus į sistemą duomenis
Prieš sąlyga	Vartotojas meniu pasirenka grafinės sąsajos peržiūros funkciją
Aktorius	Vartotojas
Sistema	Grafinės vartotojo sąsajos generavimas naudojant UML diagramas
Sužadinimo sąlyga	1.Vartotojas pasirenka grafinės vartotojo sąsajos peržiūros funkciją
Veiklos taisyklės	-
Ryšiai su kitais PA	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1.Vartotojas pasirenka grafinės sąsajos peržiūros funkcija	1. Parodomi sugeneruoti grafinės sąsajos langai.
Po sąlyga	Peržiūrėti sugeneruoti sistemos langai
Alternatyvos (nesėkmės atvejai)	-
Specialūs (nefunkciniai) reikalavimai	-

Panaudojimo atvejo „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ sekų diagrama pateikiama 40 paveiksle.



40 pav. PA „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ sekų diagrama

Panaudojimo atvejo „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ analizės klasių diagrama pateikiama 41 paveiksle.



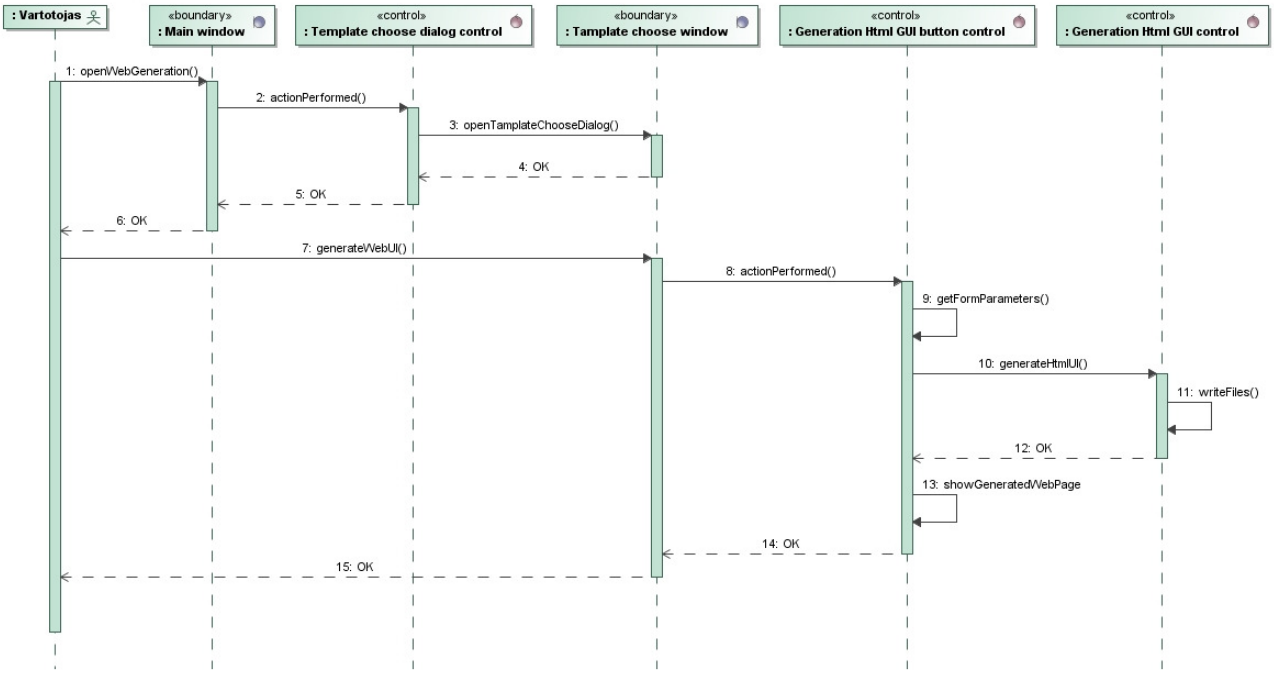
41 pav. PA „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ analizės klasių diagrama

Panaudojimo atvejo „Peržiūrėti internetinę grafinę vartotojo sąsają“ specifikacija 15 lentelėje.

15 lentelė. PA „Peržiūrėti aplikacijos grafinę vartotojo sąsają“ specifikacija

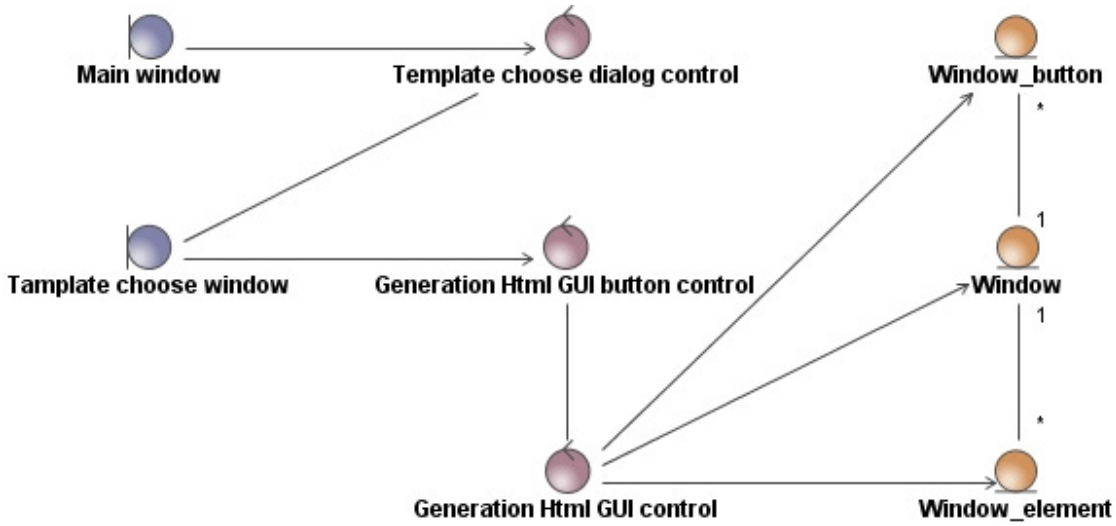
Panaudojimo atvejis	Peržiūrėti internetinę grafinę vartotojo sąsają
Tikslas	Peržiūrėti sugeneruotą internetinę grafinę vartotojo sąsają
Aprašymas	Įkėlus diagramas, galima peržiūrėti sugeneruotą grafinę vartotojo sąsają pagal įkeltus į sistemą duomenis
Prieš sąlyga	Vartotojas meniu pasirenka grafinės vartotojo sąsajos peržiūros funkciją
Aktorius	Vartotojas
Sistema	Grafinės vartotojo sąsajos generavimas naudojant UML diagramas
Sužadinimo sąlyga	1.Vartotojas pasirenka internetinės grafinės vartotojo sąsajos peržiūros funkciją
Veiklos taisyklės	-
Ryšiai su kitais PA	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1.Vartotojas pasirenka grafinės vartotojo sąsajos peržiūros funkcija	1. Parodomi sugeneruoti internetinės grafinės sąsajos langai
Po sąlyga	Peržiūrėti sugeneruoti internetinės sistemos langai
Alternatyvos (nesėkmės atvejai)	-
Specialūs (nefunkciniai) reikalavimai	-

Panaudojimo atvejo „Peržiūrėti internetinę grafinę vartotojo sąsają“ sekų diagrama pateikiama 42 paveiksle.



42 pav. PA „Peržiūrėti internetinę grafinę vartotojo sąsają“ sekų diagrama

Panaudojimo atvejo „Peržiūrėti internetinę grafinę vartotojo sąsają“ analizės klasių diagrama pateikiama 43 paveiksle.



43 pav. PA „Peržiūrėti internetinę grafinę vartotojo sąsają“ analizės klasių diagrama

3.4 Reikalavimų analizės apibendrinimas

1. Pastebėta, kad panaudojimo atvejus „Sukurti pradinę IS grafinę sąsają“ ir „Programuoti IS grafinę sąsają“ kompiuterizuojamame modelyje apibendrina vienas panaudojimo atvejis „Generuoti grafinę vartotojo sąsają“.
2. Vartotojo grafinei sąsajai generuoti reikalingos panaudojimo atvejų, klasių ir veiklos diagramos.
3. Diagramas siūloma braižyti pasinaudojus CASE įrankiu.

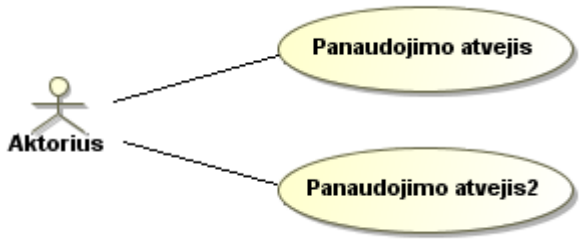
4. ALGORITMO APRAŠAS

4.1 UML diagramos ir jų ryšys su grafine vartotojo sąsaja

4.1.1 Panaudojimo atvejų diagrama

Panaudojimo atvejų diagrama vaizduoja sistemos funkcionalumą. Ji išskaido sistemos funkcijas į atskirus panaudojimo atvejus ir jais atspindi, ką sistema daro, tačiau nenurodo, kaip tai įgyvendinama. Panaudojimo atvejų diagrama vaizduoja sistemos elgseną išorinių vartotojų atžvilgiu.

Pagrindiniai šioje diagramoje naudojami elementai – aktorius, panaudojimo atvejis ir jų tarpusavio ryšiai (44 pav.). Aktoriumi nebūtinai turi būti realus žmogus – aktoriumi vaizduojama rolė, taigi juo gal būti ir kita sistema. Kiekvienas panaudojimo atvejis atspindi vieną iš sistemos funkcionalumų.



44 pav. Panaudojimo atvejų ir aktoriaus pavaizdavimas

Panaudojimo atvejai su aktoriais ir kitais panaudojimo atvejais sujungiami šiais ryšiais: asociacijos, apibendrinimo, įtraukimo (angl. *include*) arba išplėtimo (angl. *extend*) ryšiu. Jų aprašymai pateiktas 16 lentelėje.

16 lentelė. Panaudojimo atvejų diagramos ryšiu aprašymas

Ryšio pavadinimas	**Atvaizdavimas**	**Aprašymas**
Asociacija		Ryšys, kuris naudojamas sujungti aktorius su panaudojimo atvejais. Jis reiškia, kad toks aktorius gali vykdyti tam tikrą panaudojimo atvejį.
Apibendrinimo		Apibendrinimo ryšys naudojamas sujungti tėvinio ir vaikiniui panaudojimo atvejams, kai vaikinio panaudojimo atvejo scenarijus paveldi dalį tėvo scenarijaus. Apibendrinimo ryšiu gali būti sujungti ir aktoriai. Vaikinis aktoriaus elementas gali atlikti tėviniui elementui priskirtus panaudojimo atvejus.

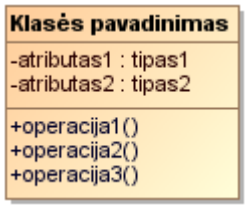
Įtraukimo ryšys		Įtraukimo ryšys tarp A ir B panaudojimo atvejų reiškia, kad panaudojimo atvejis A gali būti atliktas tik atlikus panaudojimo atvejį B.
Išplėtimo ryšys		Tai ryšys, naudojamas sujungti du panaudojimo atvejus. Tėvinis panaudojimo atvejis yra išplečiamas tam tikra papildoma elgsena. Paprastai nurodoma, kokia sąlyga galioja išplėtimo veiksmui.

Panaudojimo atvejų diagrama kuriamame grafinės vartotojo sąsajos generavimo įrankyje padės identifikuoti grafinės vartotojo sąsajos langus.

4.1.2 Klasių diagrama

Klasių diagrama – tai statinės struktūros modelis, grafiškai atvaizduojantis klases, jų ryšius ir apribojimus. Kiekviena klasė atitinka kuriamos sistemos objektą. Vienai sistemai gali būti sudaromos kelios klasių diagramos, kiekviena klasė gali būti naudojama keliose diagramose. Klases taip pat galima suskirstyti į paketus.

Kiekviena klasė gali turėti savo pavadinimą, atributus ir operacijas (45 pav.).



45 pav. Klasės pavaizdavimas

Klasės tarpusavyje gali būti sujungtos ryšiais: asociacija, apibendrinimas, agregavimas, klasifikacija, specializacija, kompozicija ar rekursyviuoju ryšiu. Plačiau ryšiai aprašyti 17 lentelėje.

17 lentelė. Klasių ryšių aprašas

Ryšio pavadinimas	**Atvaizdavimas**	**Aprašymas**
Asociacija		Paprasčiausias ryšys tarp klasių. Šis ryšys taip pat reiškia, kad yra sujungti ir sujungtų klasių objektai. Asociacijos galai turi vardus, kurie rodo, kokį vaidmenį (rolę) tame ryšyje atlieka objektas.
Klasifikacija		Klasifikacijos ryšys rodo objekto priklausomybę tam tikrai klasei. Šis ryšys retai naudojamas diagramose,

		bet jis egzistuoja tarp kiekvienos klasės ir jos objektų.
Apibendrinimas		Apibendrinimo ryšys yra ryšys tarp specifiško tipo ir bendresnio tipo klasių. Bendresnio tipo klasė šiuo atveju ne tik paveldi specifiško tipo klasės savybes, bet gali turėti ir savo savybių. Šiam ryšiui priešingas yra specializavimo ryšys.
Agregavimas		Agregavimo ryšys reiškia, kad sudėtinis objektas (agregatas) susideda iš kitų objektų.
Kompozicija		Kompozicija yra agregavimo ryšio forma. Komponentas gali priklausyti tik vienam sudėtiniam objektui ir komponentinio objekto gyvavimas visiškai priklauso nuo sudėtinio objekto.
Rekursyvusis ryšys		Tai ryšys tarp tos pačios klasės egzempliorių.

Kartu su klasių ryšiais nurodomas ir ryšio kardinalumas. Tai - ryšio egzempliorių skaičius, kurį vienos klasės objektas gali turėti su kitos klasės objektais.

Klasių diagrama įrankyje reikalinga langų elementams aprašyti.

4.1.3 Veiklos diagrama

Veiklos diagrama yra srautų diagrama, rodanti valdymo srautų perėjimus nuo vienos veiklos prie kitos. Paprastai ši diagrama papildo panaudojimo atvejų diagramą, nurodant veiksmų eiliškumą. Kiekviena veiklos diagrama turi pradžios ir pabaigos taškus, veiklas ir ryšius tarp jų (46 pav.). Veiklos gali vykti lygiagrečiai, taip pat gali būti panaudoti sprendimų taškai.

46 pav. Veiklos diagramos pagrindiniai elementai ir jų sujungimas

Veiklos diagrama įrankyje vaizduoja navigaciją tarp sugeneruotos grafinės vartotojo sąsajos langų.

4.2 Apribojimai diagramoms

Panaudojimo atvejų diagramoje būtinai turi būti:

- bent vienas panaudojimo atvejis
- bent vienas aktorius.

Klasių diagramoje turi būti:

- bent viena klasė su bent vienu atributu.
- kiekviena klasė turi būti susieta su bent vienu panaudojimo atveju
- sukuriamas atributas, kurio tipo pavadinimas sutampa su panaudojimo atvejo, su kuriuo klasė yra sujungta, pavadinimu.3
- turi būti ryšiai tarp klasių.

Veiklos diagramos apribojimai:

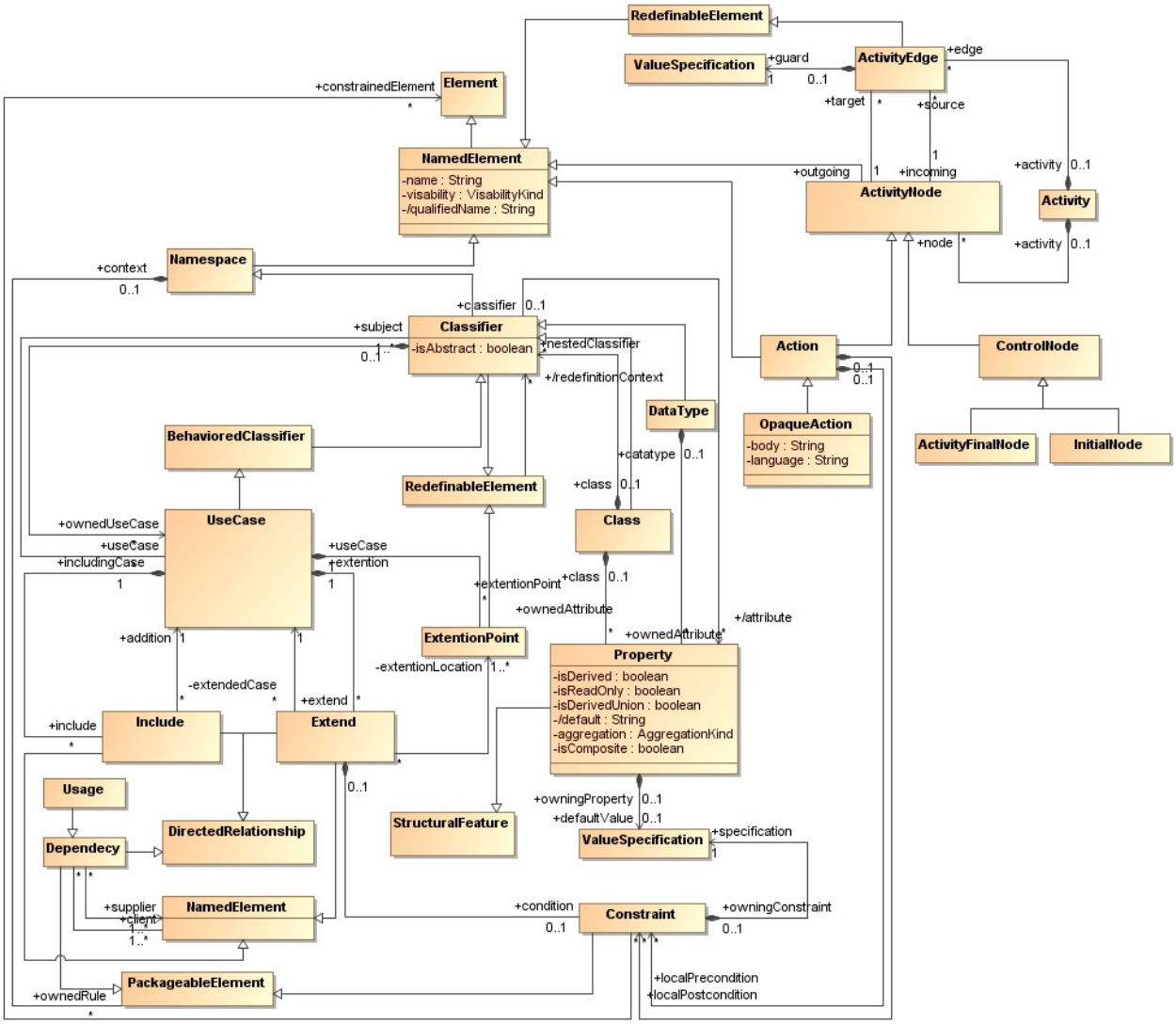
- kiekviena veikla turi sietis su bent vienu panaudojimo atveju.
- Veiklos diagrama turi turėti savo pradžią ir pabaigą
- Veiklos diagrama turi būti sudaryta bent iš vienos veiklos.

4.3 Naudojamų UML diagramų metamodelis

Remiantis UML 2.0 metamodelio aprašymu bus sudaryti metamodeliai diagramoms, kurias naudojamos kuriant programinę įrangą grafinės vartotojo sąsajos generavimui.

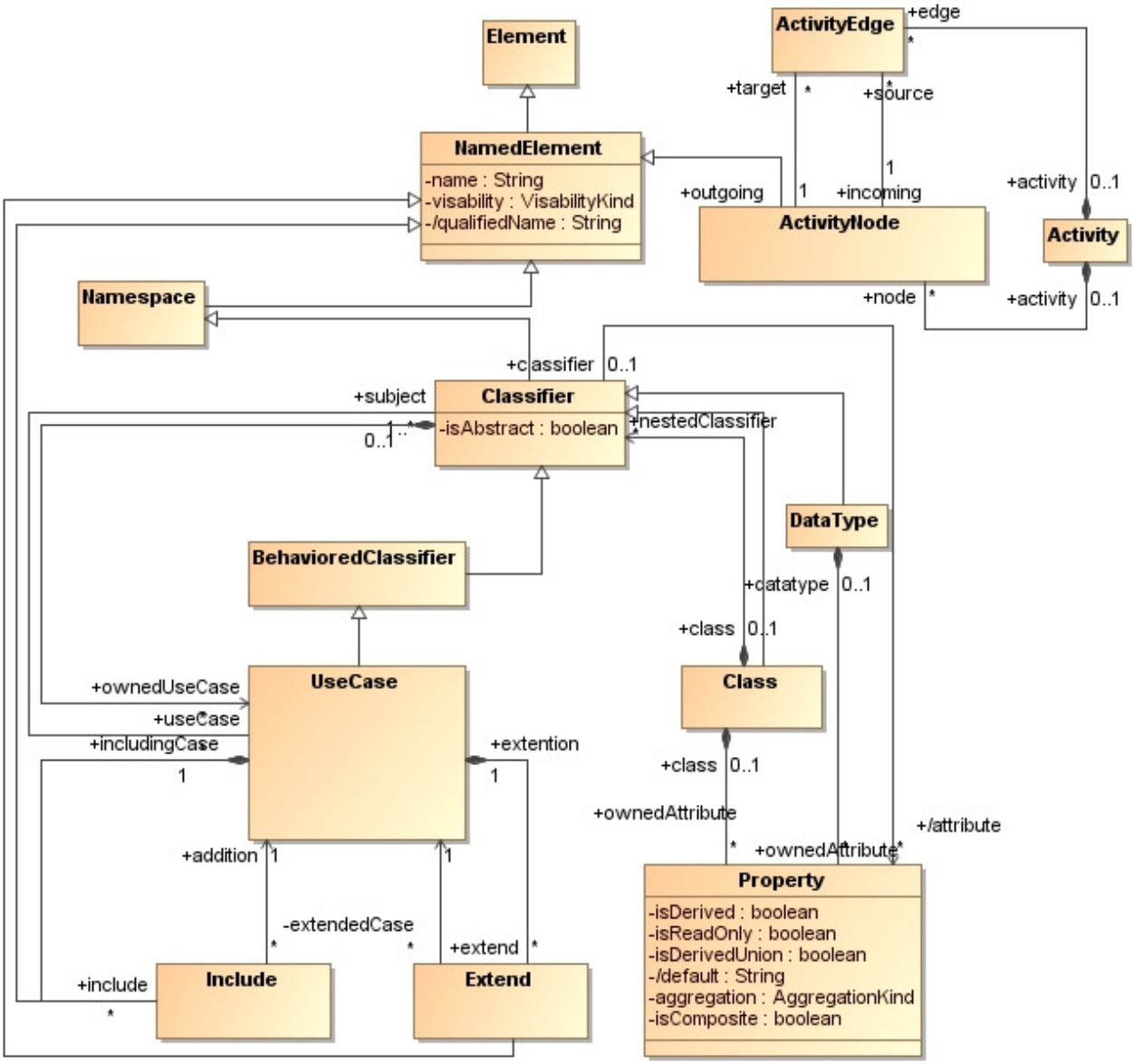
UML metamodelį sudaro trys didesnės elementų kategorijos [4]: klasifikatoriai (angl. *classifier*), įvykiai (angl. *events*) ir elgsenos (angl. *behaviors*). Klasifikatorius apima aprašomų objektų rinkinį. Objektas yra unikalus elementas, kuris turi būseną bei ryšius su kitais objektais. Įvykiai aprašo galimus įvykti reiškinius. Reiškinys šiuo atveju yra tam tikra eiliškumą turinti įvykių seka, susijusi su projektuojama sistema. Elgsenos aprašo įvykių baigtis, susijusias su sistemoje veikiančiu algoritmu atsižvelgiant į jo žingsnius ir taisykles, pagal kurias tas algoritmas yra vykdomas.

Kadangi UML 2.0 metamodeliai yra universalūs ir gali būti naudojami įvairiausiose situacijose, 47 paveiksle pavaizduotos atrinktos UML metamodelio panaudojimo atvejų, klasių ir veiklų klasės bei klasės, susijusios su minėtomis diagramomis. Tai yra duomenys, kurie įkeliami į kuriamą įrankį. Šios metamodelio klasės bus transformuojamos į langų metaklases.



47 pav. Abstraktesnis UML metamodelio fragmentas su naudojamais elementais

Pagrindinės klasės, kurios naudojamos šiame darbe yra *UseCase*, *Class*, *Property*, *Activity*, *ActivityNode*, *ActivityEdge*. Jos, kartu su paveldimumo ryšiais, atvaizduotos 48 paveiksle.



48 pav. Žemesnio lygmens UML metamodelis

48 paveiksle pavaizduotų metaklasių detalesnis aprašas pateiktas 18 lentelėje.

18 lentelė. Klasių metamodelio klasių aprašymas

Pavadinimas	**Atributai, jų tipai ir reikšmės**	**Aprašymas**
Element	Nėra.	Tai sudėtinė modelio dalis, kuri gali turėti kitus elementus. *Element* yra abstrakti metaklasė, kuri naudojama kaip apibendrinanti klasė kitoms metaklasėms.
NamedElement	name: string – *NamedElement* pavadinimas	Tai modelio elementas, kuris gali turėti savo pavadinimą. Šis

	visability: VisabilityKind – apibūdina *NamedElement* vietą *NameSpace* visuose modeliuose. /qualifiedName: String – pavadinimas, leidžiantis identifikuoti *NamedElement* vietą *NameSpace* hierarchijoje.	elementas naudojamas elemento identifikavimui *NameSpace*, kurioje jis yra aprašytas. *NamedElement* – abstrakti metaklasė.
NameSpace	Nėra.	Modelio elementas, kuris susideda iš rinkinio elementų, turinčių pavadinimus, ir galinčių būti identifikuoti pagal juos. Kiekvienas *NamedElement* turi būti aprašytas bent vienoje *NameSpace*. *NameSpace* gali turėti savo apribojimus.
Classifier	Nėra.	Abstrakti metaklasė, kuri apibūdina įvykių rinkinį, atsižvelgiant į jų ypatybes. *Classifier* taip pat yra tipas ir gali turėti savus apibendrinimus, norint apibrėžti kitų klasifikatorių apibendrinimo ryšius.
BehavioredClassifier	Nėra.	Gali turėti grafinės sąsajos realizacijos elementą.
UseCase	Nėra.	Panaudojimo atvejis aprašo sistemos įvykių specifikaciją. Nurodoma, ką sistema gali atlikti ir kokie aktoriai dalyvauja dirbant su sistema.
Include	Nėra.	Apimantis ryšys, kuris apibrėžia, kad tam tikras panaudojimo atvejis yra įskaitytas į kitą panaudojimo atvejį.
Extend	Nėra.	Išplečiamojo panaudojimo atvejo ryšys, kuris apibūdina sąlygas, kurias

		įvykdžius, panaudojimo atvejis yra laikomas atliktu.
Class	Nėra.	*Class* apibūdina objektų, turinčių panašias ypatybes, rinkinį. Tai tam tikras klasifikatorius su atributais, priklausomybėmis ir operacijomis.
Property	isDerived: Boolean – nustato, ar atributo reikšmė turi kitus informacijos šaltinius. Numatyta reikšmė – *false*. isReadOnly : Boolean – jei šio atributo reikšmė – *true*, jis gali būti tik nuskaitomas, bet nekeičiamas. Numatyta reikšmė – *false*. isDerivedUnion : Boolean – nustato, ar atributas yra duomenų rinkinys. Numatyta reikšmė – *false*. /default: string – numatyta atributo reikšmė. Tai gaunami duomenys. aggregation: AggregationKind – aprašo atributo agregaciją. Numatyta reikšmė – none. / isComposite: Boolean – nustato, ar reikšmė yra sudėtine, ar ne.	*Property* atspindi klasės atributus arba atributų rinkinius.
DataType	Nėra.	Tai yra tipas, pagal kurį identifikuojama tam tikro atributo reikšmė. Dažniausiai naudojamas programuojant duomenų tipo

		atpažinimui.
AcitivityEdge	Nėra.	Abstrakti klasė, aprašanti dviejų veiklų tarpusavio ryšį.
AcitivityNode	Nėra.	Abstrakti klasė, numatanti veiklų žingsnius.
Activity	Nėra.	Parametrizuoto elgesio specifikacija, aprašanti veiklą.

Iš panaudojimo atvejų metaklasių išsaugomas panaudojimo atvejo pavadinimas, kuris generuojant grafinę vartotojo sąsaja virs lango pavadinimu. Klasės sistemoje apibūdina langų elementus, taigi reikia išsaugoti įvedamų klasių identifikacinius numerius, pavadinimus, taip pat klasių atributų identifikacinius numerius pavadinimus ir tipus. Veiklų informacija, atspindinti langų navigaciją, - tai jų pavadinimai ir tarpusavio ryšiai. Duomenys įvedami į sistemą aprašyti 19 lentelėje.

19 lentelė. Duomenys, įvedami į sistemą ir jų paaiškinimas

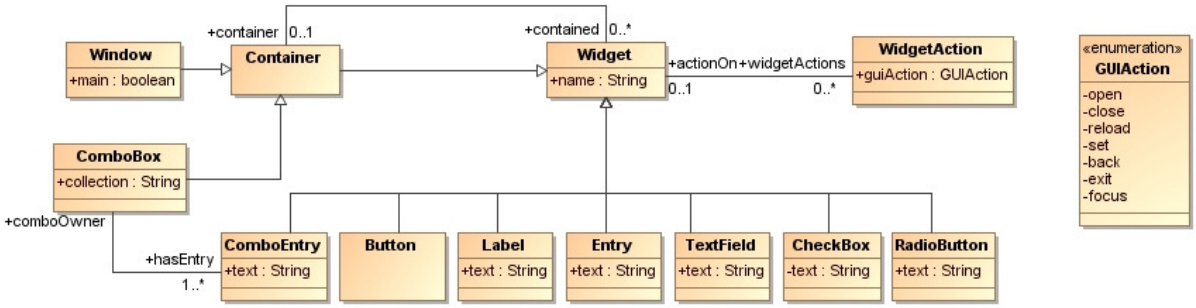
Metaklasė	**Atributas**	**Paaiškinimas**
UseCase	name: string	Panaudojimo atvejo pavadinimas
Class	name: string	Klasės pavadinimas
Property	/default: string	Atributo reikšmė
DataType	datatype: string	Atributo tipas
ActivityNode	name: string	Veiklos pavadinimas
ActivityEdge	source: string target: string	*Source* atribute saugomas esamos veiklos identifikacinis numeris; *target* – veiklos, į kurią nukreipiama esama veikla, identifikacinis numeris

4.4 Vartotojo sąsajos metamodeliai

4.4.1 Aplikacijos grafinės sąsajos metamodelis

Šiame darbe grafinės sąsajos elementams aprašyti naudojamas [4] pasiūlytas grafinės sąsajos metamodelis (49 pav.) Grafinė vartotojo sąsaja išskaidoma į atskirus grafinius elementus, kurie atvaizduojami konteineryje. Šis elementas taip pat yra priskiriamas prie grafinių elementų. Prie grafinių elementų priskiriami mygtukai, įrašai, informacijos įrašas, tekstiniai laukai, pasirinkimų elementai, sąrašo įrašas ir kt. Pats grafinės sąsajos langas ir

informacijos sąrašas nėra laikomi grafiniais elementais dėl savo savybės turėti savo elementus.



49 pav. Aplikacijos grafinės vartotojo sąsajos metamodelis

Metaklasių klasių aprašas pateiktas 20 lentelėje.

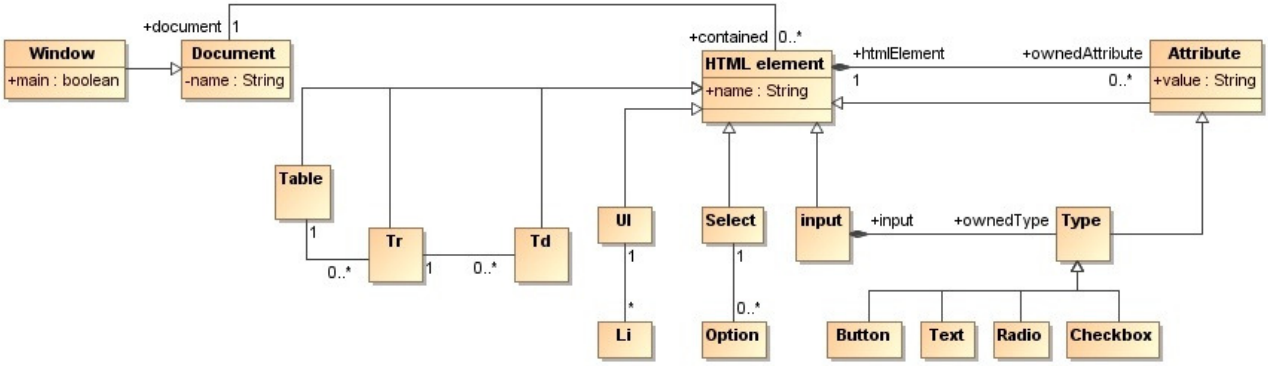
20 lentelė. Grafinės vartotojo sąsajos metamodelio klasių aprašymas

Pavadinimas	**Atributai, jų tipai ir reikšmės**	**Aprašymas**
Window	main: Boolean – numato, ar langas yra pagrindinis, ar ne.	Elementas, galintis turėti kitus grafinius elementus. Langai atvaizduojami medžio hierarchija.
Container	Nėra.	Abstraktus grafinio elemento tipas. Gali turėti kitus elementus.
Widget	name: string – grafinio elemento pavadinimas	Grafinis elementas.
WidgetAction	guiAction: GUIAction – numato, kokio tipo yra veiksmas	Aprašo trigerių inicijuotus veiksmus.
ComboBox	collection: string – aprašo *ComboBox* elementų rinkinį	Grafinis elementas, turintis *ComboEntry*. Gali būti suvokiamas kaip lentelė.
ComboEntry	text: String – įrašas su informacija	Grafinis elementas, atspindintis *ComboBox* elemento egzempliorius. Jei *ComboBox* yra interpretuojamas kaip lentelė, tuomet *ComboEntry* yra tos lentelės stulpelio įrašai.
Button	Nėra.	Grafinis elementas, inicijuojantis veiksmą.
Label	text: String – įvesta informacija	Grafinis elementas, naudojamas informacijos pateikimui. Šios

		informacijos vartotojas keisti negali.
Entry	text: String – duomenys	Informacijos įrašas apie duomenis.
TextField	text: String – tekstiniai duomenys	Tekstinis laukas.
CheckBox	text: String – informacija, aprašanti pasirinkimą	Grafinis elementas, skirtas sprendimo pasirinkimui. Galimi keli pasirinkimai.
RadioButton	text: String – informacija, aprašanti pasirinkimą	Grafinis elementas, skirtas sprendimo pasirinkimui. Galimas tik vienas pasirinkimas.

4.4.2 Internetinės grafinės vartotojo sąsajos metamodelis

Internetinės grafinės vartotojo sąsajos metamodelis pateiktas 50 paveiksle [3].



50 pav. Internetinės grafinės vartotojo sąsajos metamodelis

Metaklasių klasių aprašas pateiktas 21 lentelėje.

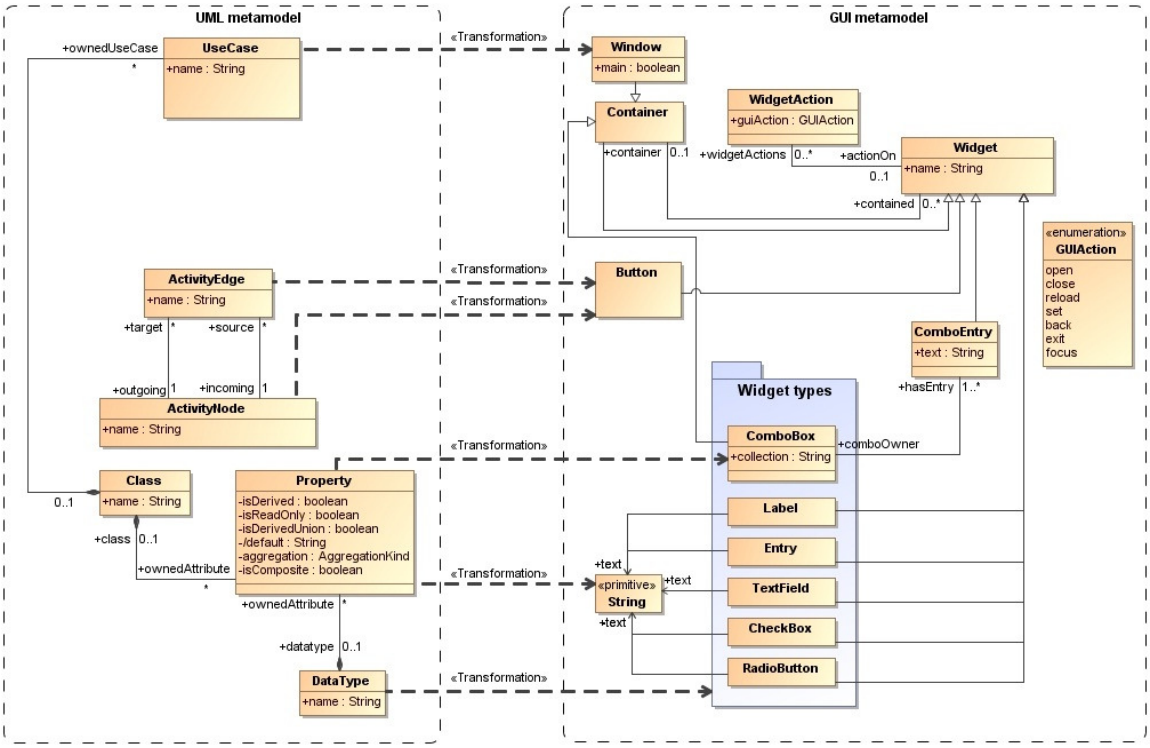
21 lentelė. Internetinės grafinės vartotojo sąsajos metamodelio klasių aprašymas

Pavadinimas	**Atributai, jų tipai ir reikšmės**	**Aprašymas**
Window	main: Boolean – numato, ar langas yra pagrindinis, ar ne.	Elementas, galintis turėti kitus grafinius elementus. Langai atvaizduojami medžio hierarchija.
Document	Nėra.	HTML failas.
HTML element	name: string – grafinio elemento pavadinimas	Grafinis elementas.
Attribute	value: string -	HTML elemento atributas.
Table	Nėra.	Grafinis elementas, skirtas atvaizduoti duomenims.
Tr	Nėra.	Lentelės stulpelis.
Td	Nėra.	Lentelės eilutė.

Select	Nėra.	Grafinis elementas, kuriame atvaizduojamas sąrašas (*ComboBox* atitikmuo).
Option	Nėra.	Grafinis elementas, atspindintis *Select* elemento egzempliorius.
Input	Nėra.	HTML elementas.
Type	Nėra.	*HTML* elemento atributas.
Button	Nėra.	*Input* elemento tipas, inicijuojantis veiksmą .
Text	Nėra.	*Input* elemento tipas, kuriame rašomas tekstas.
Radio	Nėra.	*Input* elemento tipas, skirtas sprendimo pasirinkimui. Galimas tik vienas pasirinkimas.
Checkbox	Nėra.	*Input* elemento tipas, skirtas sprendimo pasirinkimui. Galimi keli pasirinkimai.

4.5 UML metaklasių transformacija į aplikacijos grafinės vartotojo sąsajos metaklases

Šiame darbe siūlomo transformavimo iš UML diagramų į aplikacijos vartotojo grafinės sąsajos prototipą algoritmo principai pateikiami 51 paveiksle. Transformacija susideda iš keleto žingsnių. Pirmiausiai transformuojami panaudojimo atvejai. Kiekviena panaudojimo atvejo metaklasė *UseCase* atitinka grafinės sąsajos metaklasę *Window*. Lango pavadinimas (metaklasės *Window* atributas *name*) gaunamas iš metaklasės *UseCase* atributo *name*.



51 pav. UML metamodelio klasių transformacija į aplikacijos grafinės sąsajos metamodelio elementus

Sekantis žingsnis – transformuoti klasių diagramos elementus – klases (metaklasė *Class*) ir jų atributus (metaklasė *Property*). Klasių (*Class*) atributai (*Property*) transformuojami į atitinkamus lango elementus, kuriuos apibendrina metaklasė *Widget*, pagal jų tipą. Šiuos lango elementus atitinka metaklasės *Label*, *Entry*, *TextField*, *CheckBox*, *RadioButton*, *ComboBox*. Atributo vardas (metaklasės *Widget* atributas *name*) gaunamas iš *Property* metaklasės atributo *default*, o tipas nustatomas iš *DataType* metaklasės. Tokiu būdu patikrinus visas klases, tos klasės ir jų atributai yra transformuojami į grafinės vartotojo sąsajos elementų metaklases. Ryšys tarp klasės ir panaudojimo atvejo nustato, kurie atributai kuriam langui priklauso. Transformuojamas *Class* – *UseCase* ryšys į ryšį tarp *Window* – *Widget* metaklasių.

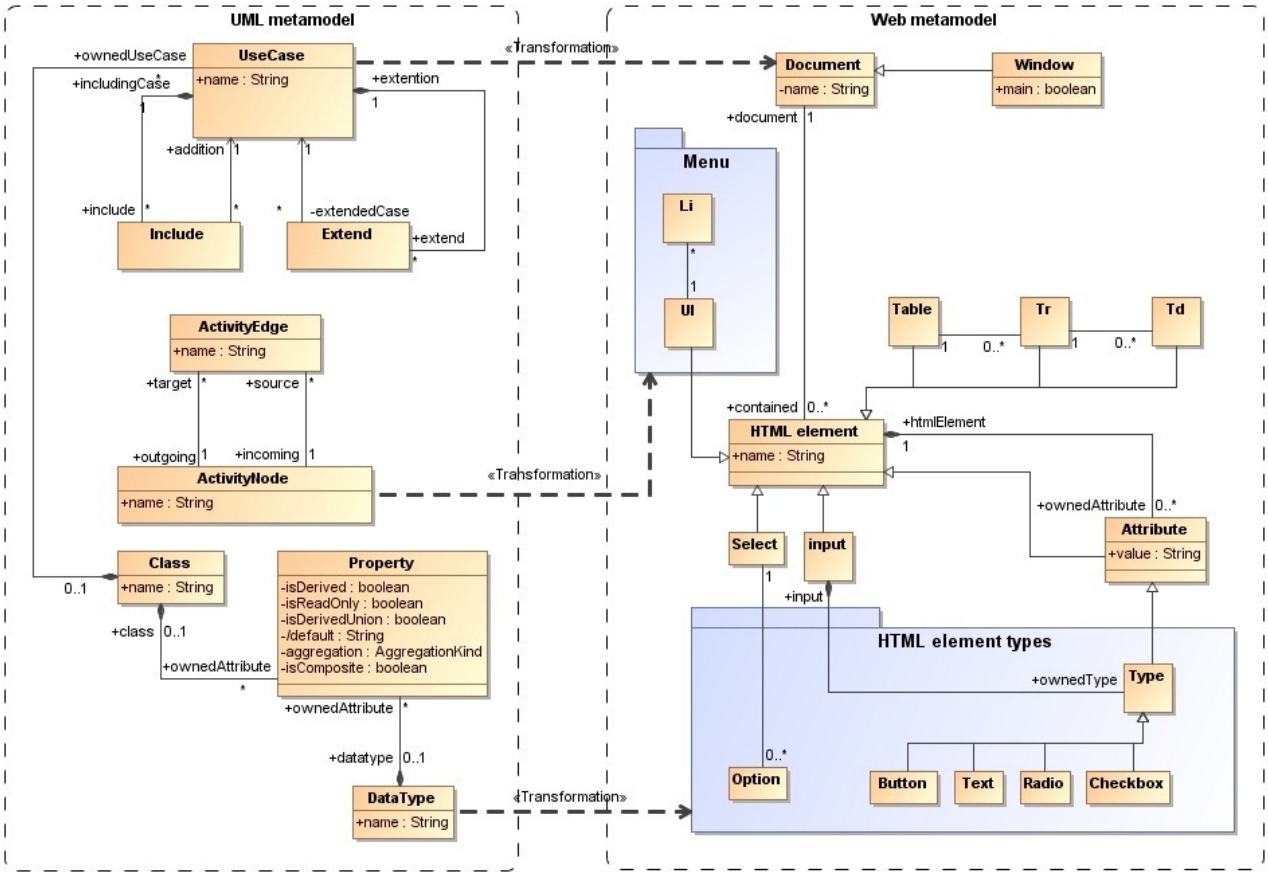
Veiklos diagramų elementai – veiklos (metaklasė *ActivityNode*) transformuojamos į grafinės sąsajos elementus mygtukus (metaklasė *Button*). Mygtuko pavadinimas (klasės *Button* atributas *name*) gaunamas iš *AcivityNode* atributo *name*. Navigavimo planas, kuris nurodo kurį sekantį langą atidaryti, gaunamas iš veiklos ryšių (metaklasė *ActivityEdge*). Ryšiai gaunami iš klasės *ActivityEdge* atributų *source* ir *target*. Kiekviena veikla (klasė *AcitivityNode*) yra susieta su panaudojimo atveju.

Atlikus diagramų metaklasių transformavimą į langų metaklases, pastarosios gali būti atvaizduojamos į vartotojo sąsajos elementus, taip sudarant grafinį vartotojo sąsajos prototipo vaizdą.

4.6 UML metaklasių transformacija į internetinės grafinės vartotojo sąsajos metaklases

UML metaklasių transformacija į internetinės grafinės sąsajos metaklases vyksta panašiai kaip 4.5 skyriuje aprašyta pasiūlyta transformacija. Skiriasi tik elementai, į kuriuos transformuojamos UML metaklasės.

Transformacija taip pat pradedama nuo panaudojimo atvejų. Kiekviena panaudojimo atvejo metaklasė *UseCase* atitinka grafinės sąsajos metaklasę *Document*. Lango pavadinimas (metaklasės *Document* atributas *name*) gaunamas iš metaklasės *UseCase* atributo *name*.



52 pav. UML metamodelio klasių transformacija į internetinės grafinės sąsajos metamodelio elementus

Toliau transformuojama klasių diagrama. Metaklasė *DataType* transformuojami į HTML elementus, o *ActivityNode* – į *Ul* ir *Li* HTML elementus, iš kurių sudaromas meniu.

Atlikus transformaciją, internetinės grafinės sąsajos elementai atvaizduojami vartotojo sąsajoje sudarant puslapius, mygtukus ir grafinius elementus.

4.7 Algoritmas grafinės vartotojo sąsajos generavimui

4.7.1 Algoritmas aplikacijos grafinės vartotojo sąsajos generavimui

Algoritmas grafinei vartotojo sąsajai generuoti pateiktas 53 paveiksle. Tolesniuose skyriuose jis bus aptariamas detaliau.

53 pav. Algoritmas grafinei vartotojo sąsajai generuoti

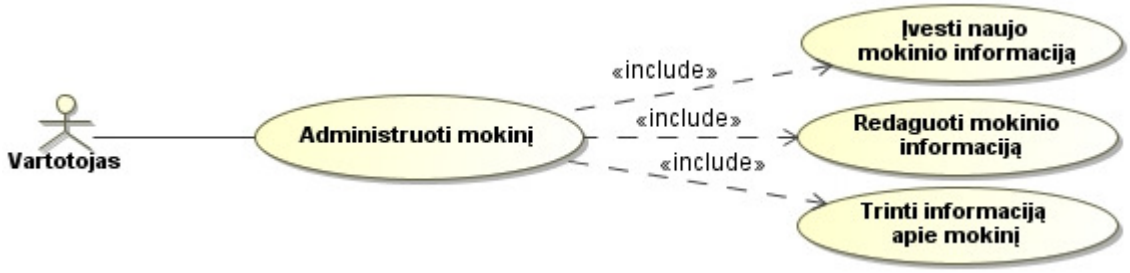
4.7.1.1 Diagramų masyvų išsaugojimas

Šiame skyriuje detaliai aprašomas algoritmas, kartu parodant kaip jis vykdomas su pavyzdinės grafinės sąsajos kūrimu.

Tarkime, kad reikia sukurti mokyklos informacijos sistemos grafinę sąsają. Mokyklos IS skirta administruoti informaciją apie mokinius, būrelius, pamokas bei gaunamus pažymius. Pagrindinės tokios sistemos funkcijos: informacijos apie mokinius, lankomas pamokas bei būrelius ir gaunamus pažymius suvedimas bei administravimas. Šiame skyriuje pateikti tik diagramų fragmentai, o 7.1 skyriuje – pilnos diagramos su platesniais paaiškinimais. Remiantis pateiktais diagramų fragmentais bus sudaroma projektinė sistemos grafinė vartotojo sąsaja naudojantis įrankiu *MagicDraw UML*.

Mokyklos informacijos sistemai sudaromos panaudojimo atvejų, klasių bei veiklos diagramos. Tai bus pradiniai duomenys, iš kurių bus kuriama grafinė vartotojo sąsaja.

Panaudojimo atvejų diagramos fragmentas pavaizduotas 54 paveiksle. Panaudojimo atvejai: administruoti mokinį, įvesti naujo mokinio informaciją, redaguoti mokinio informaciją bei trinti informaciją apie mokinį.

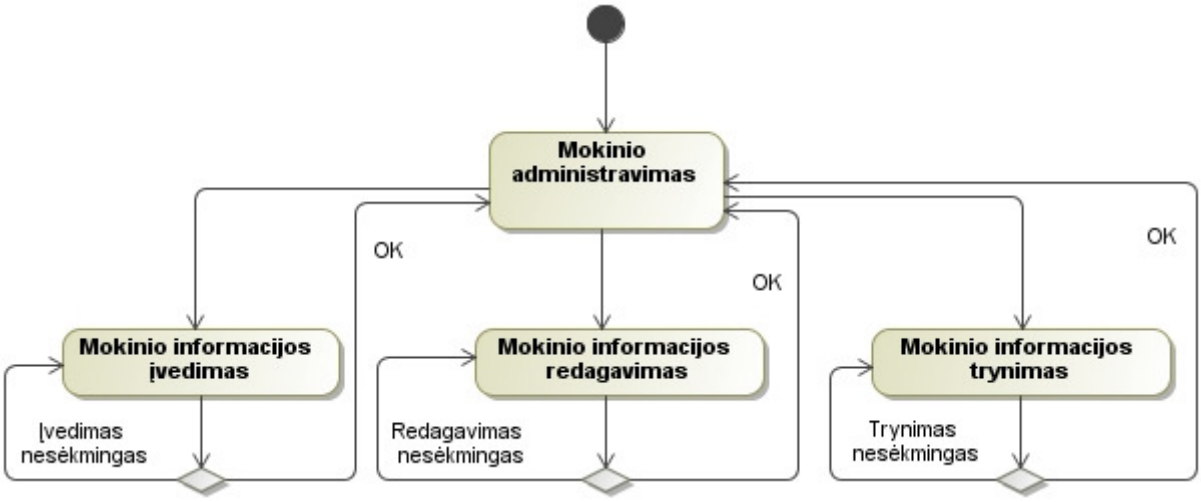


54 pav. Panaudojimo atvejų diagramos fragmentas pavyzdinei Mokyklos IS

Mokyklos informacijos sistemai taip pat sudaroma klasių diagrama. Jos fragmentas pateiktas 55 paveiksle. Kadangi dirbama tik su mokinio informacija, joje yra tik viena klasė „Mokinys“.

55 pav. Klasių diagramos fragmentas pavyzdinei Mokyklos IS

Mokyklos informacinės sistemos veiklos diagrama pavaizduota 56 paveiksle. Vartotojas darbą pradeda pagrindiniame mokinio administravimo lange, o iš jo gali patekti į kitus langus, t.y. informacijos įvedimo bei redagavimo. Mokinio informacijos trynimo funkcija numatoma atlikti mokinio administravimo lange esančiame sąraše.



56 pav. Veiklos diagramos fragmentas pavyzdinei Mokyklos IS

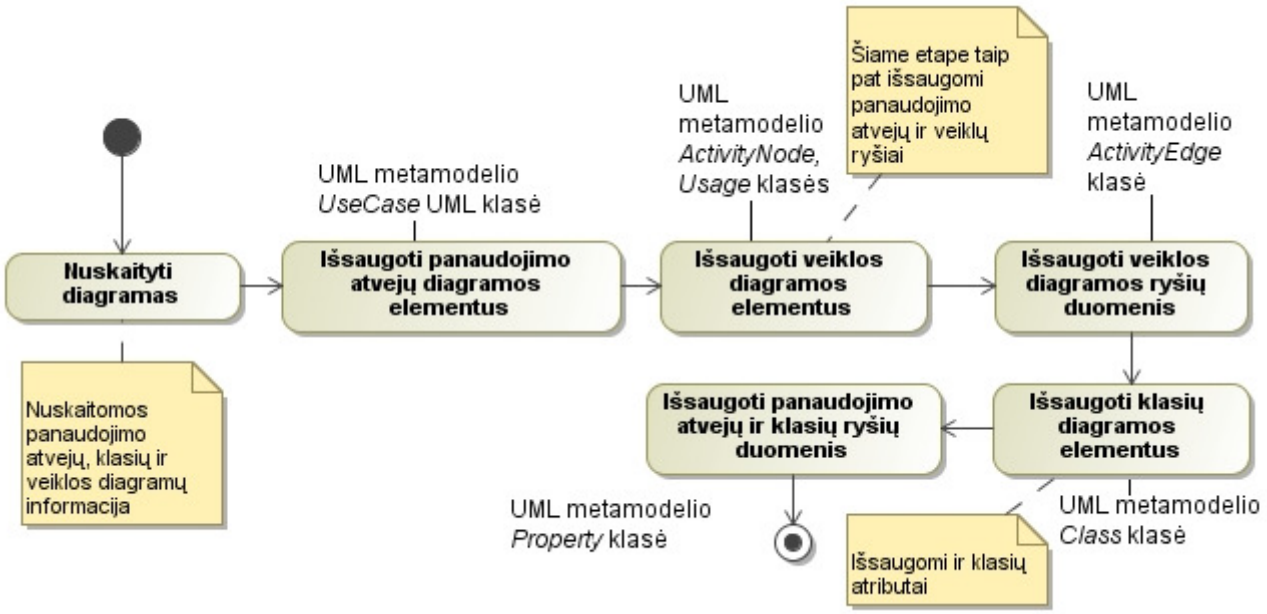
Toliau nagrinėjant pavyzdžio fragmentą, bus aprašomas algoritmas grafinei vartotojo sąsajai generuoti.

Pirmiausiai išsaugojami panaudojimo atvejai – „Administruoti mokinį“, „Įvesti naujo mokinio informaciją“, „Redaguoti mokinio informaciją“, „Trinti mokinio informaciją“.

Toliau išsaugomos veiklos – „Mokinio administravimas“, „Mokinio informacijos įvedimas“, „Mokinio informacijos redagavimas“ ir „Mokinio informacijos trynimas“. Kadangi kiekviena veikla siejasi su panaudojimo atveju yra atrenkamas panaudojimo atvejis, su kuriuo susieta veikla ir tik tada veikla yra išsaugoma. Pavyzdžiui, panaudojimo atvejis „Įvesti naujo mokinio duomenis“ susijęs su veikla „Mokinio informacijos įvedimas“. Išsaugomos ne tik pačios veiklos, bet ir jų ryšiai. Veiklos ryšiai paimami iš elemento „edge“, kuris aprašytas ankstesniuose skyriuose.

Išsaugojus šiuos duomenis, nuskaitoma klasių diagramos informacija. Nagrinėjamame pavyzdyje yra tik viena klasė „Mokinys“, jos duomenys ir išsaugomi. Klasės taip pat turi būti susietos su panaudojimo atvejais, todėl įkeliant klases saugojami ir ryšiai tarp panaudojimo

atvejų. Įrašius klasių informaciją, išsaugomi ir joms priskirti atributai. Algoritmą vaizduojanti veiklos diagrama pateikta 57 paveiksle.



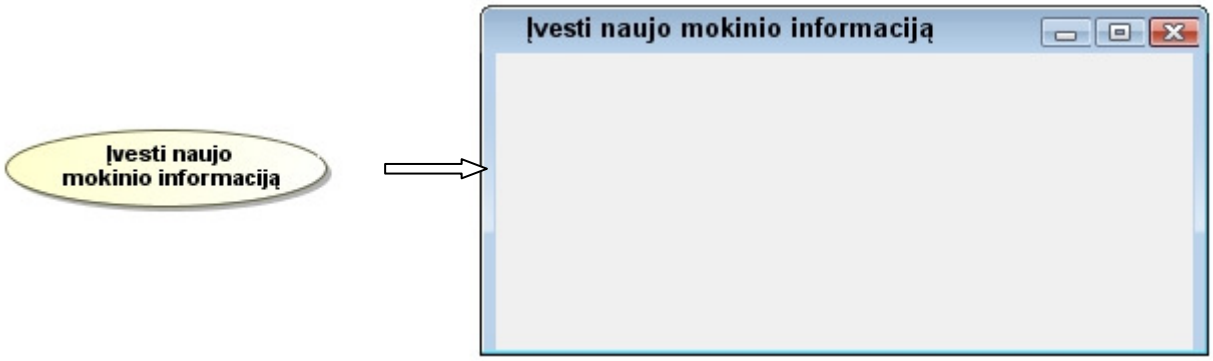
57 pav. Diagramų išsaugojimas

4.7.1.2 Diagramų metaklasių transformavimas į langų klases

Diagramų metaklasių transformavimas į langų klases pradedamas panaudojimo atvejų nuskaitymu. Kiekvienam panaudojimo atvejui sukuriamas grafinės sąsajos langas su priskirtais duomenimis, t.y. lango pavadinimu.

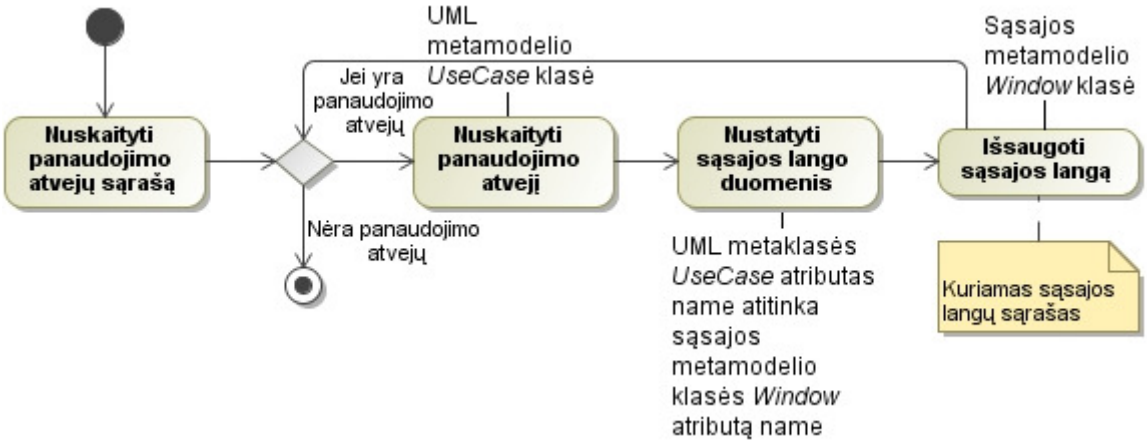
Pagal pavyzdį kuriamoje grafinėje sąsajoje turi būti sukurti trys langai – „Mokinių administravimo“, „Mokinio informacijos įvedimo“ ir „Mokinio informacijos redagavimo“. Jų pavadinimai atitinka panaudojimo atvejų pavadinimams. Plačiau bus panagrinėtas dviejų sąsajos langų sudarymas – „Mokinių administravimo“ ir „Mokinio informacijos įvedimo“. Šie langai parinkti todėl, kad juose turi būti atvaizduojami skirtingi grafiniai elementai.

Pavyzdžiui, panaudojimo atvejui „Įvesti naujo mokinio informaciją“ bus sukuriamas langas su tokiu pat pavadinimu. Tai pavaizduota 58 paveiksle.



58 pav. Sukurtas naujas sąsajos langas pagal panaudojimo atvejo duomenis

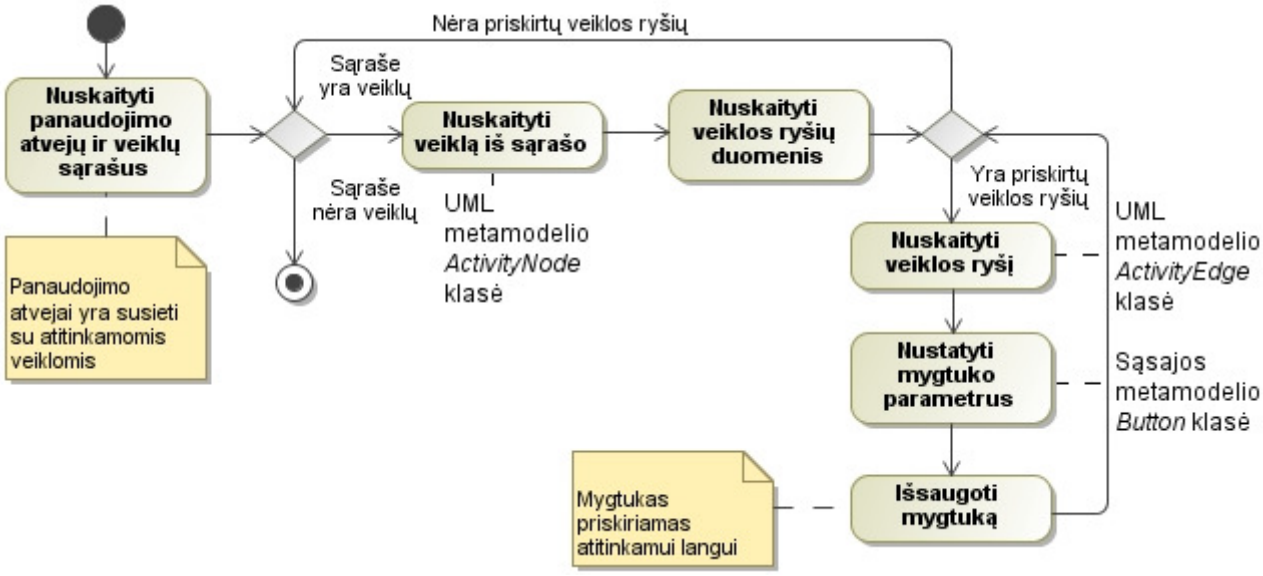
Kitiems panaudojimo atvejams tokiu pat būdu sukuriami langai.



59 pav. Langų sukūrimas pagal panaudojimo atvejus

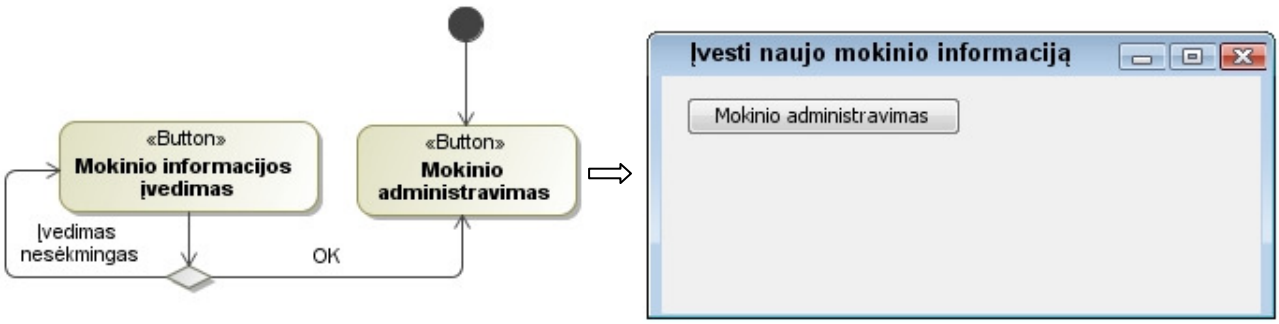
Sukūrus langus, kiekvienam iš jų priskiriami langų mygtukai. Kiekvienas lango mygtukas atitinka veiklą iš veiklos diagramos.

60 paveiksle pateikiamas lango mygtuko išsaugojimo algoritmas.

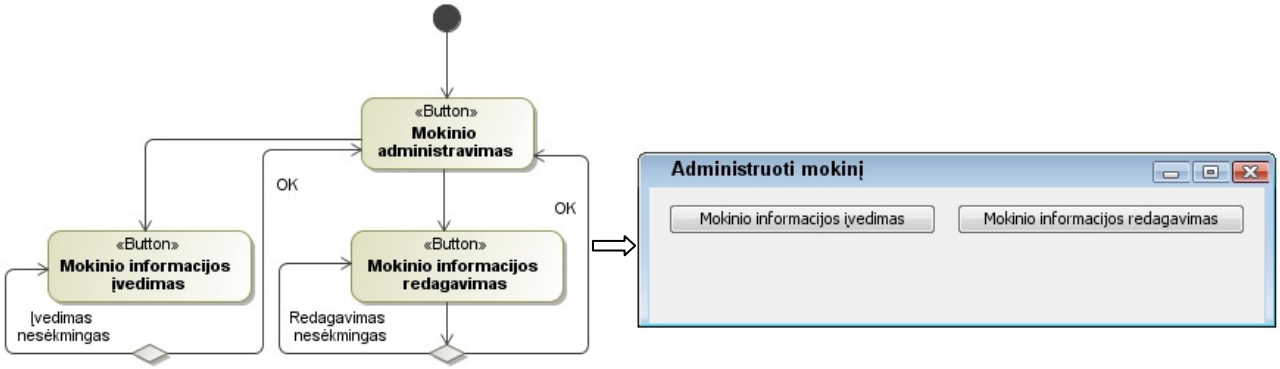


60 pav. Lango mygtukų išsaugojimo algoritmas

Pavyzdžiui, mokinio įvedimo lange bus mygtukas „Mokinio administravimas“ (61 pav.), kurį paspaudus bus grįžtama į administravimo langą, o sąsajos lange „Administruoti mokinį“ bus du mygtukai: „Įvesti naujo mokinio informaciją“ ir „Redaguoti mokinio informaciją“ (62 pav.).

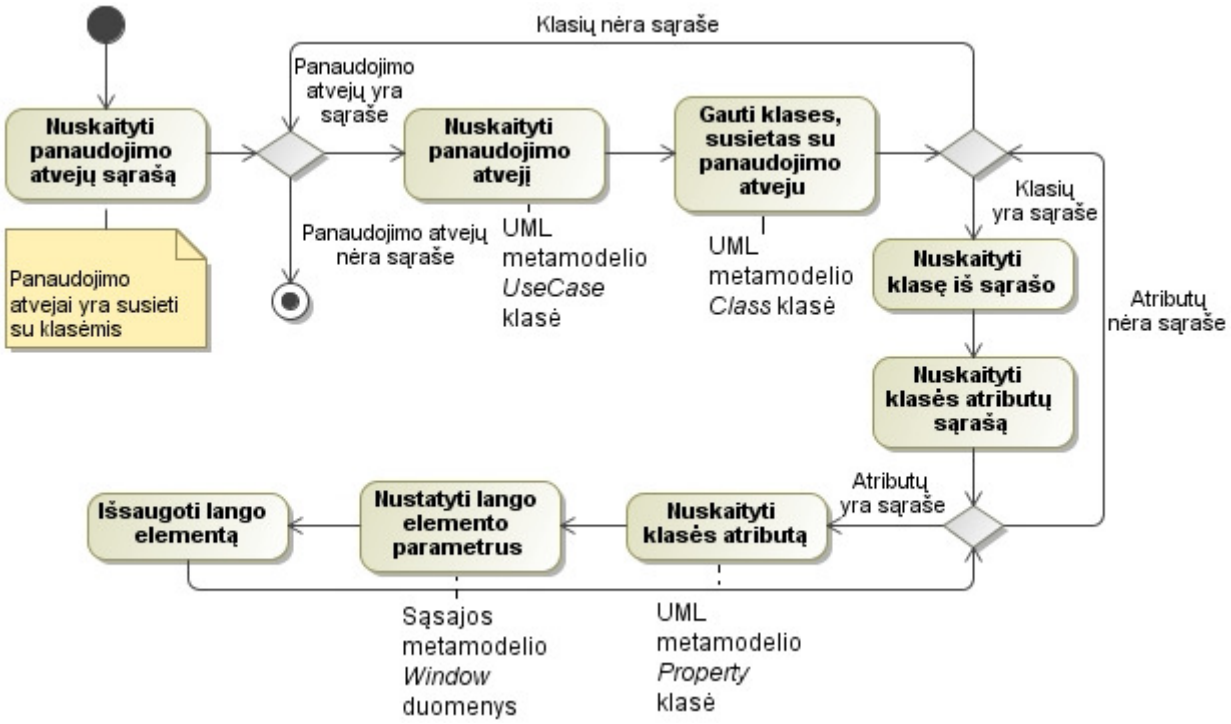


61 pav. Navigacijos kūrimas mokinio įvedimo lange remiantis veiklos diagramos fragmentu



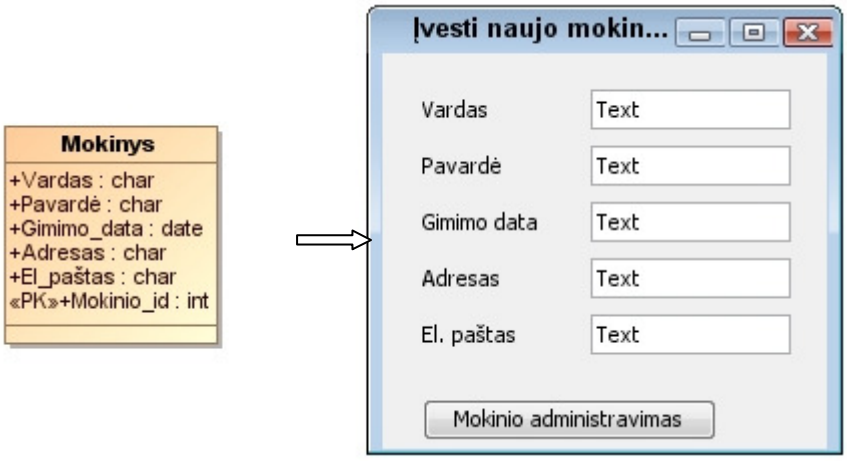
62 pav. Navigacijos kūrimas mokinių administravimo lange remiantis veiklos diagramos fragmentu

Toliau išsaugomi lango elementai, kurių kiekvienas atitinka klasės atributą. Kadangi panaudojimo atvejis atitinka langą, galima nustatyti, kurie atributai priklauso kuriam panaudojimo atvejui ir taip juos priskirti langui. Panaudojimo atvejai siejasi su klasėmis, o kiekviena klasė siejasi su jai priskirtais atributais. Lango elementų išsaugojimo algoritmas pateiktas 63 paveiksle.



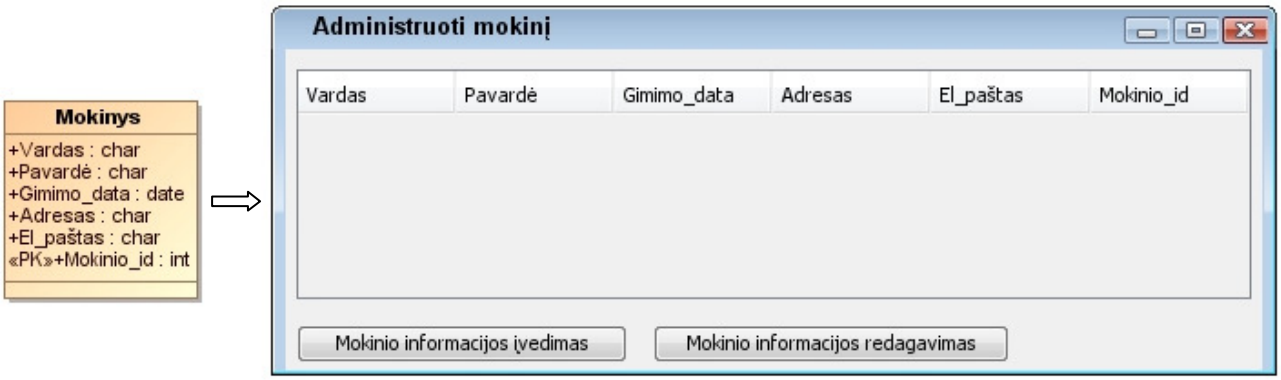
63 pav. Lango elementų išsaugojimo algoritmas

Pavyzdžiui, įvykdžius mygtukų ir lango elementų priskyrimo langams algoritmus, sukurtame „Mokinio informacijos įvedimo“ lange turėtų būti sukuriami šie elementai – Vardas, Pavardė, Gimimo_data, Adresas, El_paštas, Mokinio_id (64 pav.).



64 pav. „Mokinio informacijos įvedimo“ lango elementų kūrimas pagal klasių diagramą

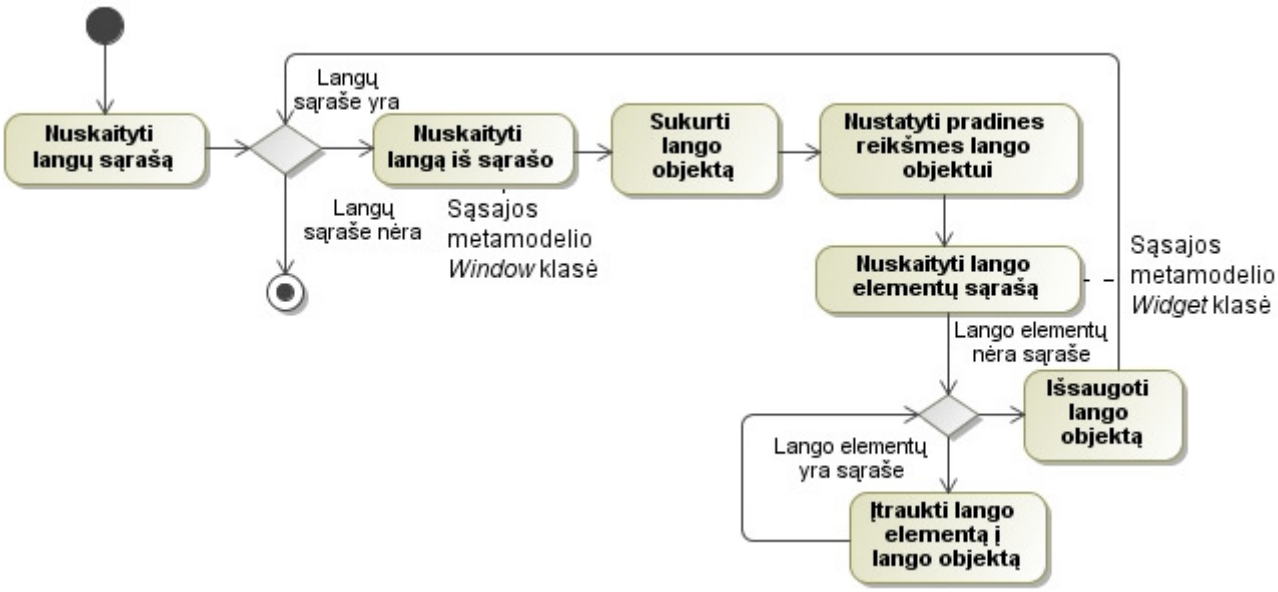
Mokinių administravimo lange turėtų būti atvaizduojamas sąrašas, kadangi duomenys nėra nei įvedami, nei redaguojami. Tie patys klasės „Mokinys“ atributai atvaizduojami stulpeliais (65 pav.).



65 pav. Mokinių administravimo lango elementų įkėlimas

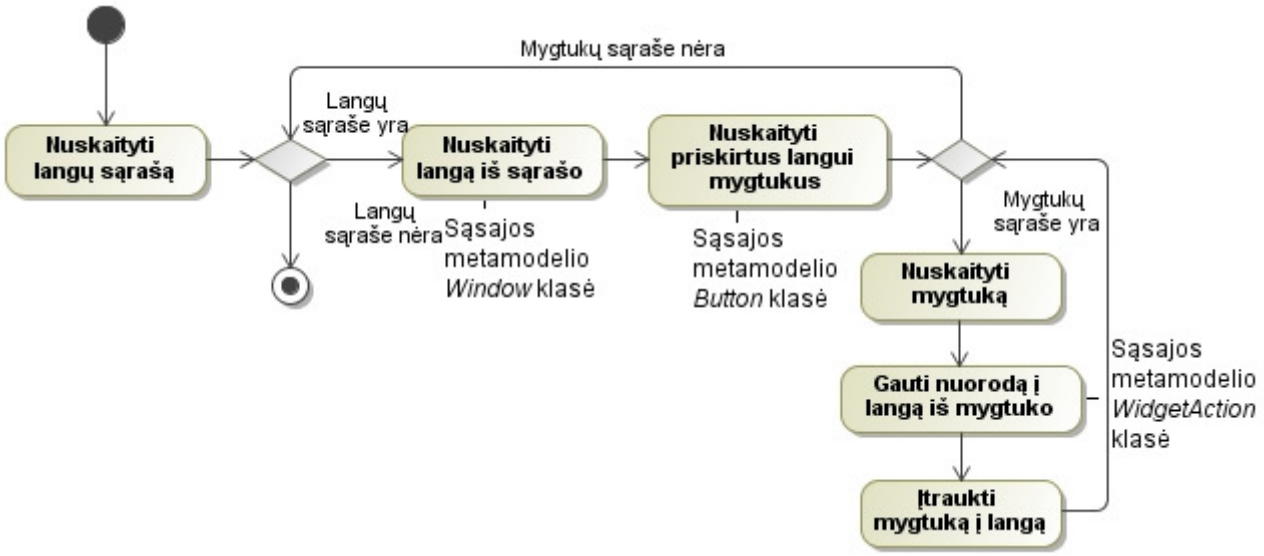
4.7.1.3 Lango klasių transformavimas į vartotojo sąsajos elementus

Pirmiausiai nuskaitomas langų sąrašas ir sukuriami langų dialogai, į kuriuos vėliau bus sudėti visi langams priskirti elementai. Verta pažymėti, kad sukurtame objekte, įtraukiant elementą, yra tikrinamas jo tipas ir pagal tai nusprendžiama, kokį sąsajos elementą įtraukti. Lango klasių transformavimo į vartotojo sąsajos elementus algoritmas pateiktas 66 paveiksle.



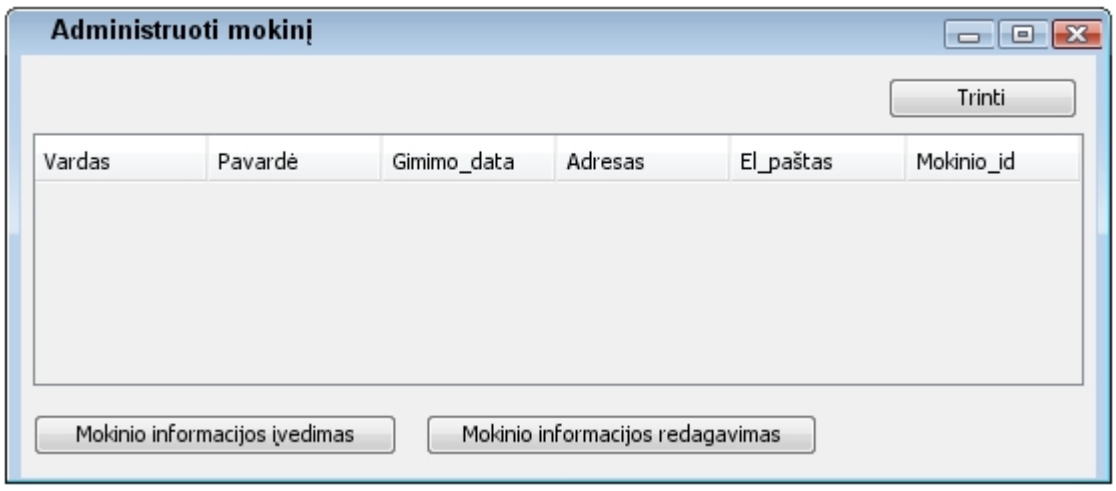
66 pav. Lango klasių transformavimas į vartotojo sąsajos elementus

Įtraukus lango elementus į objektus, į langą sukeliami jau sukurti navigavimo mygtukai. Sukuriami mygtukų objektai ir nustatoma, kuriuos langus mygtukai atidarys juos paspaudus. Langų navigacijos algoritmas pateikiamas 67 paveiksle.



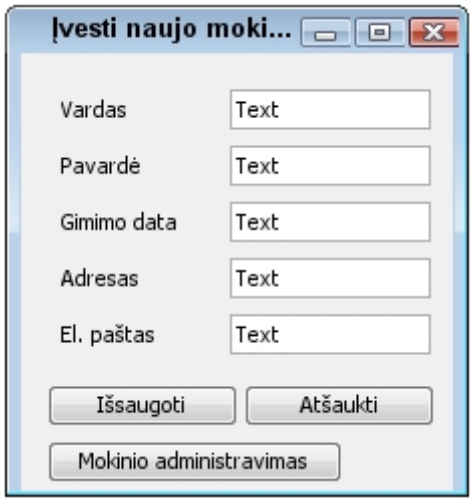
67 pav. Langų navigacijos sukūrimo algoritmas

Toliau pateikti sukurti grafinės sąsajos langai – rezultatas, kuris gaunamas naudojant panaudojimo atvejų, klasių bei veiklos diagramas. Pagal aprašytą algoritmą pavyzdinės Mokyklos informacijos sistemos pagrindinis „Mokinių administravimo“ langas turėtų atrodyti taip, kaip pavaizduota 68 paveiksle. Lange įkeliamas mokinių sąrašas su klasėje „Mokinys“ esančiais atributais, atvaizduojamais stulpeliais. Mokinių trynimo funkcija bus atliekama mygtuko „Trinti“ pagalba. Kadangi „Mokinio administravimo“ veikla susieta su veiklomis „Mokinio informacijos įvedimas“ ir „Mokinio informacijos redagavimas“, lange įkeliami mygtukai su šių veiklų pavadinimais.



68 pav. Mokinio administravimo projektinis langas

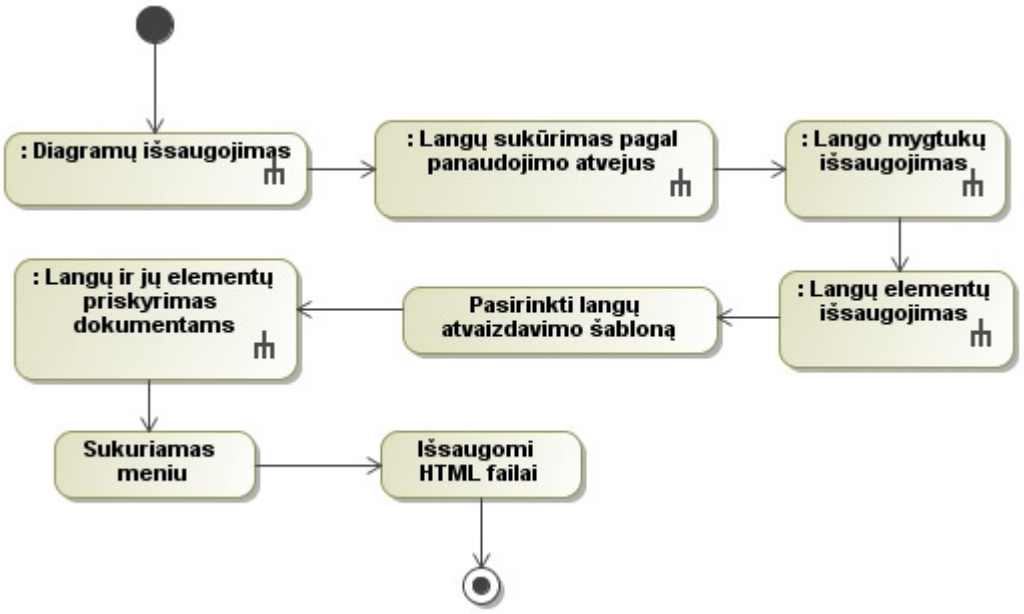
„Mokinio informacijos įvedimo“ langas turėtų atrodyti taip, kaip pavaizduota 69 paveiksle. Jame įkeliami elementai pagal klasėje „Klasė“ sukurtus atributus. Taip pat sukuriamas mygtukas „Mokinio administravimui“ dėl navigacijos tarp langų. Kadangi formoje įvedamą informaciją turi būti galima išsaugoti arba, jos neįvedus, atšaukti šį žingsnį, sukuriami du papildomi mygtukai „Išsaugoti“ ir „Atšaukti“.



69 pav. Naujo mokinio informacijos įvedimo projektinis langas

4.7.2 Algoritmas internetinės grafinės vartotojo sąsajos generavimui

Tarkime, norime sugeneruoti internetinę vartotojo sąsają. Jai sugeneruoti vykdomi tie patys etapai, kaip ir aplikacijos sąsajos generavimui – pirmiausia diagramų informacija išsaugoma, diagramų klasės transformuojamos į lango klases, o šios – į sąsajos elementų klases, kurių duomenys galiausiai atvaizduojami sukurtuose sąsajos languose. Internetinės grafinės vartotojo sąsajos generavimo algoritmas pateiktas 70 paveiksle.

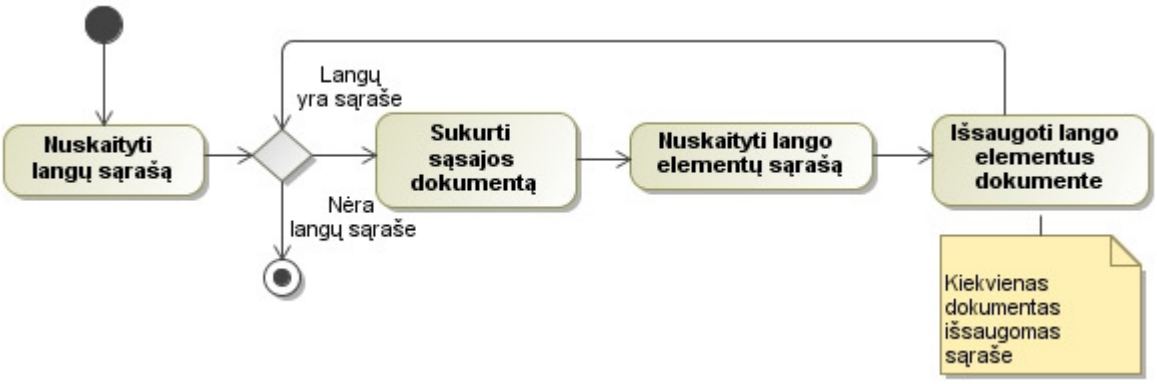


70 pav. Internetinės grafinės vartotojo sąsajos generavimo algoritmas

Kaip matome iš paveikslo, etapai, aprašyti 4.7.1.1 - 4.7.1.2 skyriuose sutampa, tačiau skiriasi lango klasių transformavimas į vartotojo sąsajos elementus algoritmas.

Internetinės grafinės sąsajos generavimui naudojami paruošti šablonai, t.y. paruošti langų maketai su nurodyta vieta meniu, mygtukų ir kt. elementų atvaizdavimui.

Algoritme sukuriami dokumentai, kurie bus reikalingi sąsajos atvaizdavimui. Kiekvienam langui yra sukuriami atskiri dokumentai, o šiems priskiriami elementai.



71 pav. Langų ir jų elementų priskyrimas dokumentams

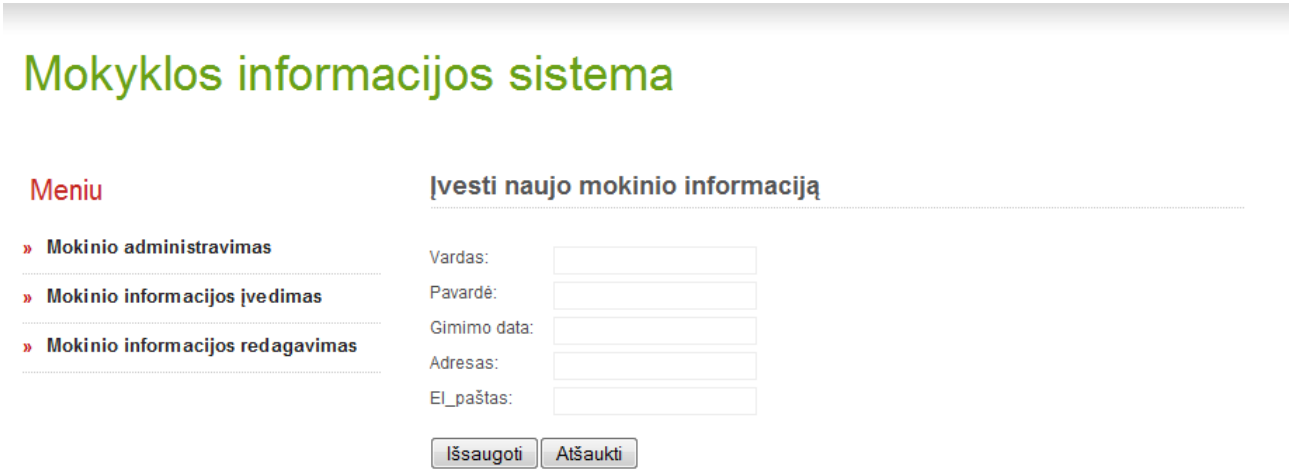
Turint dokumentą, jam sukuriamas meniu. Meniu kuriamas pagal tai, kokie yra išsaugoti langų pavadinimai. Jie atvaizduojami šablone numatytoje vietoje. Kiekviename iš sukurtų dokumentų meniu atvaizduojamas vienodai.

Algoritmo vykdymo eigoje kiekvienas dokumentas virsta realiu failu, kuriame atvaizduojamas parinktas šablono stilius, meniu, grafiniai elementai ir kt.

Nagrinėjamo Mokyklos informacijos sistemos fragmento „Mokinių administravimo“ langas internetinėje sąsajoje turėtų atrodyti taip, kaip pavaizduota 72 paveiksle.

72 pav. Mokinio administravimo projektinis langas internete

Šiame lange atvaizduojami tokie patys elementai kaip ir aplikacijos grafinėje vartotojo sąsajoje, tačiau vietoj mygtukų navigacijai tarp langų sukuriamas meniu. Jis matomas visuose sąsajos languose. Naujo mokinio informacijos įvedimo langas pavaizduotas 73 paveiksle.

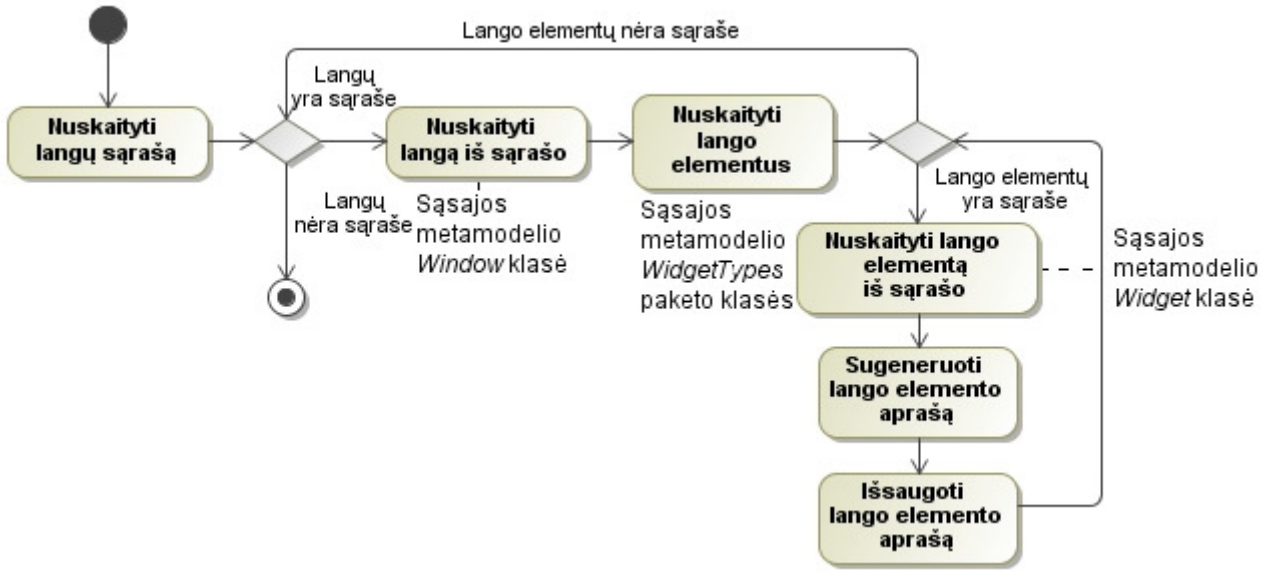


73 pav. Naujo mokinio informacijos įvedimo projektinis langas

4.8 Grafinės vartotojo sąsajos atvaizdavimo programiniu kodu algoritmas

Sugeneravus programinį grafinės sąsajos langą tuo pačiu metu yra sugeneruojamas jo programinis kodas. Kodo generavimui naudojami transformuoti elementai. Programinis kodas išsaugomas teksto kintamajame. 74 paveiksle pavaizduotas kodo generavimo algoritmo, kuris tinka daugeliui programavimo kalbų.

Iš pradžių yra sukuriami naudojamų kintamųjų aprašai. Kintamuosius sudaro teksto laukai, mygtukai, pasirinkimai ir t.t.



74 pav. Grafinės vartotojo sąsajos atvaizdavimo programiniu kodu algoritmas

Visas sugeneruotas programinis kodas išsaugomas failuose. Kodui generuoti naudojamas bendras kodo šablonas su nurodytomis vietomis, kur bus išsaugoti kintamieji, o kur bus išsaugoti sukurti elementai.

Toliau pateiktas sudarytas kodas „Mokinių administravimo“ langui pagal 53 paveiksle aprašytą algoritmą. Kodas pateikiamas tiek *Java*, tiek *HTML* kalba.

Java programinis kodas „Mokinių administravimo“ langui:

```
import info.clearthought.layout.TableLayout;
import javax.swing.*;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.util.Iterator;
public class NewJDialog extends javax.swing.JDialog {
              private int _elementsCount = 7;
              private int _rowHeight = 20;
              private int _columnWidth = 80;
              private JMenuBar jMainMenu;
              private JPanel jListPanel;
              private JButton jButton2;
              private JButton jButton3;
              private int _position;
              private double[] _rows;
              private double[] _columns;
              private JMenu jFileMenu;
              private JMenu jExitMenu;
              private JMenuItem jExit;
              private ButtonGroup buttonGroup;
              private JPanel jButtonsPanel;
public static void main(String[] args) {
              JFrame frame = new JFrame();
              NewJDialog inst = new NewJDialog(frame);
              inst.setVisible(true);
              }
```

```
public NewJDialog(JFrame frame) {
		super(frame);
		initGUI();
}
private void initGUI() {
		try {
		double[] formColumns = new double[]{ 7.0, TableLayout.FILL, TableLayout.FILL,
22.0, 7.0};
		TableLayout thisLayout = new TableLayout(new double[][] {
		{ 7.0, TableLayout.FILL, TableLayout.FILL, TableLayout.FILL, TableLayout.FILL,
7.0 },formColumns })
				thisLayout.setHGap(5);
				thisLayout.setVGap(5);
		getContentPane().setLayout(thisLayout);
				initFormSize();
		//Menu
		{
		jMainMenu = new JMenuBar();
		setJMenuBar(jMainMenu);
		jFileMenu = new JMenu("Failas");
		jFileMenu.getAccessibleContext().setAccessibleDescription("Failas");
		jMainMenu.add(jFileMenu);
		jExitMenu = new JMenu("Atsijungti");
		jExitMenu.getAccessibleContext().setAccessibleDescription("Atsijungti");
		jMainMenu.add(jExitMenu);
jExit = new JMenuItem("Baigti darbą");
		jExit.getAccessibleContext().setAccessibleDescription("Baigti darbą");
				jExit.addActionListener(new GenCloseDialog(this));
		jExitMenu.add(jExit);
}
{
		buttonGroup = new ButtonGroup();
}
		//All rows between label and button are filled with window elements
		}
		{
		jButtonsPanel = new JPanel();
		jButtonsPanel.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 0));
		}
		//generate list panel
		{
		jListPanel = new JPanel();
		TableLayout jPanel2Layout = new TableLayout(new double[][] {_columns,
		{ 22.0, 30.0, 40.0, 40.0, 5.0 } });
		jPanel2Layout.setHGap(5);
		jPanel2Layout.setVGap(5);
		jListPanel.setLayout(jPanel2Layout);
		jListPanel.setBorder(BorderFactory.createTitledBorder("Sąrašas"));
		{
		jButton3 = new JButton();
		jListPanel.add(jButton3, "1, 0");
		jButton3.setText("Redaguoti");
		}
```

```
            {
            jButton2 = new JButton();
            jListPanel.add(jButton2, "2, 0");
            jButton2.setText("Trinti");
            }
}
pack();
setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
DefaultTableModel model = new DefaultTableModel();
String[] columnNames = new String[]{
"el_pastas","lytis","adresas","gimimo_data","pavarde","vardas","mokinio_id",};
int dataSize = 10;
int index = 0;
Object[][] data = new Object[_elementsCount][dataSize];

for (int i = 0; i < _elementsCount; i++) {
for (int j = i; j < dataSize; j++) {
data[i][j] = "";
}
}
model.setDataVector(data, columnNames);
JTable table = new JTable(model);
table.setFillsViewportHeight(true);
JScrollPane scrollTable = new JScrollPane( table );
jListPanel.add(scrollTable,"1, 1, "+(_columns.length-1)+", 3");
            } catch (Exception e) {
                        e.printStackTrace();
            }
}
private void initFormSize() {
            int add = 2;
            int columnAdd = 4;
            int columnsCount = _elementsCount < columnAdd ? columnAdd : _elementsCount;
            _rows = new double[_elementsCount+add];
            _columns = new double[columnsCount];
            for (int i = 0; i < _elementsCount+add; i++) {
                        _rows[i] = _rowHeight;
            }
            _columns[0] = 10;
            for (int i = 1; i < columnsCount; i++) {
                        _columns[i] = _columnWidth;
            }
            _position = 0;
            }
      }
```

HTML programinis kodas „Mokinių administravimo“ langui:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
```

```
<title>Administruoti mokinį</title>
<meta name="keywords" content="" />
<meta name="Hanging" content="" />
<link href="styles/default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<div id="wrapper">
    <div id="header">
      <div id="logo">
        <h1>Mokyklos IS</h1>
      </div>
  </div>
  <div id="page">
   <div id="sidebar1" class="sidebar">
      <ul>
        <li>
          <h2>Meniu</h2>
          <ul>
              <li><a href="Administruoti mokinį.html">Administruoti mokinį</a></li>
              <li><a href="Redaguoti mokinio  informaciją.html">Redaguoti mokinio
informaciją</a></li>
              <li><a href="Įvesti naujo  mokinio informaciją.html">Įvesti naujo  mokinio
informaciją</a></li>
              </ul>
      </ul>
    </div>
    <div id="content">
      <div class="post">
        <h1 class="title">Administruoti mokinį</h1>
              <div>
                          <input type="button" name="buton" value="Redaguoti" onclick="" />
                          <input type="button" name="buton" value="Trinti" onclick="" />
              </div>
              <table cellpadding="0" cellspacing="0" border="0" id="tableList">
                          <tr>
                                      <th>Adresas</th>
                                      <th>El_paštas</th>
                                      <th>Gimimo_data</th>
                                      <th>Mokinio_id</th>
                                      <th>Pavardė</th>
                                      <th>Vardas</th>
                          </tr>
              </table>
      </div>
  </div>
<div style="clear: both;"> </div>
</div>
</div>
</div>
<div id="footer-wrapper">
  <div id="footer">
    <p class="copyright">&copy;  2010 Visos teisės saugomos.</p>
  </div>
```

```
</div>
</body>
</html>
```

5. GRAFINĖS VARTOTOJO SĄSAJOS GENERAVIMO IŠ UML DIAGRAMŲ ĮRANKIO PROJEKTAS

5.1 Grafinės vartotojo sąsajos generavimo metodo pagrindimas ir esmės išdėstymas

Grafinės vartotojo sąsajos generavimo metodas pagrįstas trijų UML diagramų, sudaromų sistemų projektavimo metu, panaudojimu. Panaudojimo atvejų, klasių ir veiklos diagramos sudaromos pagal duomenims keliamus reikalavimus.

Į grafinės vartotojos sąsajos generavimo įrankį įkeliamos diagramos, nubraižytos paketu *MagicDraw UML*. Kadangi viename projekte gali būti ne tik reikalingos diagramos, o ir daugelis kitų sistemų projektavime naudojamų diagramų, be to reikalingose diagramose gali būti naudojami ne tik sąsajos generavimui skirti elementai, panaudojimo atvejų, klasių bei veiklų diagramų aktualiems sąsajos generavimui elementams, t.y. panaudojimo atvejų diagramoje – panaudojimo atvejams, klasių diagramoje – klasėms, veiklos diagramoje – veikloms, pažymėti naudojami stereotipai. Tokiu būdu bus iškirti tik tie elementai, kurie dalyvaus vartotojo grafinės sąsajos generavime. Plačiau apie stereotipus bus aprašyta 6.3 skiltyje.

Panaudojimo atvejų diagramoje nurodoma, kurie panaudojimo atvejai kuriamoje sistemoje turi būti atvaizduojami kaip grafinės sąsajos langai. Klasių diagrama suprantama kaip kuriamos sistemos grafinės sąsajos elementų visuma. Pagal klasių atributus sukuriami grafiniai elementai, parenkant jiems atitinkantį grafinio elemento tipą. Klasės ir panaudojimo atvejai turi būti susieti, kadangi pats įrankis negalėtų atrinkti kurios klasės atributus pavaizduoti konkrečiame lange. Veiklos diagrama atspindi navigaciją tarp grafinės sąsajos langų. Navigacija tarp langų gali vykti meniu arba mygtukų pagalba.

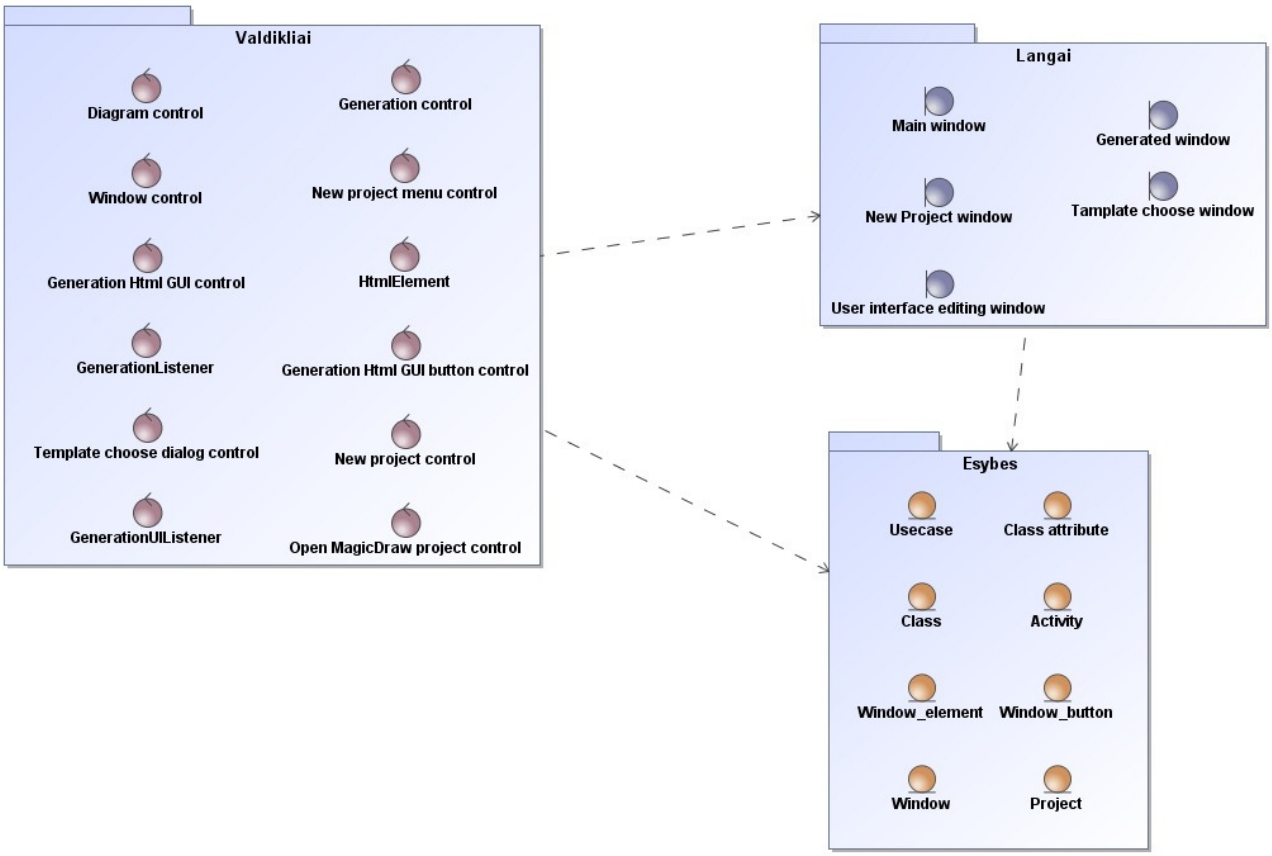
5.2 Sistemos architektūra

Sistemos architektūrą sudaro valdikliai, langai, esybės. Vartotojo sąsajos generavimo iš UML diagramų sistemos valdiklių paketą sudaro *Diagram control*, *Generation control*, Window *control*, *New project menu control*, *Generation HTML GUI control*, *HTML element*, *Generation Listener*, *Generation HTML GUI button control*, *Template choosing dialog control*, New *project control*, *Generation UI Listener*, *Open MagicDraw project control*.

Langai – *Main window*, *Generated window, New project window, Template choose window, User interface editing window.*

Pagrindinės esybės – *UseCase, Class_attribute, Class, Activity, Window_element, Window_button, Window, Project.*

Loginės sistemos architektūros diagrama pavaizduota 75 paveiksle.



75 pav. Loginė sistemos architektūra

Detalesnis loginės sistemos architektūros valdiklių aprašas pateiktas 23 lentelėje.

23 lentelė. Loginės sistemos architektūros valdiklių aprašas

Pavadinimas	**Aprašymas**
Diagram control	Diagramų valdiklis, kontroliuojantis jų nuskaitymą ir išsaugojimą.
Generation control	Aplikacijos grafinės vartotojo sąsajos generavimo valdiklis. Transformuoja diagramų metaklases į langų metaklases, dalyvauja atvaizduojant grafinę sąsają.
Window control	Grafinės vartotojo sąsajos langų valdiklis. Dalyvauja langų redagavime.
New project menu control	Atidaro naujo projekto sukūrimo langą.
Generation HTML GUI control	Internetinės grafinės vartotojo sąsajos generavimo valdiklis. Transformuoja diagramų metaklases į langų metaklases, dalyvauja atvaizduojant grafinę sąsają.
HTML element	Abstrakti klasė, naudojama aprašyti internetinės grafinės vartotojo sąsajos elementams.
Generation Listener	Paleidžia diagramų transformaciją.

Generation HTML GUI button control	Internetinės grafinės vartotojo sąsajos generavimo paleidimo valdiklis.
Template choosing dialog control	Valdiklis, kuris kontroliuoja internetinės grafinės sąsajos šablono pasirinkimą.
New project control	Valdiklis, kuris kontroliuoja naujo projekto sukūrimą.
Generation UI Listener	Inicijuoja grafinės vartotojo sąsajos generavimą.
Open MagicDraw project control	Valdiklis, kuris kontroliuoja *MagicDraw UML* įrankio XML failo įkėlimą į sistemą.

Detalesnis loginės sistemos architektūros langų aprašas pateiktas 24 lentelėje.

24 lentelė. Loginės sistemos architektūros langų aprašas

Pavadinimas	**Aprašymas**
Main window	Pagrindinis grafinės vartotojo sąsajos langas.
Generated window	Sugeneruotas grafinės vartotojo sąsajos langas.
New project window	Naujo projekto sukūrimo langas.
Template choose window	Internetinės grafinės vartotojo sąsajos šablono pasirinkimo langas.
User interface editing window	Grafinės vartotojo sąsajos redagavimo langas.

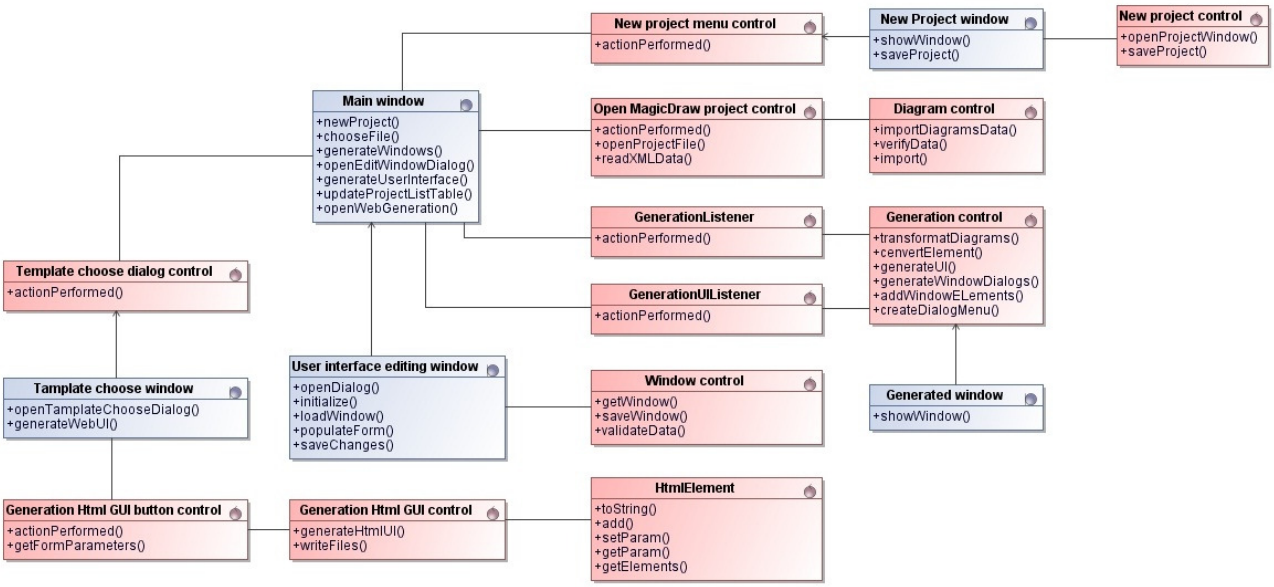
Detalesnis loginės sistemos architektūros esybių aprašas pateiktas 25 lentelėje.

25 lentelė. Loginės sistemos architektūros esybių aprašas

Pavadinimas	**Aprašymas**
UseCase	Panaudojimo atvejis iš UML veiklos diagramos.
Class_attribute	Klasės atributas iš UML klasių diagramos.
Class	Klasė iš UML klasių diagramos.
Activity	Veikla iš UML veiklos diagramos.
Window_element	Lango elementas, priklausantis tam tikram langui.
Window_button	Lango mygtukas, priklausantis tam tikram langui.
Window	Sugeneruotos grafinės vartotojo sąsajos langas.
Project	Projektas, kuriam generuojama grafinė vartotojo sąsaja.

5.3 Detalus projektas

Klasių diagrama su operacijomis pateikta 76 paveiksle.



76 pav. Detali grafinės sąsajos generavimo sistemos klasių diagrama

26 lentelėje pateiktas klasių operacijų aprašas.

26 lentelė. Klasių operacijų aprašymas

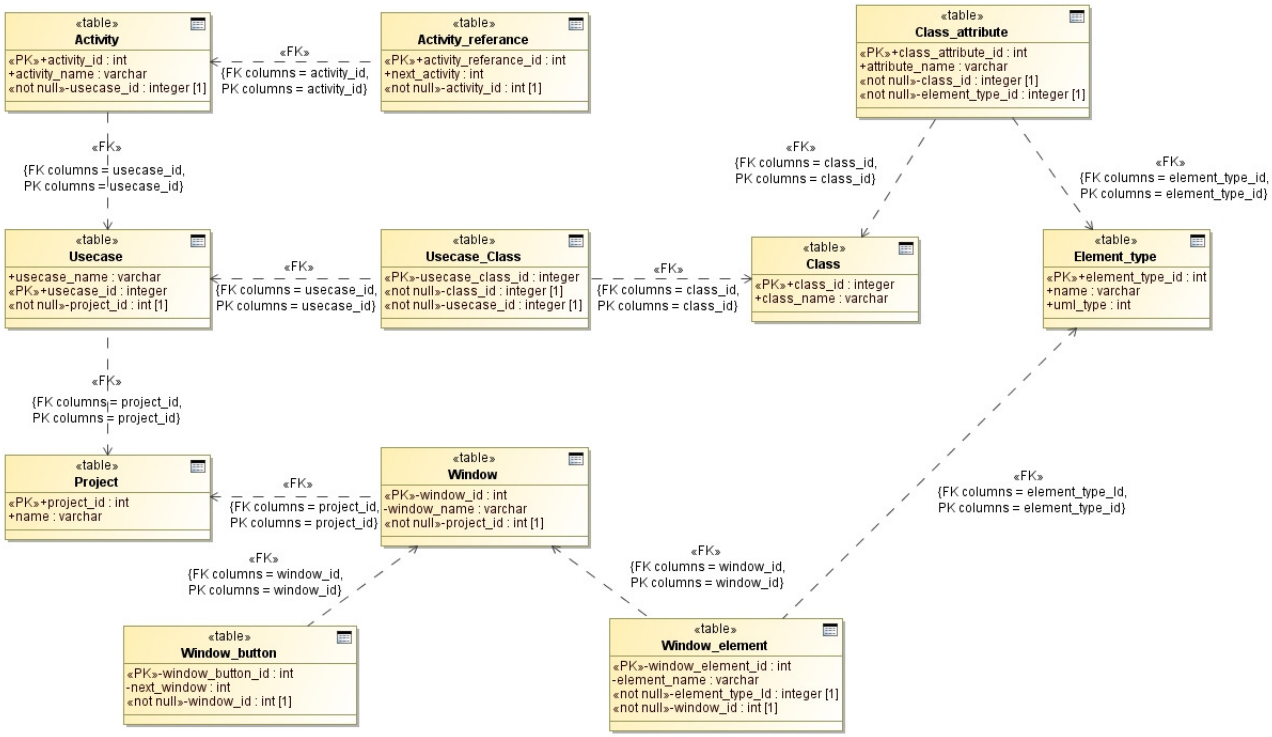
Klasė	**Operacijos pavadinimas**	**Aprašymas**
Main Window	*newProject()*	Iškviečia naujo projekto atidarymą.
	chooseFile()	Iškviečia failo pasirinkimo formą.
	generateWindows()	Iškviečia transformacijos vykdymo funkciją.
	openEditWindowDialog()	Iškviečia lango redagavimo formą.
	generateUserInterface()	Iškviečia programinės vartotojo sąsajos generavimo funkciją.
	updateProjectListTable()	Atnaujina projektų sąrašą.
	openWebGeneration()	Iškviečia internetinės sąsajos generavimą.
Diagram control	*importDiagramsData()*	Valdo diagramų įkėlimą ir tikrinimą.
	verifyData()	Patikrina diagramų informaciją.
	import()	Įkelia diagramas į duomenų bazę.
Generation control	*transformatDiagrams()*	Transformuoja diagramų metaklases į langų metaklases.
	cenvertElement()	Konvertuoja klasių diagramos elemento tipą į lango elemento tipą.
	generateUI()	Generuoja programinę grafinę sąsają.
	generateWindowDialogs()	Sukuria generuojamus langus.
	addWindowElements()	Įtraukia į dialogą lango elementus.
	createDialogMenu()	Įtraukia į dialogą sugeneruotą meniu.
Generation Html GUI	*actionPerformed()*	Iškviečia internetinės sąsajos generavimo

button control		funkciją.
	getFormParameters()	Gauna papildomus parametrus reikalingus internetinės sąsajos generavimui, kuriuos įveda vartotojas.
Generation Html GUI control	*generateHtmlUI()*	Sugeneruoja internetinę sąsają.
	writeFiles()	Įrašo internetinės sąsajos kodą į failus.
GenerationListener	*actionPerformed()*	Iškviečia diagramų transformacijos funkciją.
GenerationUIListener	*actionPerformed()*	Iškviečia programinės sąsajos generavimo funkciją.
HtmlElement	*toString()*	Grąžina suformuotą HTML elemento tekstą.
	add()	Įtraukia elementą.
	setParam()	Įtraukia papildomą parametrą, kuris bus naudojamas generuojant elemento kodą.
	getParam()	Grąžina įtrauktą elementą.
	getElements()	Grąžina įtrauktų elementų sąrašą.
New project control	*openProjectWindow()*	Atidaro naujo projekto kūrimo langą.
	saveProject()	Išsaugo naują projektą.
New project menu control	*actionPerformed()*	Iškviečia naujo projekto atidarymo funkciją.
Open MagicDraw project control	*actionPerformed()*	Iškviečia failo pasirinkimo dialogą.
	openProjectFile()	Atidaro *MagicDraw* projekto failą.
	readXMLData()	Nuskaito diagramų informaciją.
Template choose dialog control	*actionPerformed()*	Iškviečia internetinio šablono pasirinkimo langą.
Window control	*getWindow()*	Grąžina langą.
	saveWindow()	Išsaugo lango parametrus.
	validateData()	Patikrina įvestus duomenis.
Generated window	*showWindow()*	Atvaizduoja langą.
New Project window	*showWindow()*	Parodo ir nustato pradinius duomenis naujo projekto sukūrimo langui.
	saveProject()	Išsaugo projekto duomenis.
Tamplate choose window	*openTamplateChooseDialog()*	Atidaro šablono pasirinkimo langą.
	generateWebUI()	Iškviečia internetinės sąsajos generavimą.
User interface editing	*openDialog()*	Atidaro lango dialogą.

window	*initialize()*	Nustato pradinius formos duomenis.
	loadWindow()	Nuskaito lango duomenis.
	populateForm()	Įrašo lango duomenis į formos laukus.
	saveChanges()	Išsaugo lango pakeitimus.

5.4 Duomenų bazės schema

Duomenų bazės schema pavaizduota 77 paveiksle.



77 pav. Duomenų bazės schema

Activity

Lentelės „Activity“ atributų specifikacija pateikiama 27 lentelėje.

27 lentelė. Lentelės „Activity“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
activity_id	Integer	Taip	Pirminis raktas	Veiklos identifikacinis numeris
activity_name	Char	Taip		Veiklos pavadinimas
usecase_id	Integer	Taip	Išorinis raktas	Panaudojimo atvejo identifikacinis numeris

Activity_reference

Lentelės „Activity_reference“ atributų specifikacija pateikiama 28 lentelėje.

28 lentelė. Lentelės „Activity_reference“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
activity_reference_id	Integer	Taip	Pirminis raktas	Veiklos ryšio identifikacinis numeris
next_activity	Integer	Taip		Sekančios veiklos identifikacinis numeris
activity_id	Integer	Taip	Išorinis raktas	Veiklos identifikacinis numeris

UseCase

Lentelės „UseCase“ atributų specifikacija pateikiama 29 lentelėje.

29 lentelė. Lentelės „UseCase“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
usecase_name	Char	Taip		Panaudojimo atvejo pavadinimas
usecase_id	Integer	Taip	Pirminis raktas	Panaudojimo atvejo identifikacinis numeris
project_id	Integer	Taip	Išorinis raktas	Projekto identifikacinis numeris

Usecase_class

Lentelės „Usecase_class“ atributų specifikacija pateikiama 30 lentelėje.

30 lentelė. Lentelės „Usecase_class“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
usecase_class_id	Integer	Taip	Pirminis raktas	Lentelės identifikacinis numeris
class_id	Integer	Taip	Išorinis raktas	Klasės identifikacinis numeris
usecase_id	Integer	Taip	Išorinis raktas	Panaudojimo atvejo identifikacinis numeris

Class

Lentelės „Class“ atributų specifikacija pateikiama 31 lentelėje.

31 lentelė. Lentelės „Class“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
class_id	Integer	Taip		Klasės identifikacinis numeris
class_name	Char	Taip		Klasės pavadinimas

Element_type

Lentelės „Element_type“ atributų specifikacija pateikiama 32 lentelėje.

32 lentelė. Lentelės „Element_type“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
element_type_id	Integer	Taip	Pirminis raktas	Elemento tipo identifikacinis numeris
name	Char	Taip		Elemento tipo pavadinimas
uml_type	Integer	Taip		Elemento tipo aprašymas UML

Class_attribute

Lentelės „Class_attribute“ atributų specifikacija pateikiama 33 lentelėje.

33 lentelė. Lentelės „Class_attribute“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
class_attribute_id	Integer	Taip	Pirminis raktas	Klasės atributo identifikacinis numeris
attribute_name	Char	Taip		Klasės atributo pavadinimas
class_id	Integer	Taip	Išorinis raktas	Klasės identifikacinis numeris
class_name	Char	Taip	Išorinis raktas	Klasės pavadinimas

Project

Lentelės „Project“ atributų specifikacija pateikiama 34 lentelėje.

34 lentelė. Lentelės „Project“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
project_id	Integer	Taip	Pirminis raktas	Projekto identifikacinis numeris
project_name	Char	Taip		Projekto pavadinimas

Window

Lentelės „Window“ atributų specifikacija pateikiama 35 lentelėje.

35 lentelė. Lentelės „Window“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
window_id	Integer	Taip	Pirminis raktas	Lango identifikacinis numeris
window_name	Char	Taip		Lango pavadinimas
project_id	Integer	Taip	Išorinis raktas	Projekto identifikacinis numeris

Window_button

Lentelės „Window_button“ atributų specifikacija pateikiama 36 lentelėje.

36 lentelė. Lentelės „Window_button“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
window_button_id	Integer	Taip	Pirminis raktas	Lango mygtuko identifikacinis numeris
next_window	Integer	Taip		Sekančio lango, į kurį nukreipia mygtukas, identifikacinis numeris
window_id	Integer	Taip	Išorinis raktas	Lango identifikacinis numeris

Window_element

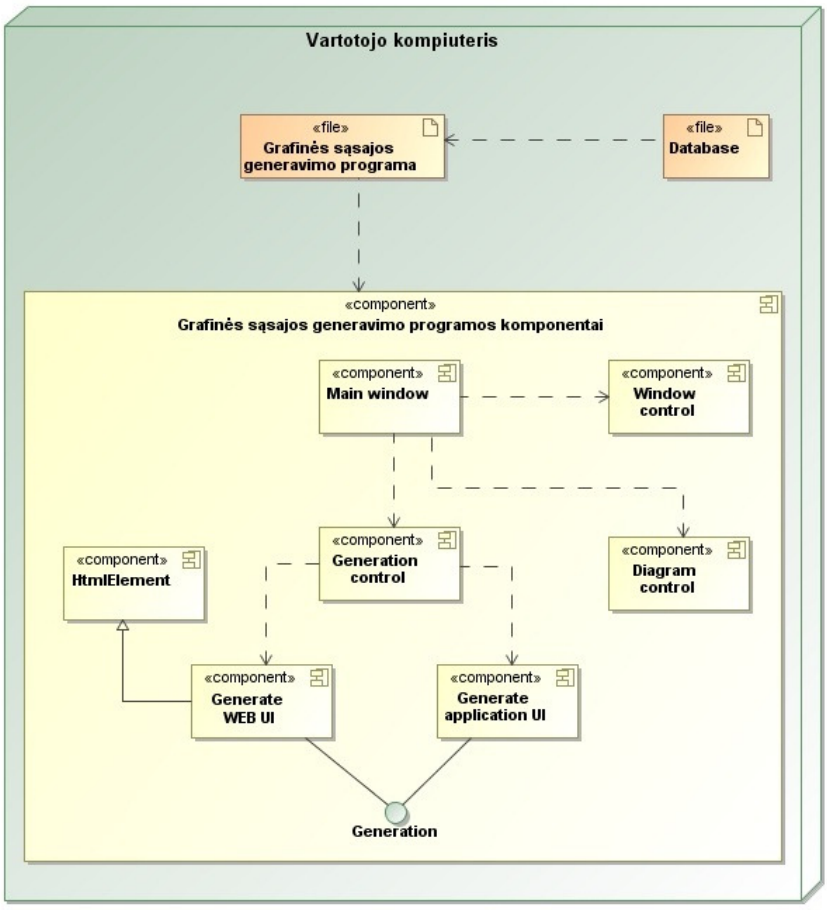
Lentelės „Window_element“ atributų specifikacija pateikiama 37 lentelėje.

37 lentelė. Lentelės „Window_element“ atributų specifikacija

Lauko pavadinimas	**Tipas**	**Ar būtinas?**	**Raktas ar indeksas?**	**Aprašymas**
window_element_id	Integer	Taip	Pirminis raktas	Lango grafinio elemento identifikacinis numeris
element_name	Char	Taip		Grafinio elemento pavadinimas
element_type_id	Integer	Taip	Išorinis raktas	Elemento tipo identifikacinis numeris
window_id	Integer	Taip	Išorinis raktas	Lango identifikacinis numeris

5.5 Realizacijos modelis

78 paveiksle pavaizduotame modelyje parodyti valdikliai, su kuriais sąveikauja grafinės vartotojo sąsajos generavimo įrankis.



78 pav. Programinės realizacijos modelis

6. REALIZACIJA

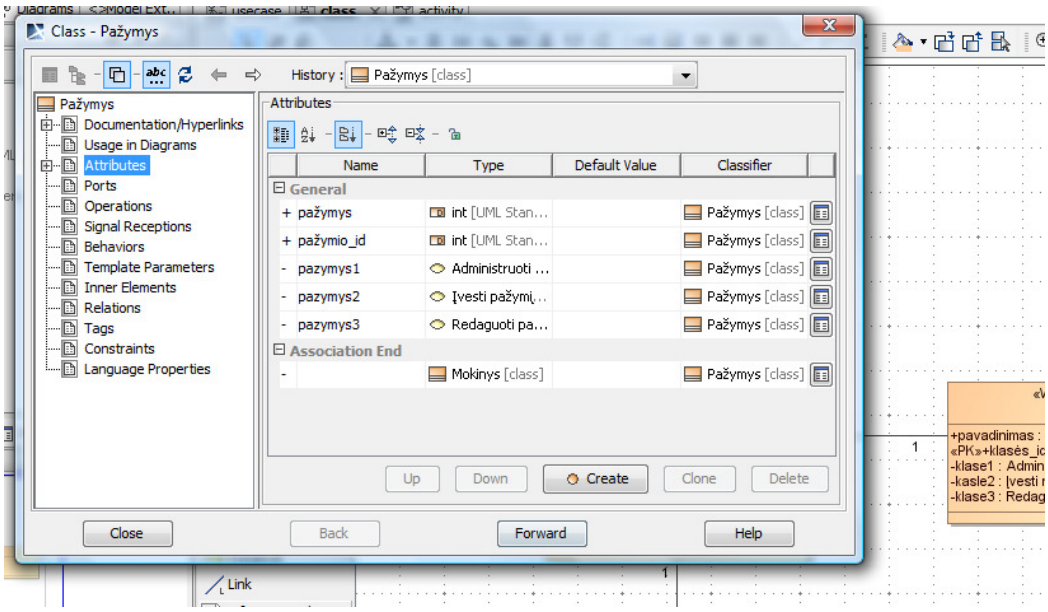
6.1 Diagramų susiejimas *MagicDraw* pakete

Įkeliamos į grafinės vartotojo sąsajos generavimo įrankį diagramos turi būti korektiškai sudarytos bei susietos tarpusavyje. Klasių ir veiklos diagramos privalo būti susietos su panaudojimo atvejais.

Kadangi įrankis pats savaime nesupranta, kuri klasė yra susiejama su tam tikru panaudojimo atveju, kiekvienoje klasėje reikia sukurti specifinius atributus su susiejamo panaudojimo atvejo tipu. Pavyzdžiui, klasę „Pažymys“ norint susieti su pažymio administravimo, įvedimo bei redagavimo sąsajos langais (t.y. panaudojimo atvejais) sukuriami trys atributai su atitinkamais panaudojimo atvejo tipais.

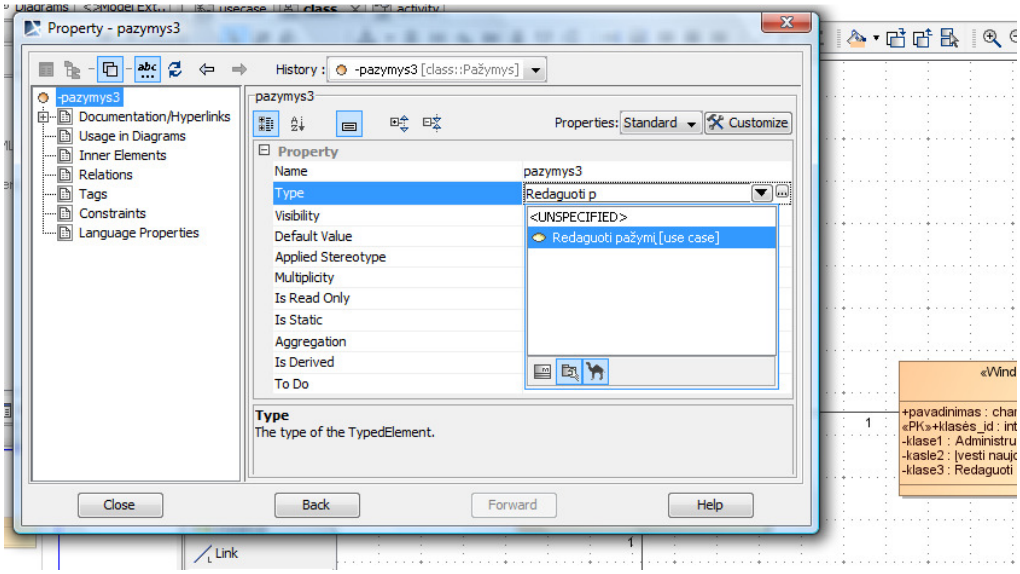
Norint sukurti klasę su panaudojimo atveju *MagicDraw UML* įrankyje atliekami šie veiksmai:

1. Pasirinktos klasės specifikacijoje pasirenkama *Attributes* skiltis;
2. Spaudžiamas mygtukas *Create* (79 pav.);



79 pav. Klasės „Pažymys“ specifikacijos langas pasirinkus *Attributes* skiltį

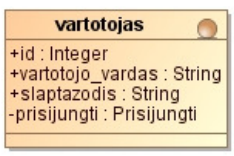
3. Įvedamas atributo pavadinimas bei pasirenkamas jo tipas (80 pav.);



80 pav. Klasės „Pažymys“ naujo atributo kūrimo langas

4. Mygtuko *Close* pagalba išsaugomi duomenys ir grįžtama į diagramos kūrimo langą.

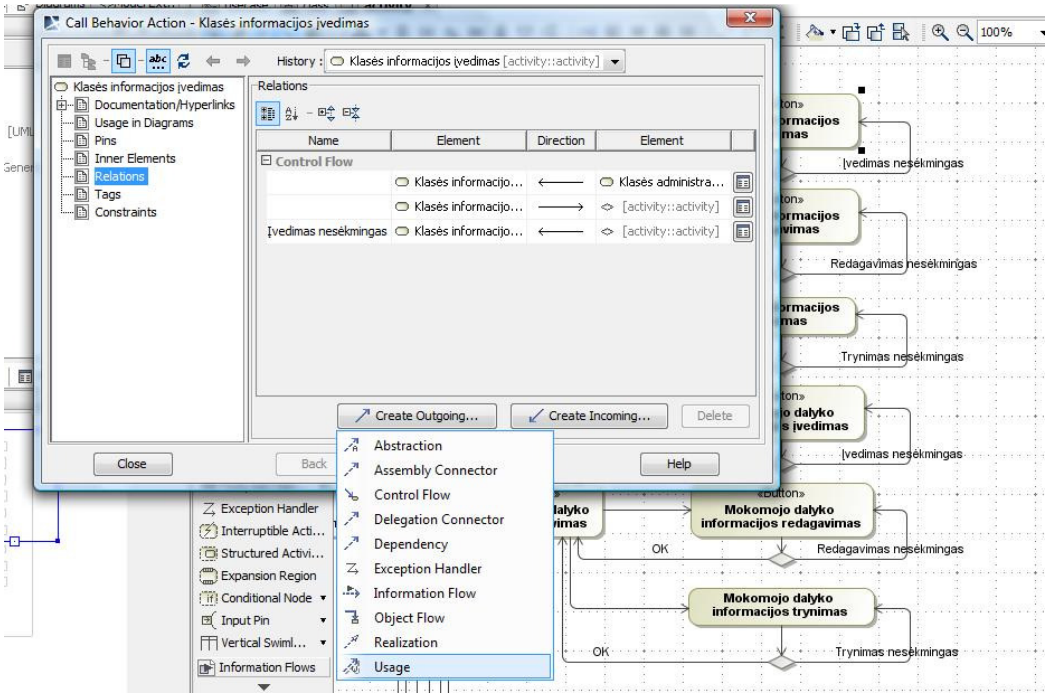
Žemiau pateiktame 81paveiksle matomas pridėtas „Prisijungti“ panaudojimo atvejis. Tai matoma iš – prisijungti:Prisijungti atributo.



81 pav. Panaudojimo atvejo ryšio nustatymo pavyzdys

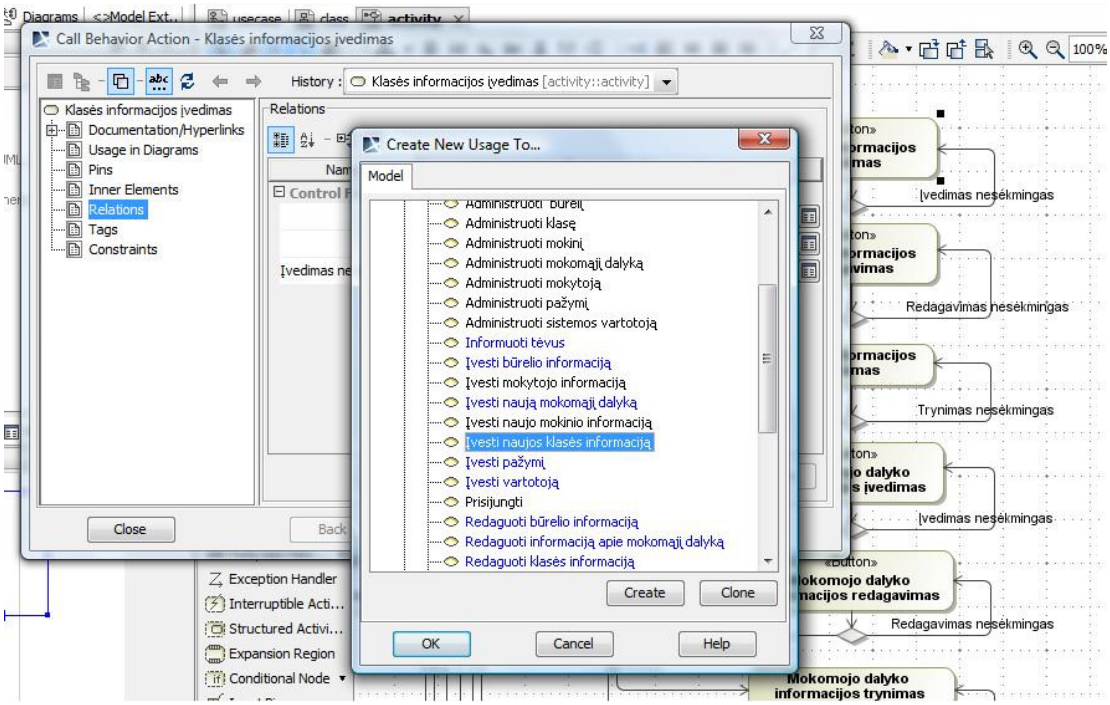
Veiklų diagramoje veiklos turi būti susietos su konkrečiais panaudojimo atvejais iš jau sudarytos Panaudojimo atvejų diagramos. Tai atliekama šiais veiksmais:

1. Pasirinktos veiklos specifikacijoje pasirenkama *Relations* skiltis;
2. Spaudžiamas mygtukas *Create Outgoing* ir pasirenkamas *Usage* ryšio tipas (82 pav.);



82 pav. Pasirinktos veiklos specifikacijos langas pasirinkus *Relations* skiltį

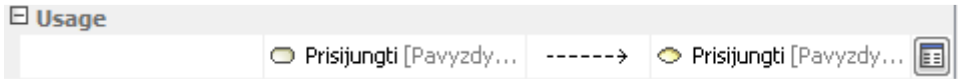
3. Atsidariusiame lange pažymimas panaudojimo atvejis, su kuriuo siejama pasirinkta veikla (83 pav.) ir paspaudžiamas mygtukas *OK*;



83 pav. Pasirinktos veiklos susiejimas su esamu panaudojimo atveju

4. Mygtuko *Close* pagalba išsaugomi duomenys ir grįžtama į diagramos kūrimo langą.

84 paveiksle pavaizduotas ryšys tarp veiklos ir panaudojimo atvejo.



84 pav. „Prisijungti“ veiklos susiejimas

6.2 *MagicDraw* XML dokumento nuskaitymas

MagicDraw XML dokumento atributai:

packagedElement – saugoma pagrindinė paketo informacija, atskiriama pagal *xmi:type* atributą. Elemento atributai pateikiami 38 lentelėje.

38 lentelė. packagedElement atributai

Atributai	**Aprašymas**	**Elementai**
uml:UseCase	Saugomi panaudojimo atvejo pavadinimas ir ID numeris	
uml:Class	Saugomi klasės pavadinimas ir ID numeris	*ownedAttribute, type*
uml:Activity	Saugomi veiklų pavadinimai ir ID numeriai	*node, edge*
uml:Usage	Išsaugo veiklos ir panaudojimo atvejo susiejimo duomenis	*supplier, client*

Stereotipe – saugoma stereotipų informacija (39 lentelė).

39 lentelė. Stereotipe atributai

Tipas	**Aprašymas**
name	Stereotipo pavadinimas
stereotypeID	Stereotipo ID numeris

OwnedAttribute – saugomi klasių atributai ir jų tipai (40 lentelė).

40 lentelė. OwnedAttribute atributai

Tipas	**Aprašymas**
name	Klasės atributo pavadinimas
xmi:id	Klasės atributo ID numeris

Node – saugomas veiklos pavadinimas (41 lentelė).

41 lentelė. Node atributai

Tipas	**Aprašymas**

name	Veiklos pavadinimas
xmi:id	Veiklos ID numeris

Edge – naudojamas atpažinti ryšius tarp veiklų (42 lentelė).

42 lentelė. Edge atributai

Tipas	**Aprašymas**
source	Veiklos ID numeris
target	Susietos sekančios veiklos ID numeris

Supplier – saugoma panaudojimo atvejo ID numeris (43 lentelė).

43 lentelė. Supplier atributai

Tipas	**Aprašymas**
xmi:idref	Panaudojimo atvejo ID numeris

Client – saugoma veiklos ID numeris (44 lentelė).

44 lentelė. Client atributai

Tipas	**Aprašymas**
xmi:idref	Veiklos ID numeris

6.2.1 XML dokumentų nuskaitymo būdai

Pagrindiniai XML dokumentų nuskaitymo būdai – DOM ir SAX.

DOM (angl. *Document Object Model)* – visa dokumento struktūra nuskaitoma į atmintį ir visi veiksmai atliekami su atmintyje nuskaitytu dokumento elementų medžiu.

SAX (angl. *Simple API for XML*) – įvykiais grindžiamas XML dokumento nuskaitymas. Tokiu būdu nuskaitant dokumentą yra iškviečiamos funkcijos prieš pradedant ir baigiant skaityti dokumentą bei prieš pradedant skaityti ir baigiant skaityti elementą. Tokiu būdu galima nuskaityti reikiamą informaciją iš dokumento neįkeliant viso dokumento į atmintį.

Pasirinktas dokumento nuskaitymo būdas yra įvykiais grindžiamas (SAX). Šiam metodui įgyvendinti yra naudojamos keturios funkcijos *startDocument*, *endDocument*, *startElement*, *endElement*. Funkcijų aprašymas pateiktas 45 lentelėje.

45 lentelė. Funkcijų aprašymas

Funkcija	**Aprašymas**
startDocument()	Pirma funkcija, kuria pradedamas dokumento nuskaitymas. Gali būti iškviečiamas vieną kartą.
endDocument()	Gaunamas pranešimas apie dokumento pabaigą. Gali būti iškviečiamas tik kartą.
startElement(String namespaceURI, String localName, String qName, Attributes atts)	Funkcija, nuskaitanti elemento duomenis.
endElement()	Gaunamas pranešimas apie nuskaitomo elemento pabaigą.

Plačiau panagrinėsime funkciją *startElement().* Funkcija *startElement()* atrenka tik šiuos elementus: *packagedElement*, *ownedAttribute*, *node*, *supplier*, *client*, *edge* ir *referenceExtension*.

Jeigu elemento pavadinimas atitinka *packagedElement,* tikrinamas jo tipas. Elemento *packagedElement* tipai ir su jais susiję veiksmai pateikiami 46 lentelėje.

46 lentelė. Elemento packagedElement tipai ir su jais susiję veiksmai

***packagedElement* tipas**	**Atliekamas veiksmas**
uml:UseCase	Išsaugomas panaudojimo atvejis į masyvą.
uml:Class	Sukuriamas klasės objektas, į kurį bus surašyti atributai ir klasės pavadinimas.
uml:Activity	Nurodoma programai, kad toliau reikės saugoti veiklos diagramos elementus *node*. Šiame pakete saugoma veiklos diagramos veiklos.
uml:Usage	Nurodoma programai, kad toliau reikės išsaugoti veiklos ryšius su panaudojimo atveju. Šiame pakete yra saugojami veiklos ir panaudojimo atvejo ryšiai.

Elemento *node* tipai ir su jais susiję veiksmai pateikiami 47 lentelėje.

47 lentelė. Elemento node tipai ir su jais susiję veiksmai

***node* tipas**	**Atliekamas veiksmas**
uml:CallBehaviorAction	Išsaugomas veiklos pavadinimas.
uml:InitialNode	Išsaugomas veiklos diagramos pradžios elementas.
uml:ActivityFinalNode	Išsaugomas veiklos diagramos pabaigos elementas.

Pagal elemento pavadinimą atliekami veiksmai pateikiami 48 lentelėje.

48 lentelė. Elementai ir pagal juos atliekami veiksmai

Elemento pavadinimas	**Atliekamas veiksmas**
supplier arba *client*	Išsaugomas ryšys tarp veiklos ir panaudojimo atvejo.
edge	Išsaugomas veiklos diagramos ryšys tarp elementų.
ownedAttribute	Pridedamas klasės atributas prie jau sukurto klasės objekto.
referenceExtension	Išsaugomas sukurto atributo tipas.

Funkcija *endElement()* atrenka tik *packagedElement* elementus. Šioje funkcijoje atliekamas tikrinimas, ar jau buvo išsaugoti visi klasei priskirti atributai. Jeigu taip, tada klasė išsaugoma į masyvą.

Baigus failo nuskaitymą, atliekamas kiekvieno elemento filtravimas pagal stereotipus ir grąžinami tik tie elementai, kuriems buvo priskirti atitinkami stereotipai. Atlikus filtravimą duomenys įgauna UML metamodelio duomenų struktūrą.

Naudojantis 4.5 skyriuje aprašytu algoritmu atliekama UML diagramų metaklasių transformacija į grafinės sąsajos metaklases, kurios, sukūrus sąsajos langus, atvaizduojamos grafiniais elementais.

Iš pradžių sukuriami vartotojo sąsajos dialogai, atitinkantys nuskaitytus grafinės sąsajos langus. Į kiekvieną dialogą yra sukeliami lango elementai tikrinant jo tipą. Kiekviena programavimo kalba skirtingai aprašo grafinius elementus, todėl yra būtinas grafinės sąsajos elemento tikrinimas pagal tipą.

6.3 Diagramų stereotipai ir *MagicDraw* profilio *GUI* naudojimas

Į grafinės vartotojos sąsajos generavimo įrankį įkeliamos diagramos, nubraižytos paketu *MagicDraw UML.* Kaip jau minėta, grafinės sąsajos generavime aktualiems panaudojimo atvejų, klasių bei veiklų diagramų elementams pažymėti naudojami specifiniai stereotipai. Tokiu būdu bus išskiriami tik tie elementai, kurie dalyvaus vartotojo grafinės sąsajos generavime. Taip pat tai padės įrankiui atpažinti elementus.

Šiems stereotipams aprašyti yra sukurtas *MagicDraw UML* profilis *GUI*. *GUI* profilio failą reikia įkelti į *MagicDraw UML profiles* direktoriją kompiuteryje, o pakete jį įkelti naudojantis

File -> Use Module funkcija bei pasirinkus *GUI* profilį.

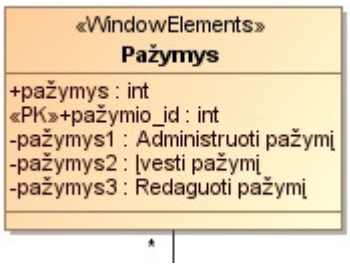
GUI profilį sudarantys stereotipai pateikti 85 paveiksle.

85 pav. Stereotipų klasės

Panaudojimo atvejų diagramoje nurodoma, kurie panaudojimo atvejai kuriamoje sistemoje turi būti atvaizduojami kaip grafinės sąsajos langai. Šiems panaudojimo atvejams naudojamas *Window* stereotipas. Panaudojimo atvejo pavyzdys su šiuo stereotipu pavaizduotas 86 pav.

86 pav. Panaudojimo atvejis „Administruoti mokytoją“ su *Window* stereotipu

Klasių diagramoje taip pat taikomi tam tikri apribojimai. Klasių diagrama suprantama kaip kuriamos sistemos grafinės sąsajos elementų visuma. Klasės, kurios turi sąryšį su konkrečiais panaudojimo atvejais, pažymimos stereotipu *WindowElements* ir sukuriami atributai su siejamo panaudojimo atvejo tipu. Klasės pavyzdys su minėtu stereotipu ir su sukurtais atributais pavaizduotas 87 paveiksle.



87 pav. Klasės pavyzdys su *WindowElements* stereotipu ir sukurtais atributais

Veiklos diagramoje naudojamas stereotipas *Button*. Veiklos diagramoje stereotipu *Button* pažymimos tos veiklos, kurios atspindi navigaciją tarp grafinės sąsajos langų. Šios veiklos languose bus sukurtos kaip mygtukai su veiklos pavadinimu. Naudojantis elementų redagavimo funkcija įrankyje juos galima keisti.

88 pav. Pavyzdinė veikla su *Button* stereotipu

Kadangi įrankis pats savaime nesupranta, kuri veikla yra susijusi su konkrečiu panaudojimo atveju, tai reikia nurodyti juos susiejant pačiame *MagicDraw UML* įrankyje. Tai

atliekama specifikacijos lange sukuriant *Outgoing Usage* ryšį ir parenkant jau sukurtą panaudojimo atvejį.

Minėtų stereotipų trumpas aprašas pateiktas 49 lentelėje.

49 lentelė. Stereotipai bei jų aprašymai

Stereotipas	**Aprašymas**
Window	Aprašo generuojamos grafinės vartotojo sąsajos langus.
Button	Aprašo generuojamos grafinės vartotojo sąsajos langų mygtukus.
WindowElements	Aprašo generuojamos grafinės vartotojo sąsajos langų elementus.

6.4 Klasių atributų tipų ir grafinių elementų tipų atitikmenys

MagicDraw UML įrankyje galimi klasių atributų tipai – *integer, int, string, char, date, double, radio, boolean, bool, textarea, combobox.*

Kai klasės atributo tipas yra *integer, int, string, char, date* arba *double*, grafinės vartotojo sąsajos formoje jis bus atvaizduotas *TextField* lauku, o jo pavadinimas – *Label*. *Radio* klasės atributo vartotojo sąsajos formoje bus atvaizduotas *RadioButton* elementu, *boolean* arba *bool* – *CheckBox* elementu, *textarea* tipo atributai – *TextArea* elementu, o *combobox* – *ComboBox* elementu.

Lentelėje (angl. *Table*) gali būti atvaizduojami visi atributų tipai.

6.5 Sistemos veikimo aprašymas

6.5.1 Sistemos instaliavimas ir paruošimas darbui

Korektiškam sistemos paleidimui kompiuteryje turi būti įdiegta ne mažesnė nei *Java 1.6* versija. Sistemos programiniai failai išsaugomi pasirinktoje kategorijoje.

Sistemos darbas pradedamas paleidus *GUI.exe* failą.

6.5.2 Pagrindiniai darbo principai

Grafinės vartotojo sąsajos įrankio valdymas vyksta meniu ir mygtukų pagalba. Sistemą paleidus vartotojo kompiuteryje atidaromas pagrindinis langas su jau sukurtų projektų sąrašu bei pagrindiniais jų duomenimis.

Norint sukurti naują projekto failą, pasirenkamas meniu punktas *Failas* -> *Naujas projektas*; norint dirbti su jau sukurtu projektu – dukart pelės klavišu paspaudžiamas projekto pavadinimas projektų sąraše.

Sukūrus projektą, pirmiausia reikia įkelti *MagicDraw UML* projekto XML failą. Tik tuomet sistema leis generuot grafinę vartotojo sąsają, ją redaguoti, kurti naujus langus/elementus bei vykdyti langų peržiūrą. Keliskart generuoti to pačio projekto grafinės

sąsajos negalima. Tolimesnis darbas su sugeneruota grafine sąsaja vykdomas redaguojant bei kuriant naujus langus, jų elementus ar mygtukus. Jie išsaugomi duomenų bazėje ir peržiūros metu bus matoma grafinė sąsaja su jau atliktais pakeitimais.

6.5.3 Grafinės sąsajos generavimo įrankio funkcijos ir jo valdymas

Įrankio valdymas vyksta meniu pagalba. Meniu parinktys ir paaiškinimai pateikti 50 lentelėje.

50 lentelė. Grafinės vartotojo sąsajos generavimo įrankio meniu parinktys

Meniu pavadinimas		**Aprašymas**
Pirmo lygio meniu	**Antro lygio meniu**	
Failas	Naujas projektas	Naujo projekto sukūrimo funkcija.
	Įkelti *MagicDraw UML* projekto failą	Įkelia pasirinktą *MagicDraw UML* projekto failą.
	Uždaryti projektą	Uždaro projektą.
	Išeiti	Išjungia sistemą.
Veiksmai	Generuoti	Sugeneruoja grafinę vartotojo sąsają su informacija iš įkelto *MagicDraw UML* failo.
	Naujų langų įtraukimas	Iškviečia langų įtraukimo langą.
	Langų redagavimas	Iškviečia langų redagavimo langą.
Peržiūra	Peržiūrėti aplikacijos grafinę sąsają	Sugeneruotos grafinės vartotojo sąsajos peržiūra aplikacijai.
	Peržiūrėti internetinę grafinę sąsają	Sugeneruotos grafinės vartotojo sąsajos peržiūra internetinei sąsajai.
Pagalba	Apie	Grafinės vartotojo sąsajos generavimo įrankio aprašymas.

Grafinės vartotojo sąsajos įrankio funkcijos ir jų veikimas plačiau aprašytas Priede nr. 1 pateiktame Vartotojo vadove.

6.6 Testavimo modelis bei duomenys

Grafinės vartotojo sąsajos iš UML diagramų generavimo įrankio testavimo tikslas yra išsiaiškinti, ar sistema veikia korektiškai.

Sistemos testavimui yra sudaromas testavimo planas, kuriame atsispindi, kokius veiksmus reikia atlikti testuojant sistemą, taip pat, kokia yra laukiama sistemos reakcija į atliekamus veiksmus. Sistemos testavimo planas yra pateiktas 51 lentelėje.

51 lentelė. Grafinės vartotojos sąsajos generavimo įrankio testavimo planas

1. Sukurti naują projektą		
1.1	Neįvedamas joks pavadinimas ir spaudžiamas mygtukas „Sukurti“.	Pranešimas „Įveskite projekto pavadinimą“.
1.2	Įvedamas pavadinimas ir spaudžiamas mygtukas „Sukurti“.	Sukuriamas naujas projektas, kuris rodomas projektų sąraše, su įvestu pavadinimu.
2. Ištrinti projektą		
2.1	Projektų sąraše nepasirenkamas projektas ir spaudžiamas mygtukas „Trinti“.	Pažymimas projektas, kurio eilutėje paspaustas mygtukas „Trinti“. Parodomas pranešimas “Ar tikrai norite ištrinti projektą?“.
2.2	Pranešime “Ar tikrai norite ištrinti projektą?“ pasirenkamas mygtukas „Taip“.	Ištrinamas pasirinktas projektas.
2.3	Pranešime “Ar tikrai norite ištrinti projektą?“ pasirenkamas mygtukas „Ne“.	Pasirinktas projektas neištrinamas. Grįžtama į projektų sąrašą.
3. Įkelti *MagicDraw* projektą		
3.1	Nepasirenkamas joks failas.	Neįkeliamas failas.
3.2	Pasirenkamas *.mdzip formato failas.	Įkeliamas *MagicDraw UML* projekto failas sukurtam projektui. Pranešimas “*MagicDraw* projektas išsaugotas duomenų bazėje. Galite generuoti grafinę vartotojo sąsają“.
4. Uždaryti projektą		
4.3	Pasirenkama funkcija „Uždaryti projektą“.	Uždaromas projektas. Įrankio pavadinime ištrinamas uždaromo projekto pavadinimas.
5. Išeiti		

5.1	Pasirenkama funkcija „Išeiti“.	Uždaromas grafinės sąsajos generavimo įrankis.
6. Generuoti		
6.1	Projektui nėra įkeltas *MagicDraw UML* projekto failas. Bandoma pasinaudoti funkcija „Generuoti“.	Pranešimas „Projektui nėra įkeltas *MagicDraw* projekto failas.“
6.2	Projektui yra įkeltas *MagicDraw* projekto failas. Bandoma pasinaudoti funkcija „Generuoti“.	Sugeneruojama grafinė vartotojo sąsaja. Pranešimas „Generavimas atliktas sėkmingai“.
7. Naujų langų įtraukimas		
7.1	Projektui nėra įkeltas *MagicDraw UML* projekto failas. Bandoma pasinaudoti funkcija „Naujų langų įtraukimas“.	Pranešimas „Projektui nėra įkeltas *MagicDraw* projekto failas.“
7.2	Projektui yra įkeltas *MagicDraw* projekto failas. Bandoma pasinaudoti funkcija „Naujų langų įtraukimas“.	Parodomas naujų langų įtraukimo langas.
7.3	Naujų langų įtraukimo lange paspaudžiamas mygtukas „Įtraukti langą“.	Pranešimas „Įveskite lango pavadinimą“.
7.4	Naujų langų įtraukimo lange paspaudžiamas mygtukas „Įtraukti langą“ įvedus lango pavadinimą.	Pranešimas „Įveskite nors vieną lango elementą.“
7.5	Naujų langų įtraukimo lange paspaudžiamas mygtukas „Įtraukti langą“ įvedus lango pavadinimą ir bent vieną elementą.	Sukuriamas naujas projekto langas su įvestais duomenimis. Jis rodomas langų sąraše. Jo duomenis galima redaguoti naudojantis langų redagavimo funkcija.
7.6	Naujų langų įtraukimo lange lango elementų įtraukimo skiltyje paspaudžiamas mygtukas „Trinti“.	Pranešimas „Pasirinkite elementą, kurį norite ištrinti“.
7.6.1	Naujų langų įtraukimo lange lango elementų įtraukimo skiltyje paspaudžiamas mygtukas „Trinti“	Pasirinktas elementas ištrinamas iš kuriamo lango. Jis neberodomas elementų sąraše.

	iš sąrašo pasirinkus elementą.	
7.7	Naujų langų įtraukimo lange lango elementų skiltyje paspaudžiamas mygtukas „Įtraukti“ neįvedus elemento duomenų.	Pranešimas „Įveskite elemento pavadinimą“.
7.7.1	Naujų langų įtraukimo lange lango elementų skiltyje paspaudžiamas mygtukas „Įtraukti“ įvedus elemento pavadinimą.	Sukuriamas naujas elementas kuriamame lange. Jo duomenys rodomi kairiau esančiame elementų sąraše.
7.8	Naujų langų įtraukimo lange lango mygtukų įtraukimo skiltyje paspaudžiamas mygtukas „Ištrinti“ nepasirinkus mygtuko iš sąrašo.	Pranešimas „Ar tikrai norite ištrinti mygtuką?“. Pasirinkus teigiamą atsakymą, parodomas pranešimas „Pasirinkite mygtuką.“, pasirinkus neigiamą atsakymą – grįžtama į mygtukų įtraukimo langą.
7.8.1	Naujų langų įtraukimo lange lango mygtukų įtraukimo skiltyje paspaudžiamas mygtukas „Ištrinti“ pasirinkus mygtuką iš sąrašo.	Pranešimas „Ar tikrai norite ištrinti mygtuką?“. Pasirinkus teigiamą atsakymą ištrinamas pasirinktas mygtukas, jis neberodomas mygtukų sąraše, pasirinkus neigiamą atsakymą – grįžtama į mygtukų įtraukimo langą, mygtukas neištrinamas.
7.9	Naujų langų įtraukimo lange lango mygtukų įtraukimo skiltyje paspaudžiamas mygtukas „Įtraukti mygtuką“ neįvedus naujo mygtuko duomenų.	Pranešimas „Įveskite mygtuko pavadinimą“.
7.9.1	Naujų langų įtraukimo lange lango mygtukų įtraukimo skiltyje paspaudžiamas mygtukas „Įtraukti mygtuką“ įvedus mygtuko pavadinimą.	Sukuriamas naujas mygtukas su įvestu pavadinimu. Mygtukas rodomas mygtukų sąraše.
7.10	Panaudojimo atvejų sąraše pasirenkamas panaudojimo atvejis ant jo pavadinimo paspaudus su kairiu pelės klavišu.	Dešiniau esančiame bloke parodomos su pasirinktu panaudojimo atveju susietos klasės.

7.11	Klasių sąraše pasirenkama klasė ant jos pavadinimo paspaudus su kairiu pelės klavišu.	Galimų atributų sąraše parodomi tos klasės atributai, kurie gali būti įkeliami į langą.
8. Langų redagavimas		
8.1	Projektui nėra įkeltas *MagicDraw* projekto failas. Bandoma pasinaudoti funkcija „Langų redagavimas“.	Pranešimas „Projektui nėra įkeltas *MagicDraw* projekto failas.“
8.2	Projektui yra įkeltas *MagicDraw* projekto failas. Bandoma pasinaudoti funkcija „Langų redagavimas“.	Parodomas langų redagavimo langas.
8.3	Langų redagavimo lange lango elementų redagavimo skiltyje paspaudžiamas mygtukas „Išsaugoti“ nepasirinkus sugeneruotos sąsajos lango.	Pranešimas „Pasirinkite redaguojamą langą“
8.4	Langų redagavimo lange lango elementų redagavimo skiltyje paspaudžiamas mygtukas „Naujas“ nepasirinkus sugeneruotos sąsajos lango.	Pranešimas „Pasirinkite redaguojamą langą“.
8.5	Langų redagavimo lange lango elementų redagavimo skiltyje paspaudžiamas mygtukas „Ištrinti“ nepasirinkus sugeneruotos sąsajos lango.	Pranešimas „Ar tikrai norite ištrinti pasirinktą lango elementą?“ Pasirinkus teigiamą atsakymą, parodomas pranešimas „Pasirinkite redaguojamą langą“. Pasirinkus neigiamą atsakymą grįžtama į langų redagavimo langą.
8.6	Langų redagavimo lange lango elementų redagavimo skiltyje paspaudžiamas mygtukas „Atnaujinti elementą“ nepasirinkus sugeneruotos sąsajos lango.	Pranešimas „Pasirinkite elementą, kurį norite atnaujinti.“
8.7	Langų redagavimo lange lango	Pranešimas „Ar tikrai norite ištrinti

	mygtukų redagavimo skiltyje paspaudžiamas mygtukas „Ištrinti“ nepasirinkus sugeneruotos sąsajos lango.	mygtuką?“. Pasirinkus teigiamą atsakymą, parodomas pranešimas „Pasirinkite mygtuką“. Pasirinkus neigiamą atsakymą grįžtama į mygtukų redagavimo langą.
8.8	Langų redagavimo lange lango mygtukų redagavimo skiltyje paspaudžiamas mygtukas „Atnaujinti mygtuką“ nepasirinkus sugeneruotos sąsajos lango.	Pranešimas „Pasirinkite mygtuką“.
8.9	Pasirenkamas langas paspaudžiant ant jo pavadinimo. Spaudžiamas mygtukas „Redaguoti“.	Parodomas redaguojamo lango pavadinimas. Lango elementų skiltyje atspausdinamas to lango elementų sąrašas parodant jų tipus.
8.10	Pasirenkamas langas paspaudžiant ant jo pavadinimo. Spaudžiamas mygtukas „Trinti“.	Pranešimas „Ar tikrai norite ištrinti langą?“ Pasirinkus teigiamą atsakymą, ištrinamas langas ir jo duomenys. Jis neberodomas langų sąraše. Pasirinkus neigiamą atsakymą grįžtama į langų redagavimo langą.
8.11	Pasirenkamas langas paspaudžiant ant jo pavadinimo. Spaudžiamas mygtukas „Peržiūrėti“.	Parodomas pasirinktas langas peržiūros režime. Langas rodomas aplikacijai.
8.1. Langų redagavimas: Lango elementų redagavimas		
8.1.1	Pasirinkto lango redagavimo lange lango elementų redagavimo skiltyje paspaudžiamas mygtukas „Naujas“.	Tame pačiame lange parodoma naujo elemento kūrimo forma.
8.1.1.1	Naujo elemento kūrimo formoje paspaudžiamas mygtukas „Įtraukti elementą“ neįvedus duomenų.	Pranešimas „Įveskite elemento pavadinimą“.
8.1.1.2	Naujo elemento kūrimo formoje įvedamas pavadinimas ir paspaudžiamas mygtukas „Įtraukti elementą“.	Naujas lango elementas išsaugomas ir parodomas prie pasirinkto lango elementų sąraše.
8.1.2	Pasirinkto lango redagavimo lange lango elementų redagavimo skiltyje	Pranešimas „Ar tikrai norite ištrinti pasirinktą lango elementą?“ Pasirinkus

	paspaudžiamas mygtukas „Ištrinti“.	teigiamą atsakymą, parodomas pranešimas „Pasirinkite lango elementą“. Pasirinkus neigiamą atsakymą grįžtama į langų redagavimo langą.
8.1.3	Pasirinkto lango redagavimo lange lango elementų redagavimo skiltyje pasirenkamas elementas paspaudžiant ant jo pele.	Pasirinkto lango elemento duomenys parodomi dešiniau esančioje formoje.
8.1.4	Pasirinkto lango redagavimo lange lango elementų redagavimo skiltyje pasirenkamas elementas ir paspaudžiamas mygtukas „Ištrinti“.	Parodomas pranešimas „Ar tikrai norite ištrinti pasirinktą lango elementą?“ Pasirinkus teigiamą atsakymą, ištrinamas elementas. Jis neberodomas elementų sąraše. Pasirinkus neigiamą atsakymą grįžtama į langų redagavimo langą.
8.1.5	Lango elemento redagavimo formoje ištrinamas elemento pavadinimas. Spaudžiamas mygtukas „Atnaujinti elementą“.	Pranešimas „Įveskite elemento pavadinimą“.
8.2. Langų redagavimas: Lango parametrų redagavimas		
8.2.1	Pasirinkto lango redagavimo lange lango parametrų redagavimo skiltyje ištrinamas lango pavadinimas. Spaudžiamas mygtukas „Išsaugoti“.	Pranešimas „Įveskite lango pavadinimą“.
8.2.2	Pasirinkto lango redagavimo lange lango parametrų redagavimo skiltyje pasirenkamas atvaizdavimo tipas „Atvaizduoti kaip formą“.	Pasirinktame lange elementai atvaizduojami formoje.
8.7.3	Pasirinkto lango redagavimo lange lango parametrų redagavimo skiltyje pasirenkamas atvaizdavimo tipas „Atvaizduoti kaip sąrašą“.	Pasirinktame lange duomenys atvaizduojami sąraše.
8.7.4	Pasirinkto lango redagavimo lange lango parametrų redagavimo	Pasirinktame lange elementai atvaizduojami formoje ir sąrašo pavidalu.

	skiltyje pasirenkamas atvaizdavimo tipas „Atvaizduoti kaip formą ir sąrašą“.	
8.7.5	Pasirinkto lango redagavimo lange lango parametrų redagavimo skiltyje pasirenkamas atvaizdavimo tipas „Nerodyti jokių elementų“.	Pasirinktame lange elementai neatvaizduojami.
8.3. Langų redagavimas: Lango mygtukų redagavimas		
8.3.1	Pasirinkto lango redagavimo lange mygtukų redagavimo skiltyje paspaudžiamas mygtukas „Ištrinti“.	Pranešimas „Ar tikrai norite ištrinti mygtuką?“. Pasirinkus teigiamą atsakymą, parodomas pranešimas „Pasirinkite mygtuką“. Pasirinkus neigiamą atsakymą grįžtama į mygtukų redagavimo langą.
8.7.5.1	Pasirinkto lango redagavimo lange mygtukų redagavimo skiltyje nepasirinkus jokio mygtuko iš sąrašo paspaudžiamas mygtukas „Atnaujinti mygtuką“.	Pranešimas „Pasirinkite mygtuką“
8.7.6	Mygtukų sąraše pelės paspaudimu pažymimas bet kuris mygtukas.	Pažymėto mygtuko duomenys parodomi dešiniau esančioje formoje.
8.7.6.1	Mygtuko redagavimo formoje ištrinamas pasirinkto mygtuko pavadinimas ir spaudžiamas mygtukas „Atnaujinti mygtuką“.	Pranešimas „Įveskite mygtuko pavadinimą“.
8.7.7	Pasirinkto lango redagavimo lange mygtukų redagavimo skiltyje paspaudžiamas mygtukas „Naujas“.	Dešiniau parodoma naujo mygtuko sukūrimo forma.
8.7.7.1	Naujo mygtuko sukūrimo formoje neįvedami jokie duomenys ir paspaudžiamas mygtukas „Įtraukti mygtuką“.	Pranešimas „Įveskite mygtuko pavadinimą“.
8.7.7.2	Naujo mygtuko sukūrimo formoje įvedamas pavadinimas ir paspaudžiamas mygtukas „Įtraukti	Sukuriamas mygtukas. Jo pavadinimas atsiranda mygtukų sąraše.

	mygtuką“.	
8.7.8	Pasirinkto lango redagavimo lange mygtukų redagavimo skiltyje pasirenkamas mygtukas iš sąrašo ir paspaudžiamas mygtukas „Ištrinti“.	Pranešimas „Ar tikrai norite ištrinti mygtuką?“. Pasirinkus teigiamą atsakymą, mygtukas ištrinamas. Jo pavadinimas neberodomas mygtukų sąraše. Pasirinkus neigiamą atsakymą grįžtama į mygtukų redagavimo langą.
9. Programinio kodo generavimas		
9.1	Pasirenkama funkcija „Generuoti programinį kodą“.	Pranešimas „Programinis kodas sugeneruotas ir išsaugotas „source“ direktorijoje“.
10. Peržiūra		
10.1	Pasirenkama funkcija „Peržiūrėti aplikacijos grafinę sąsają“	Parodoma sugeneruota projekto aplikacijos grafinė sąsaja.
10.2	Pasirenkama funkcija „Peržiūrėti internetinę grafinę sąsają“	Parodoma sugeneruota projekto internetinė grafinė sąsaja.
11. Pagalba		
11.1	Pasirenkama funkcija „Apie“	Parodoma trumpa informacija apie grafinės vartotojo sąsajos generavimo įrankį.

6.7 Sukurto grafinės sąsajos generavimo algoritmo realizacijos apibendrinimas

Sukurtas metodas remiasi trimis UML diagramomis, sudaromomis kiekvienai modeliuojamai sistemai – panaudojimo atvejų, klasių bei veiklos. Panaudojimo atvejų diagramos duomenys naudojami sugeneruoti sistemos langus, klasių – languose atvaizduoti grafinius elementus pagal jų tipus, o veiklos diagramos duomenys – langų navigacijai sukurti.

Diagramos, sudarytos naudojantis *MagicDraw UML* įrankiu išsaugomos projekto faile ir įkeliamos į grafinės vartotojos sąsajos generavimo įrankį.

Diagramų elementams, kurie bus naudojami grafinei sąsajai generuoti išskyrimui naudojami specialiai sukurto *MagicDraw UML* profilio *GUI* stereotipai. Panaudojimo atvejus žymi *Window*, klases - *WindowAttribute*, o veiklas - *Button* stereotipai.

Naudojantis sąsajos generavimo funkcija grafinė vartotojo sąsaja sugeneruojama tiek aplikacijai, tiek internetinei sąsajai ir gali būti iš karto peržiūrima.

Vartotojo sąsajos generavimo algoritmas realizuotas *Java* programavimo kalba kaip atskira programa, kuri naudoja informaciją, gautą iš *MagicDraw UML* įrankiu sukurto XML projekto failo.

Diagramų informacijai nuskaityti naudojamas SAX (angl. *Simple API for XML*) – įvykiais grindžiamas XML dokumento nuskaitymas. Baigus failo nuskaitymą, atliekamas kiekvieno elemento filtravimas pagal stereotipus ir grąžinami tik tie elementai, kuriems buvo priskirti atitinkami stereotipai. Sukuriami vartotojo sąsajos dialogai, atitinkantys nuskaitytus grafinės sąsajos langus. Į kiekvieną dialogą yra sukeliami lango elementai tikrinant jo tipą. Atvaizdavus grafinius elementus sąsajos lange, įkeliami mygtukai.

Kadangi sąsajos langai generuojami šabloniškai, pavyzdžiui, visuose languose pagal numatytas savybes atvaizduojama tiek forma, tiek duomenų sąrašas, įrankio pagalba galima keisti ne tik pačių langų duomenis, elementų atvaizdavimą, bei sukurti juose naujus elementus, mygtukus. Įrankio pagalba automatizuotai galima kurti ir naujus langus, o naudojantis mygtukų įtraukimo funkcija sukurti navigaciją tarp jų.

7. EKSPERIMENTINIS SISTEMOS TYRIMAS

7.1 Įvedami į grafinės sąsajos generavimo įrankį duomenys

Mokyklos informacinei sistemai sukurti sudaromos panaudojimo atvejų, klasių bei veiklos diagramos.

Panaudojimo atvejų diagrama pavaizduota 89 paveiksle. Pagrindiniai panaudojimo atvejai: prisijungti, administruoti klasę, administruoti mokinį, administruoti mokomąjį dalyką, administruoti pažymį, administruoti būrelį, informuoti tėvus bei administruoti sistemos vartotoją. Kadangi sistema skirta informacijos administravimui, svarbiausi atliekami veiksmai su informacija bus jos įvedimas, redagavimas bei šalinimas.

Kadangi grafinės vartotojo sąsajos generavimo įrankis kuria sistemos langus neatsižvelgdamas į pačios sistemos vartotojų teises ir jiems leidžiamus atlikti veiksmus su informacija, panaudojimo atvejų diagramoje nėra skiriamas dėmesys aktorių išskyrimui ir jų susiejimui su skirtingais panaudojimo atvejais. Šiuo atveju, sukurtas vienas aktorius – vartotojas, kuris gali atlikti visus veiksmus su informacija.

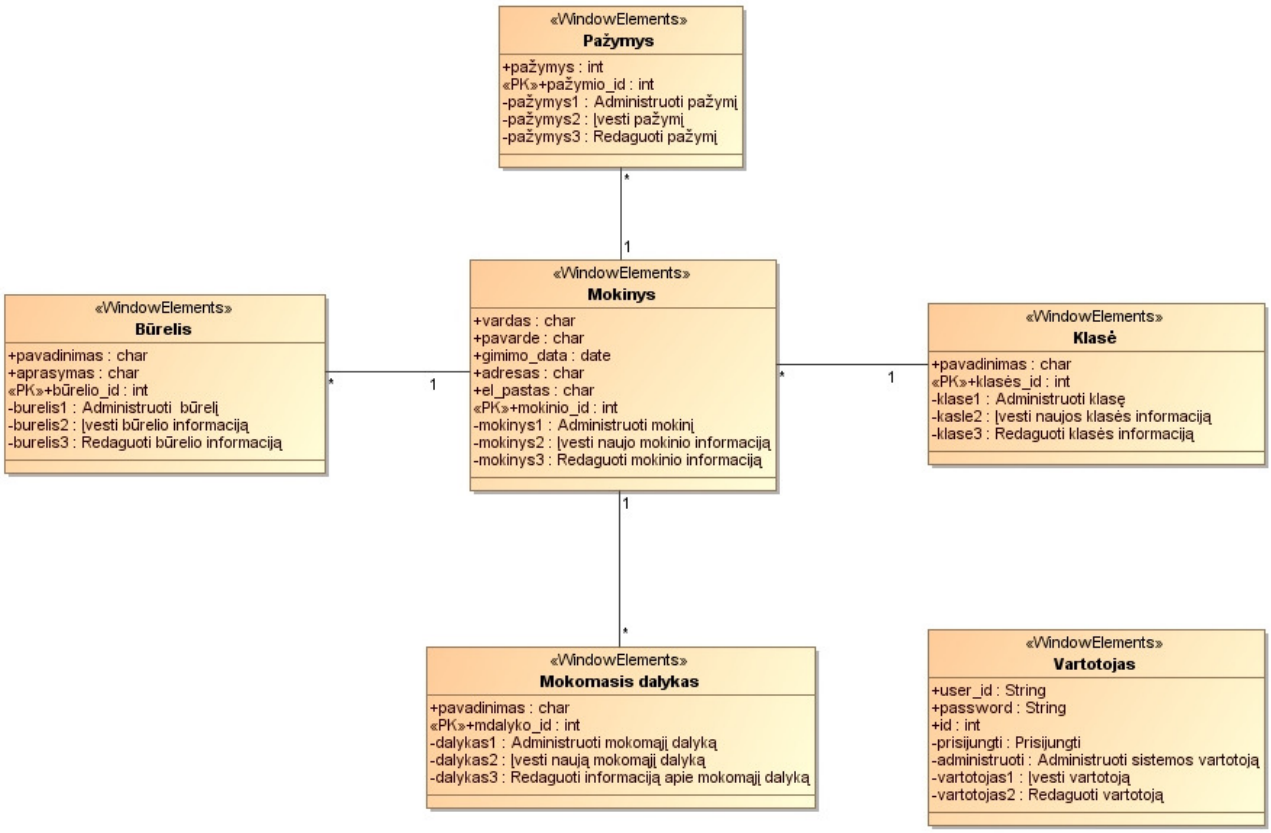
Grafinės vartotojo sąsajos generavimo įrankyje kiekvienas panaudojimo atvejis, pažymėtas *Window* stereotipu iš specialiai įrankiui sukurto *MagicDraw UML GUI* profilio, atitinka sąsajos langą. 89 paveiksle matoma, kad tie panaudojimo atvejai, kurie Mokyklos informacijos sistemos sąsajoje atitiks sistemos langus, yra pažymėti minėtu stereotipu.

89 pav. Pavyzdinės kuriamos informacinės sistemos mokyklai panaudojimo atvejų diagrama

Grafinės vartotojo sąsajos generavimui taip pat reikia sudaryti klasių diagramą. Kiekvienos klasės, pažymėtos *WindowElements* stereotipu iš anksčiau minėto *GUI* profilio, atributai grafinės sąsajos generavimo įrankyje atitinka sąsajos lango elementus. Kadangi šiuo atveju visos klasės yra susietos su kuriais nors panaudojimo atvejais Panaudojimo atveju diagramoje ir yra skirtos informacijai kaupti, visos jos pažymėtos minėtu stereotipu.

Kiekviena klasė, kurios atributai bus atvaizduojami grafinėje sąsajoje, yra susiejama su tam tikru panaudojimo atveju. Klasėje reikia sukurti specifinius atributus su susiejamo panaudojimo atvejo tipu. Pavyzdžiui, klasę „Pažymys“ norint susieti su Pažymio administravimo, įvedimo bei redagavimo sąsajos langais (t.y. panaudojimo atvejais) sukuriami trys atributai su atitinkamais panaudojimo atvejo tipais.

Mokyklos informacijos sistemos klasių diagrama pavaizduota 90 paveiksle.

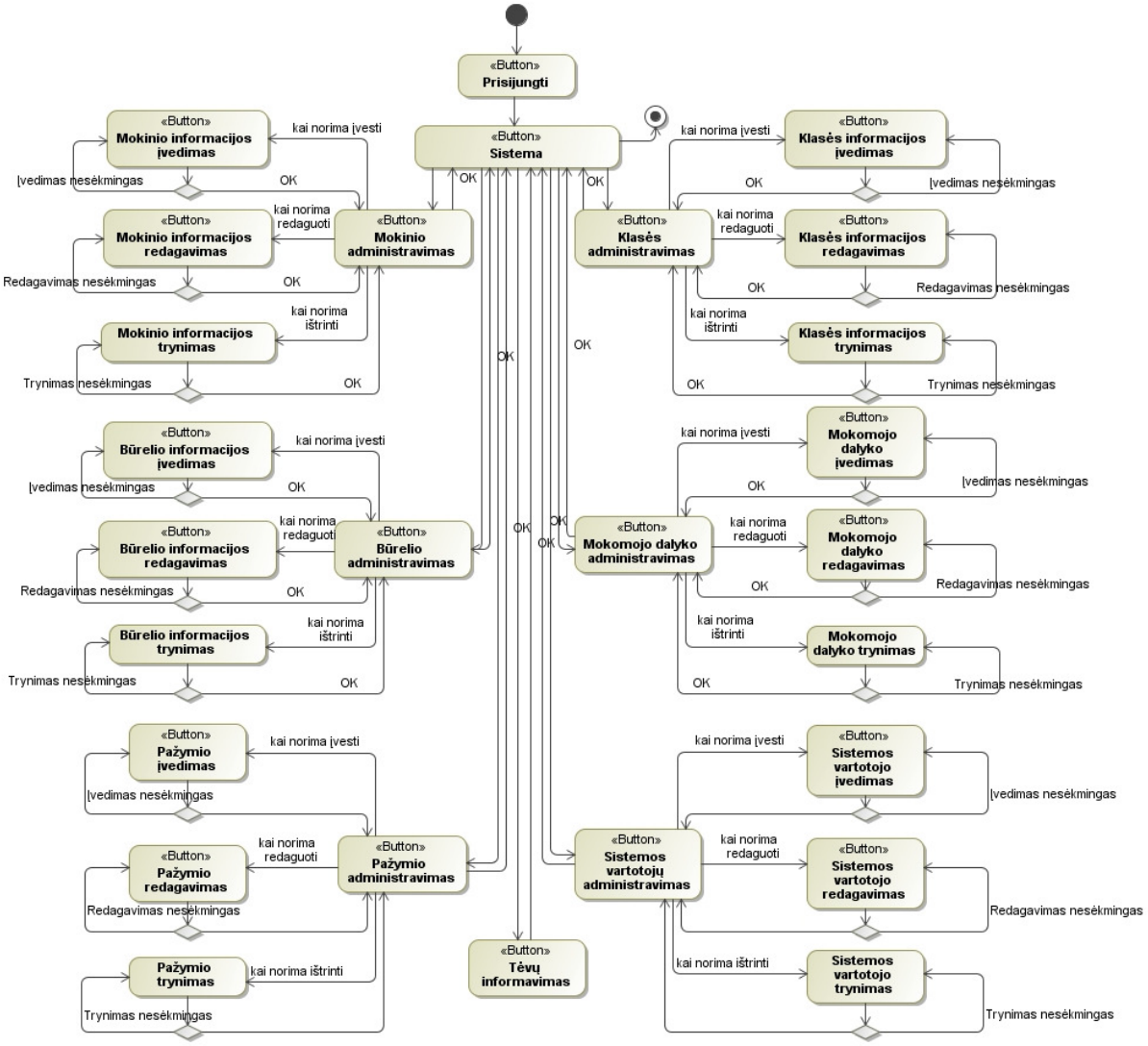


90 pav. Mokyklos informacinės sistemos klasių diagrama

Kuriamai sistemai sudaroma veiklos diagrama, kuri nurodo navigaciją tarp sistemos langų.

Veiklos diagramoje veiklos, kurios atspindi navigaciją tarp sąsajos langų pažymimos stereotipu *Button*. Taip pat veiklos turi būti susietos su konkrečiais panaudojimo atvejais iš jau sudarytos Panaudojimo atvejų diagramos.

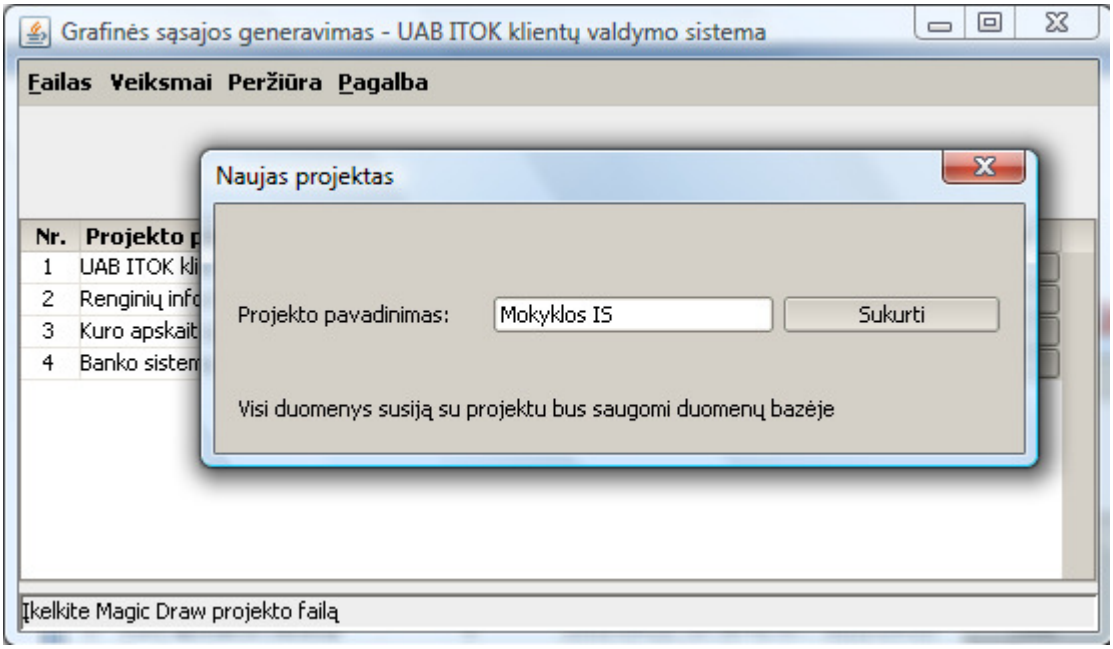
Mokyklos informacinės sistemos veiklos diagrama pavaizduota 91 paveiksle. Darbas su sistema pradedamas prisijungimu. Suvedus prisijungimo duomenis vartotojas mygtukų pagalba galės pasiekti visus informacijos administravimo langus: mokinio, klasės, pažymio, mokomojo dalyko, būrelio, sistemos vartotojo. Iš jų – į duomenų įvedimo bei redagavimo langus. Numatoma, kad informacijos trynimo funkcija bus atliekama duomenų administravimo lange esančiame sąraše. Korektiškai įvedus ar pakeitus informaciją, grįžtama į tos skilties administravimo langą, padarius klaidą – grįžtama į redagavimo langą.



91 pav. Mokyklos informacinės sistemos veiklos diagrama

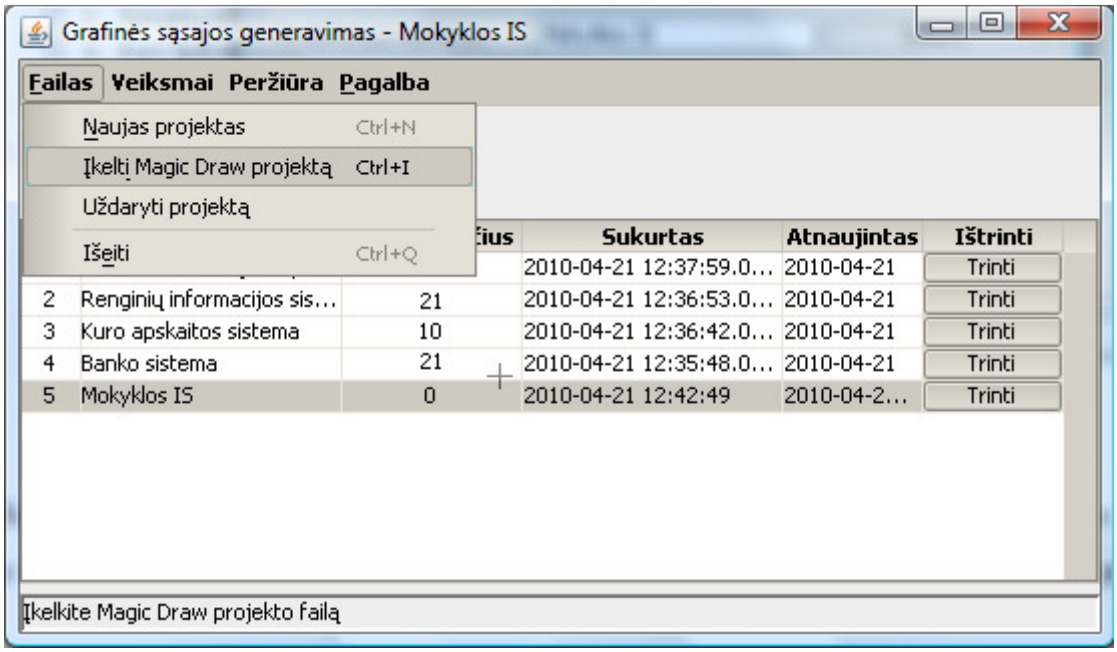
7.2 Automatinis grafinės sąsajos generavimas

Sudarius panaudojimo atvejų, klasių ir veiklos diagramas kuriamai Mokyklos informacijos sistemai, apie kurią jau trumpai užsiminta algoritmo aprašyme, *MagicDraw UML* pakete išsaugomas projektas. Grafinės sąsajos generavimo įrankyje pirmiausia sukuriamas naujas projektas naudojantis *Failas -> Naujas projektas* funkcija. Įvedamas „Mokyklos IS“ pavadinimas ir išsaugomi duomenys (92 pav.).



92 pav. Naujo projekto sukūrimo langas

Sukurto naujo projekto duomenys matomi pagrindiniame grafinės vartotojos sąsajos generavimo įrankio lange. Dukart paspaudus ant projekto pavadinimo, pradedamas darbas su pasirinktu projektu.

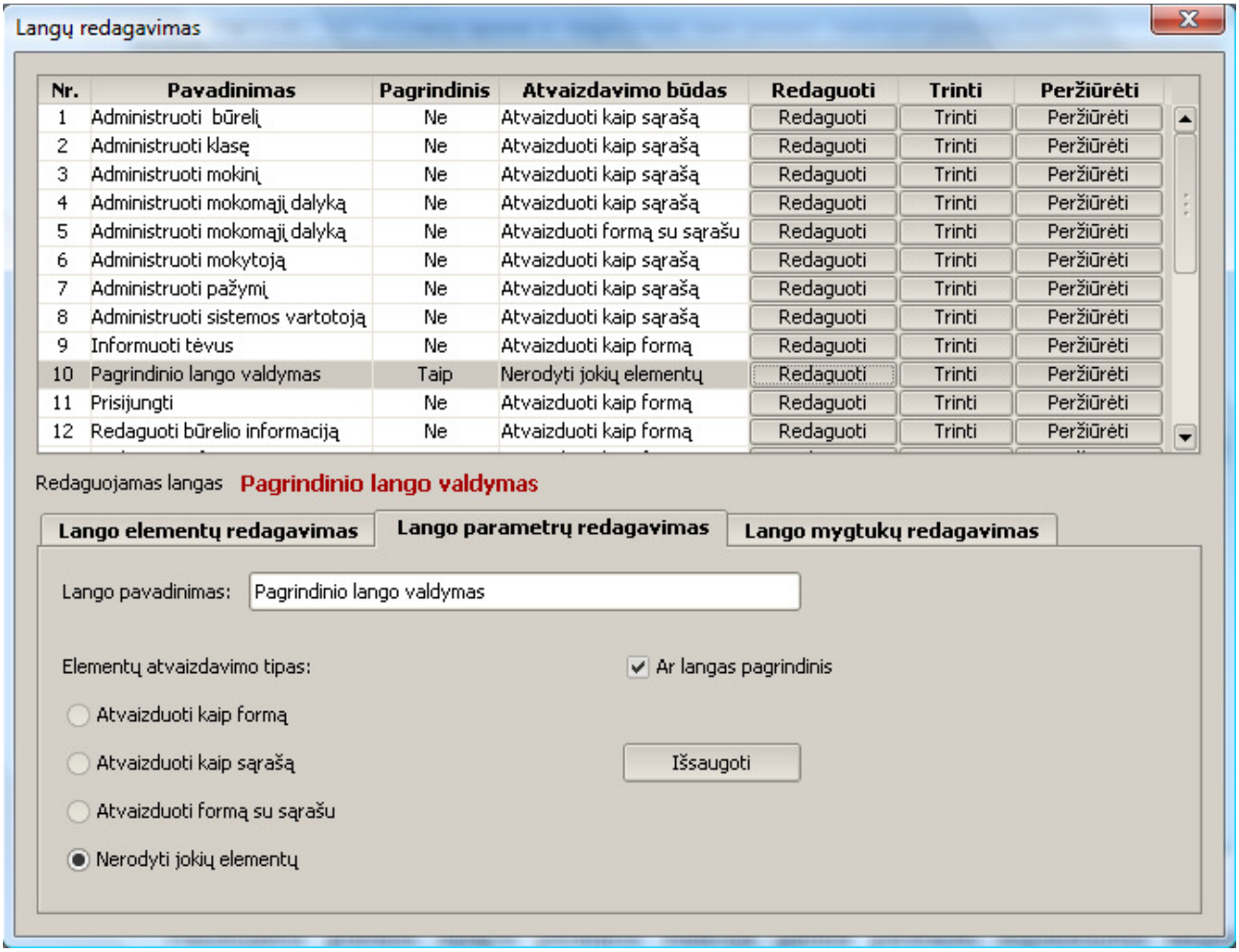


93 pav. Projektų sąrašas ir *MagicDraw* failo įkėlimo funkcijos pasirinkimas

Atsidariusiame lange pasirenkamas kompiuteryje esantis *MagicDraw UML* XML projekto failas. Apie jo duomenų išsaugojimą duomenų bazėje vartotojui praneša žinutė. Norint sugeneruoti grafinę vartotojo sąsają su įkeltomis diagramomis pasirenkama *Veiksmai -> Generuoti* funkcija. Sėkmingai atlikus generavimą, galima peržiūrėti tiek aplikacijos grafinę vartotojo sąsają, tiek pritaikytą internetinei sąsajai.

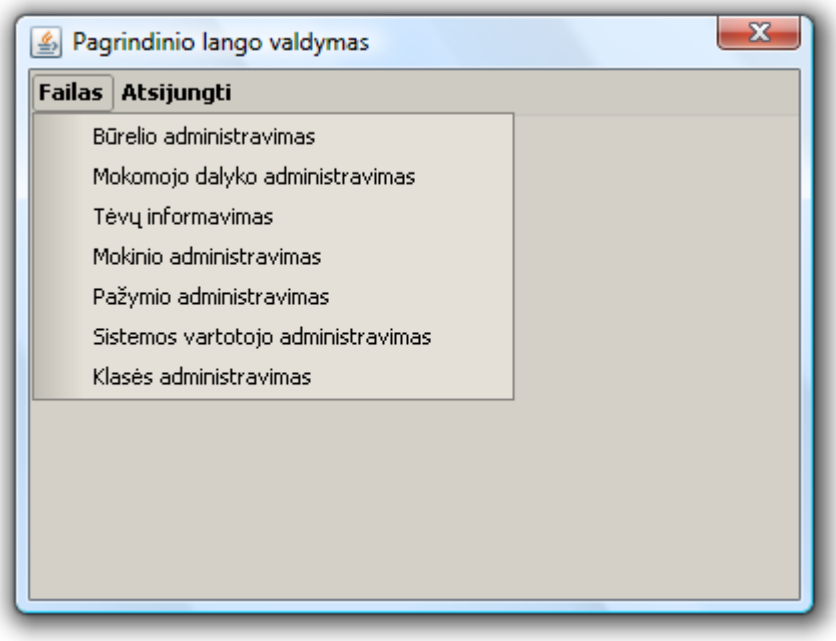
7.2.1 Aplikacijos grafinės vartotojo sąsajos peržiūra ir redagavimas

Kadangi Mokyklos informacijos sistema turi ne tik informacijos administravimo langus, bet ir pagrindinį, skirtą efektyviai navigacijai tarp kitų langų, grafinės sąsajos generavimo įrankyje reikia išskirti šį langą ir pažymėti jį kaip pagrindinį. Tai atliekama Langų redagavimo (*Veiksmai -> Langų redagavimas*) sąraše pažymėjus pagrindinį langą (šiuo atveju, Pagrindinio lango valdymas), jo parametrų redagavimo režime varnele pažymėjus parinktį „Ar langas pagrindinis“ (94 pav.) bei išsaugojus pakeitimus.



94 pav. Pagrindinio lango parametrų redagavimo režimas

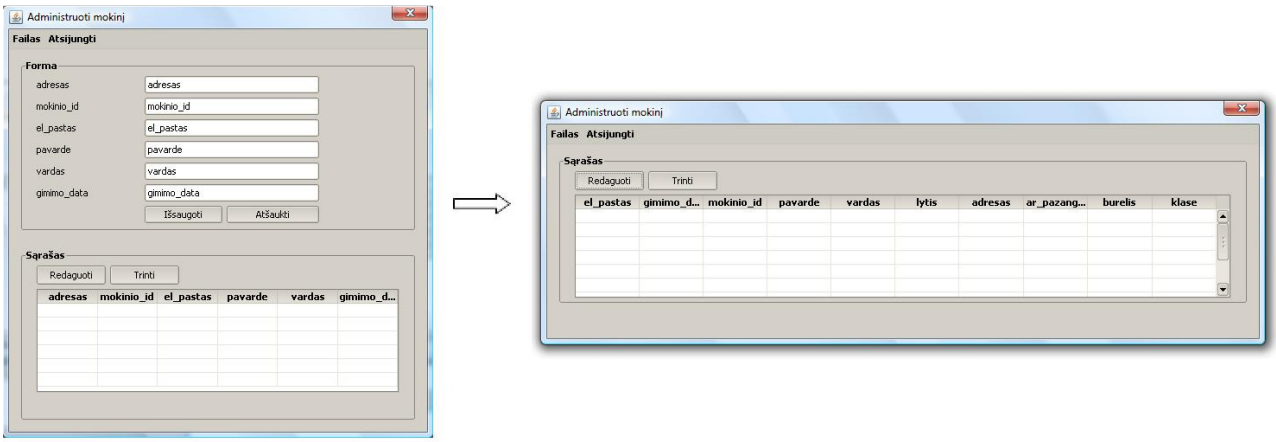
Naudojantis grafinės sąsajos peržiūros funkcija galima peržiūrėti sugeneruotus langus. Pagrindinio lango vaizdas matomas 95 paveiksle. Pagrindiniame lange sugeneruotas meniu su nuorodomis į informacijos administravimo langus: klasės, mokinio sistemos vartotojo, mokomojo dalyko, pažymio ir būrelio. Taip pat yra nuoroda į Tėvų informavimo langą. Norint išjungti sugeneruotą grafinę sąsają reikia pasirinkti *Atsijungti -> Baigti darbą* funkciją meniu juostoje.



95 pav. Pagrindinis Mokyklos informacijos sistemos langas ir nuorodos į informacijos administravimo langus

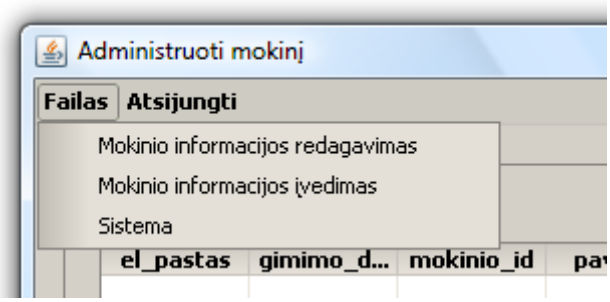
Pasirinkus, pavyzdžiui, Mokinio administravimo funkciją, atidaromas langas su klasių diagramoje sumodeliuotos klasės „Mokinys“ atributais, transformuotais į įvedimo laukelius, bei sąrašu, kuriame būtų matomi įvesti duomenys (kairysis 96 pav.). Sąraše esančią informaciją būtų galima redaguoti bei trinti mygtukų pagalba.

Generuojant langus sistema kiekvienam langui sukuria tiek formas, tiek sąrašus su susietos klasės duomenimis. Kadangi „Mokinio administravimo“ lange numatoma administruoti tik mokinių sąrašą redaguojant arba trinant duomenis, o naujų duomenų įvedimas bus vykdomas „Naujo mokinio įvedimo“ lange, naudojantis šio lango redagavimo funkcija formos ir sąrašo rodymo režimas pakeičiamas į sąrašo rodymo režimą. Po redagavimo „Mokinių administravimo“ langas atrodys taip, kaip pavaizduota 96 paveikslo dešinėje.



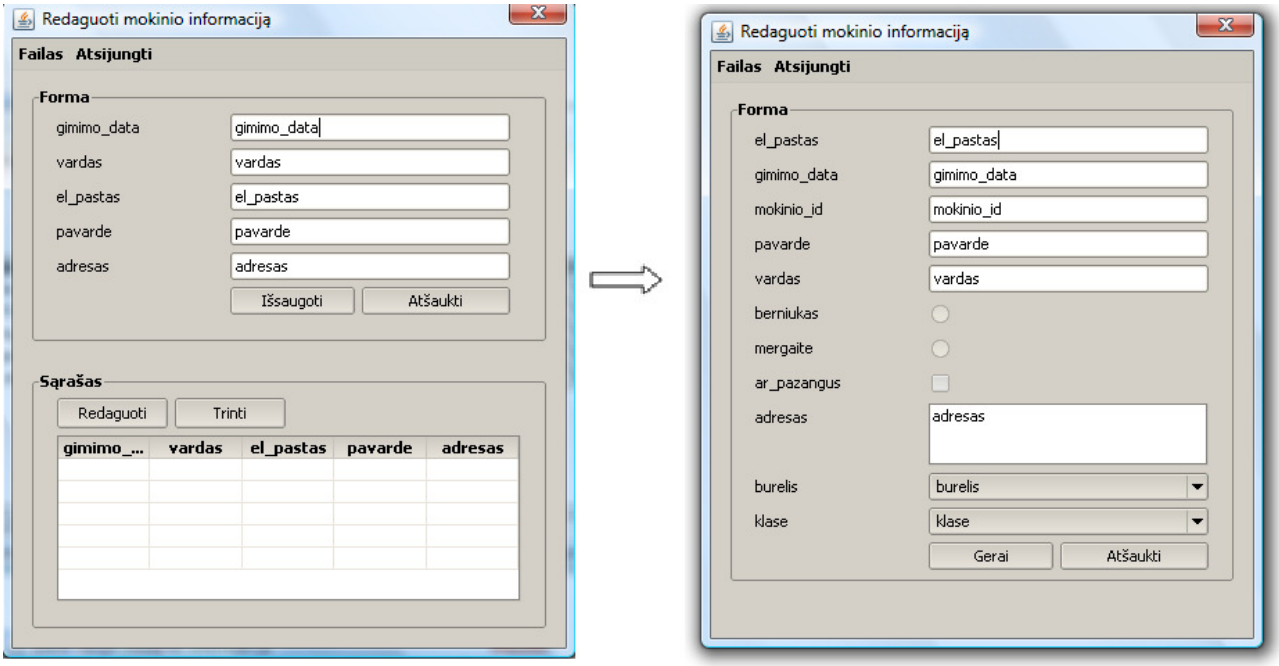
96 pav. Mokinio administravimo langas prieš ir po redagavimo

Sugeneruotoje meniu juostoje skiltyje „Failas“ yra nuorodos į su šiuo langu susijusius langus, t.y. naujo mokinio įvedimo langą, mokinio redagavimo langą (97 pav.). Taip pat, pasirinkus „Sistema“ meniu punktą, galima grįžti į pagrindinį sistemos langą.



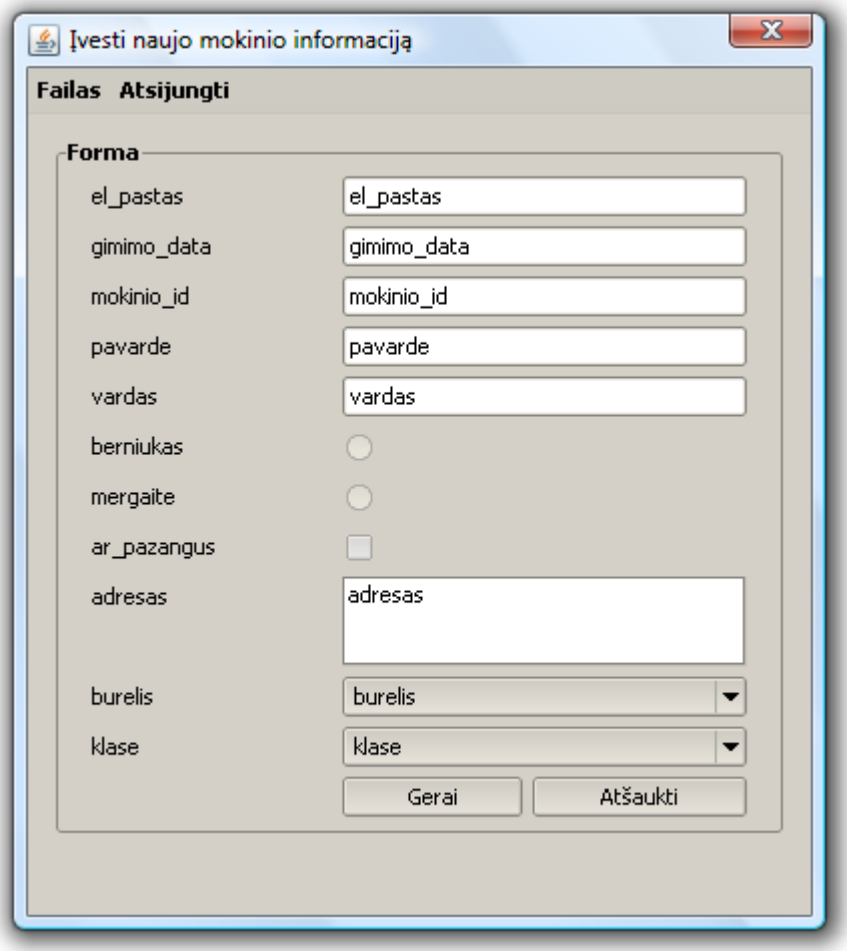
97 pav. Mokinio administravimo lango meniu parinktys

„Mokinio redagavimo“ langas prieš ir po redagavimo matomas 98 paveiksle. Sistema formoje atvaizduoja visus su tuo langu susietos klasės atributus transformuodama juos į grafinius elementus. Pasirinkto lango redagavimo režime lango elementų redagavimo skiltyje pažymimas norimas pašalinti elementas ir paspaudžiamas mygtukas „Trinti“. Šiame lange galima kurti ir naujus reikiamus formoje elementus, pavyzdžiui, elementą „ar_pazangus“, kuris nurodytų, ar konkretus mokinys mokosi pažangiai, taip pat „klasė“, kuriame iš sąrašo būtų pasirenkama klasė, kurioje mokinys mokosi.



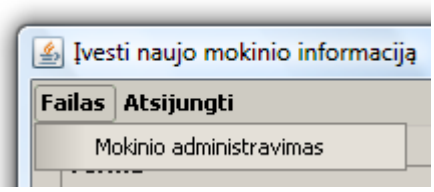
98 pav. Mokinio redagavimo langas prieš ir po redagavimo

Atitinkami veiksmai atlikti su „Naujo mokinio įvedimo“ lango elementais. Šio lango atvaizdas matomas 99 paveiksle.



99 pav. Naujo mokinio įvedimo langas

Iš naujo mokinio įvedimo ir redagavimo langų meniu pagalba galima grįžti į „Mokinio administravimo“ langą (100 pav.).



100 pav. Naujo mokinio įvedimo lango meniu parinktis

Kiti Mokyklos informacijos sistemos informacijos įvedimo ar redagavimo langai gali būti redaguojami tokiu pačiu būdu kaip ir mokinio informacijos administravimo langai.

Priede nr. 12 pateikti visi sugeneruoti pavyzdinės Mokyklos informacijos sistemos aplikacijos grafinės sąsajos langai.

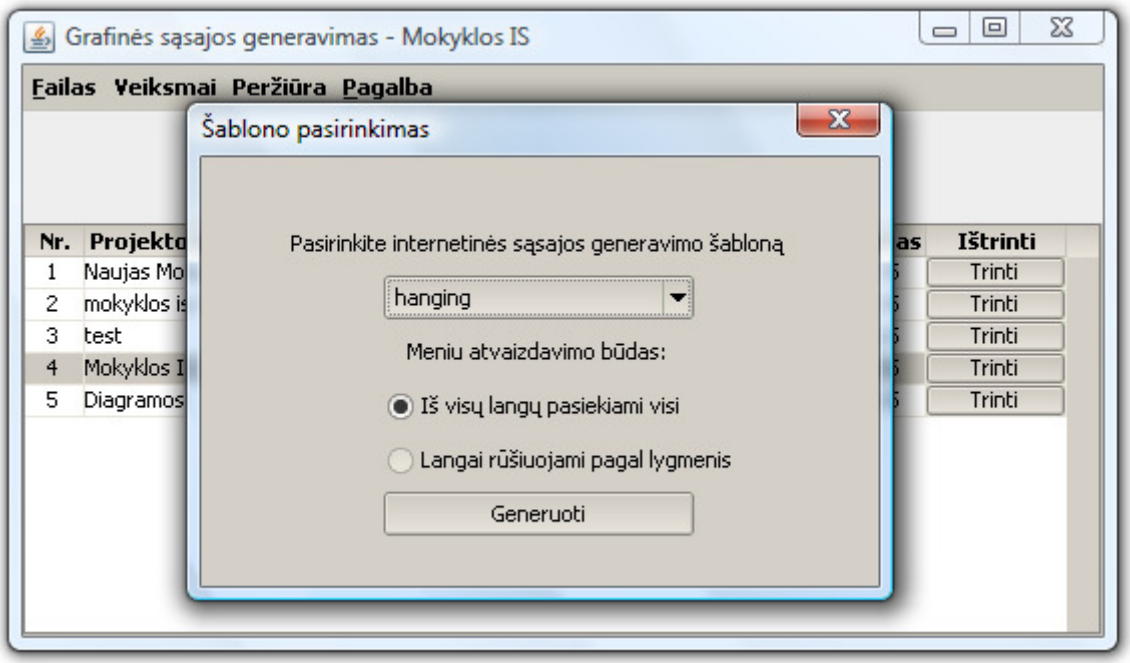
7.2.2 Grafinės sąsajos peržiūra internetinei sąsajai

Sugeneravus grafinę sąsają ją galima peržiūrėti ir pritaikytą internetinei sąsajai. Tai atliekama grafinės vartotojo sąsajos generavimo įrankyje pasirinkus funkciją *Peržiūra* ->

Peržiūrėti grafinę sąsają internetinei sąsajai. Šiai sąsajai generuoti naudojamos tos pačios diagramos ir tie patys nustatymai langams.

Pasirinkus peržiūros funkciją, sistema parodo langą (101 pav.), kuriame reikia pasirinkti internetinės sąsajos šabloną. Naudojant pasirinktą šabloną bus atvaizduojama grafinė sąsaja. Šablone yra sukurti tik blokai, jų išdėstymas, parinktos spalvos. Jokie grafiniai elementai iš anksto nėra įkeliami.

Taip pat reikia pasirinkti meniu atvaizdavimo būdą. Meniu atvaizdavimo būdai yra du: kai iš visų sąsajos langų meniu pagalba bus galima pasiekti visus sistemos langus ir kai meniu bus atvaizduojamas taip, kaip aplikacijos grafinėje sąsajoje, t.y. lygmenimis.



101 pav. Internetinės grafinės sąsajos peržiūros paleidimas

Internetinėje grafinėje sąsajoje atvaizduojami tie patys langai, jų elementai ir mygtukai kaip ir aplikacijos grafinėje sąsajoje. Visus sugeneruotus internetinės grafinės sąsajos langus pavyzdinei Mokyklos informacijos sistemai galima peržiūrėti Priede Nr. 2.

7.3 Kokybės kriterijų įvertinimas

7.3.1 Grafinės vartotojo sąsajos sugeneravime naudojami grafiniai elementai

Pagal grafinių elementų sąrašą, pateiktą 2.5.1 - 2.5.2 skyriuose sudarytos lentelės su grafiniais elementais, kurie naudojami grafinės sąsajos prototipo generavimui. Didžiausias dėmesys kreipiamas į grafinius elementus, kurie skirti darbui su informacija ar jos atvaizdavimu, o ne šriftų formatavimu, elementų sugrupavimu, paveikslėlių įkėlimu, karkasų sudarymu ir pan., todėl tokie elementai nėra įtraukti į sąsajos generavimą.

52 lentelėje pateiktas *Java* programavimo kalbos grafiniai elementai, kurie naudojami aplikacijos grafinei sąsajai sugeneruoti.

52 lentelė. Java programavimo kalbos grafiniai elementai, kurie naudojami aplikacijos grafinei sąsajai sugeneruoti

Tipas	**Elementai**
Formos	*JCheckBox, JComboBox, JRadioButton, JTextArea, JTextField,*
Mygtukai	*JButton*
Grupavimas ir maketavimas	*JPanel, JSeparator*
Langas ir dialogai	*JFrame, JWindow, JDialog,*
Tekstai	*JLabel*
Lentelė	*JList, JTable*
Meniu	*JMenu, JMenuBar, JMenuItem*
Slinkties juosta	*JScrollBar*

53 lentelėje pateikti naudojami grafiniai elementai internetinei sąsajai sugeneruoti.

53 lentelė. Naudojamų HTML grafinės sąsajos elementų sąrašas

Tipas	**Elementai**
Meta informacija	*html, head, title, meta, style*
Pagrindinė dalis	*body*
Pavadinimai	*h1, h2*
Sąrašai	*ul, li*
Teksto struktūra	*p, div*
Lentelė	*table, thead, tbody, tr, th, td*
Nuorodos	*a*
Formos	*form, label, input, select, textarea, button*
Mygtukai	*button, input type="button"*

Iš lentelėse pateiktų elementų matoma, kad grafinės vartotojo sąsajos generavime naudojami pagrindiniai grafiniai elementai, t.y. langai (angl. *window*), slinkties juostos (angl. *scroll bar*), įvairaus dydžio tekstiniai laukai (angl. *text field*, *text area*), mygtukai (angl. *button*), pasirinkimo grafiniai elementai (angl. *check box*, *radio button*), pavadinimai (angl. *label, title, h1, h2*), meniu (angl. *menu bar, menu item*), lentelės ir jų elementai (angl. *table*, *thead, tbody, tr, th, td*), formos (angl. *form*). Internetinėje grafinėje sąsajoje sukuriamos ir

nuorodos (angl. *a*), priskiriamas šablono stilius (angl. *style*), kiti standartiniai elementai (angl. *body*, *html*, *head*, *meta* ir kt.)

7.3.2 Grafinės vartotojo sąsajos generavimo laiko įvertinimas

Grafinė vartotojo sąsajos generavimo programa yra skirta sąsajos generavimui tuo atveju, kai projektavimo metu sudaroma sistemos specifikacija, taigi kitu atveju ji netinka. Bus atliktas bandymas, kurio metu sistemos specifikacija sudaroma *MagicDraw UML* įrankiu, o grafinės sąsajos prototipas nubraižomas ranka pasinaudojant Grafinės vartotojo sąsajos diagrama tame pačiame įrankyje. Laiką, kurį užtruksime sudarant grafinę sąsają, palyginsime su laiku, kurį užtruksime grafinei sąsajai sugeneruoti automatiniu būdu ir redaguoti naudojantis grafinės sąsajos generavimo įrankiu. Kiekvieno bandymo varianto duomenys (panaudojimo atvejų, klasių ir veiklų diagramos), ranka bei automatiniu būdu sudarytos grafinės sąsajos pateiktos Priede nr. 2. Internetinės grafinės vartotojo sąsajos paveikslai pateikti Priede nr. 3.

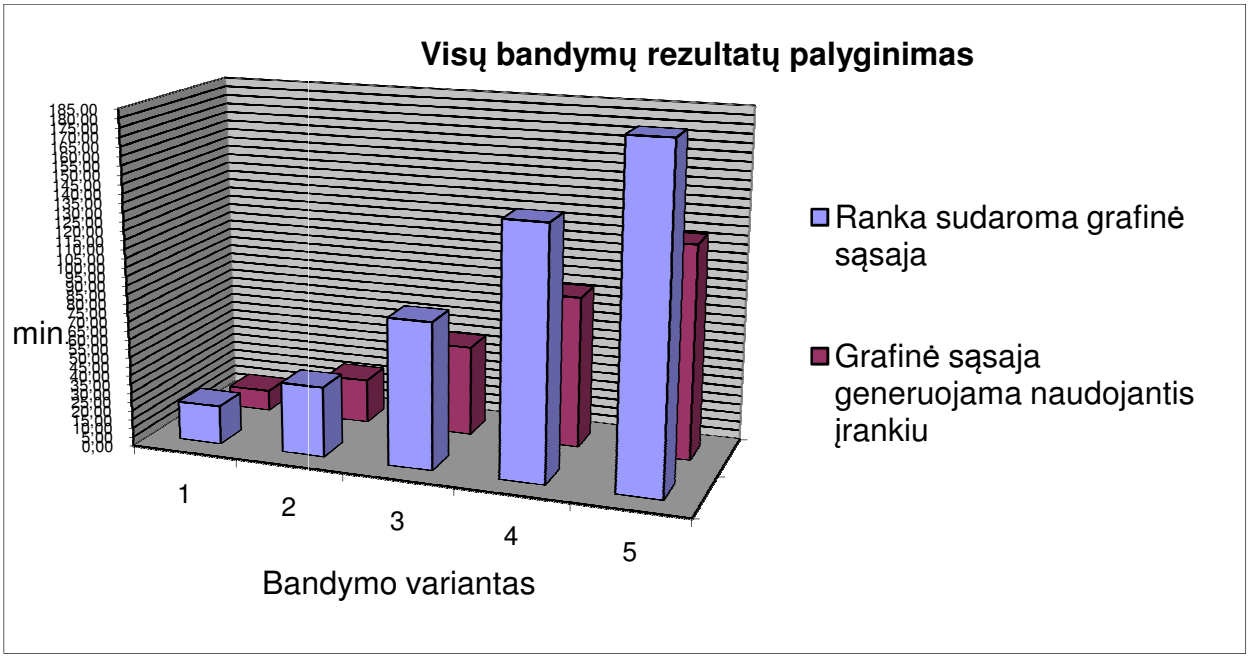
Laikas, kuris buvo užtruktas aprašytiems darbams, pateiktas 54 lentelėje.

54 lentelė. Grafinės vartotojo sąsajos sudarymo laikų statistika

		Ranka sudaroma grafinė sąsaja		**Grafinė sąsaja generuojama naudojantis įrankiu**			
Bandymo varianto nr.	**UML diagramų sudarymas, min.**	**Grafinės sąsajos sudarymas naudojantis *MagicDraw* įrankiu, min.**	**Viso, min.**	**UML diagramų paruošimas sąsajos generavimui, min.**	**Grafinės sąsajos sugeneravimas naudojantis įrankiu, min.**	**Grafinės sąsajos redagavimas iki priimtino varianto, min.**	**Viso, min.**
I var. 2 PA 1 klasė 2 veiklos	9.5min.	12 min.	**21.5 min.**	1.5 min.	0.1 min.	0.5 min.	**11.6 min.**
II var. 4 PA 1 klasė 4 veiklos	21 min.	18 min.	**39 min.**	2 min.	0.1 min.	1.5 min.	**24.6 min.**
III var. 9 PA 2 klasės 10 veiklų	43 min.	38 min.	**81 min.**	4 min.	0.1 min.	3 min.	**50.1 min.**
IV var.							

17 PA 4 klasės 18 veiklų	73 min.	65 min.	**138 min.**	5.5 min.	0.1 min.	5.5 min.	**84.1 min.**
V var. 26 PA 6 klasės 27 veiklos	102 min.	82 min.	**184 min.**	7 min.	0.1 min.	9 min.	**118.1 min.**

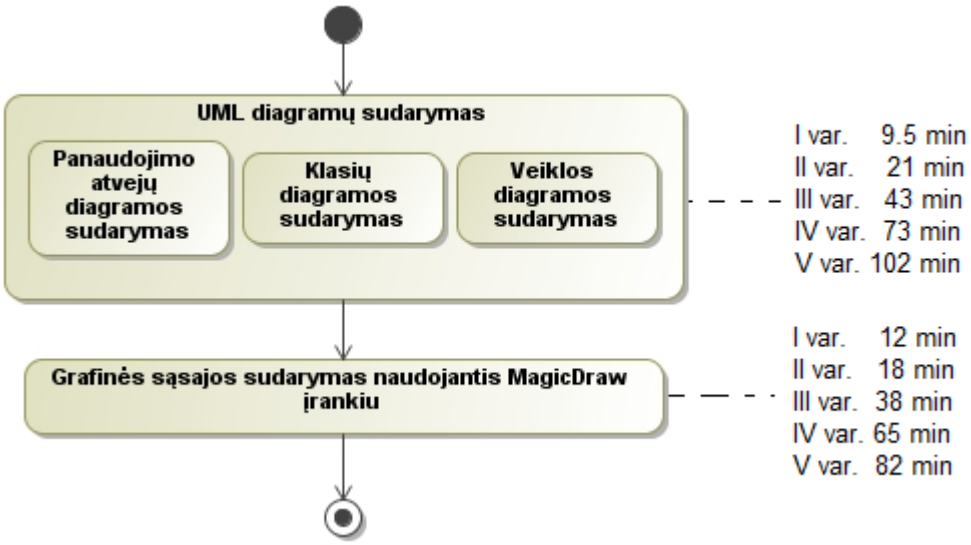
Iš lentelėje pateiktų bandymo rezultatų matoma, kad laikas, kurio reikėjo grafinės sąsajos prototipo sudarymui naudojantis *MagicDraw UML* įrankiu žymiai didesnis nei laikas, skirtas automatiniam grafinės sąsajos generavimui. Laiko pasiskirstymas pateiktas 102 paveiksle.



102 pav. Bandymuose užtrukto laiko pasiskirstymas etapuose

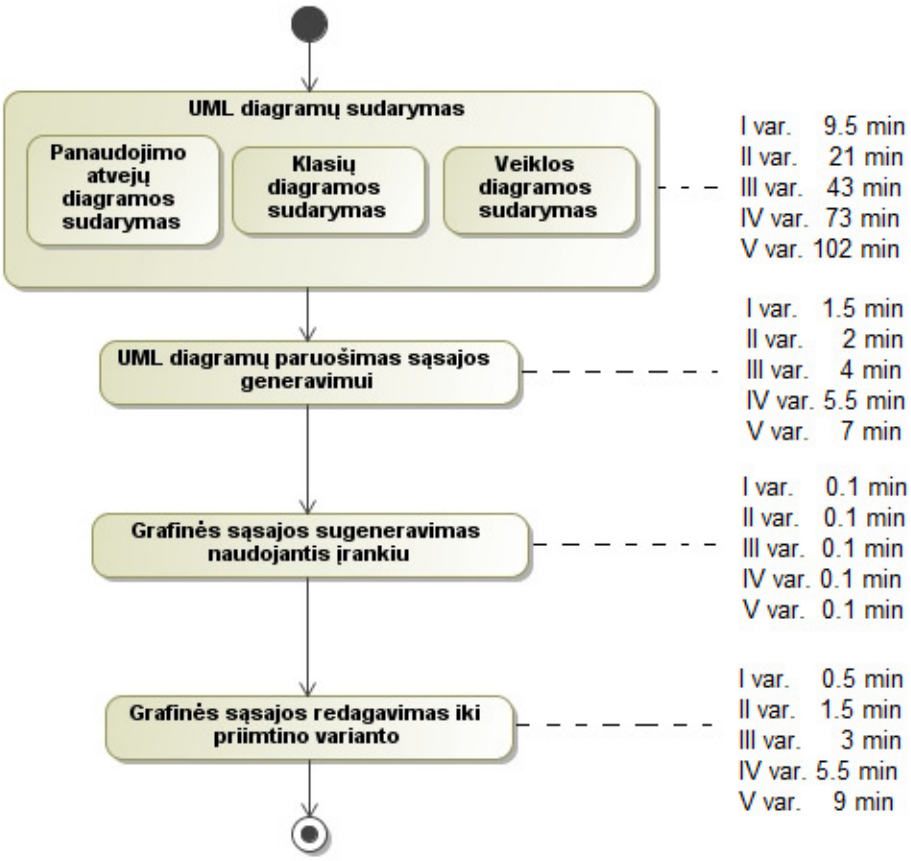
Remiantis rezultatais, taip pat galima teigti, kad naudotis įrankiu laiko atžvilgiu verta visada, net ir sudarant mažos apimties grafines sąsajas. Pažymėtina, kad kuriant didesnių sistemų grafines sąsajas laiko sutaupoma daugiau dėl pačio darbo apimties.

Grafinės vartotojo sąsajos sudarymo ranka etapai išskirstyti į UML diagramų sudarymą ir grafinės sąsajos prototipo sukūrimą naudojantis *MagicDraw UML* įrankiu. Šis procesas su rezultatais, gautais generuojant keleto sistemų grafinės sąsajos prototipus pavaizduotas 103 paveiksle.



103 pav. Grafinės vartotojo sąsajos sudarymo ranka procesas ir rezultatai

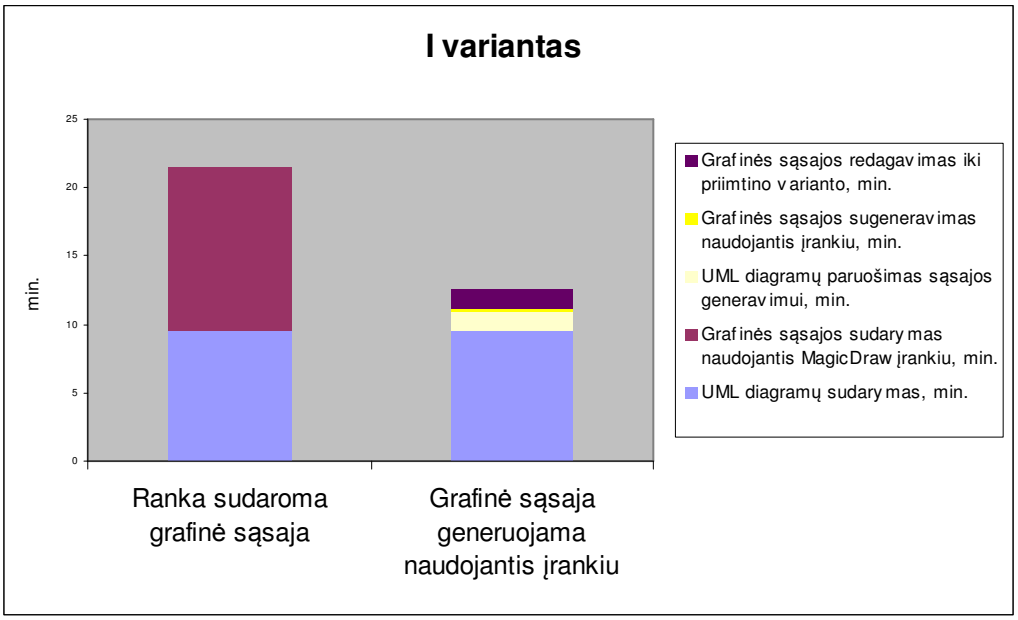
Grafinės vartotojo sąsajos sudarymo naudojantis grafinės sąsajos generavimo įrankiu etapai panašūs į sąsajos sudarymo rankiniu būdu etapus, tačiau šiuo atveju reikia ne tik sudaryti UML diagramas, bet ir jas paruošti darbui su įrankiu, sugeneruoti sąsają bei ją redaguoti, jei automatiškai sugeneruota sąsaja turi trūkumų (pavyzdžiui, trūksta mygtuko) . Šis procesas su rezultatais, gautais generuojant keleto sistemų grafinės sąsajos prototipus pavaizduotas 104 paveiksle.



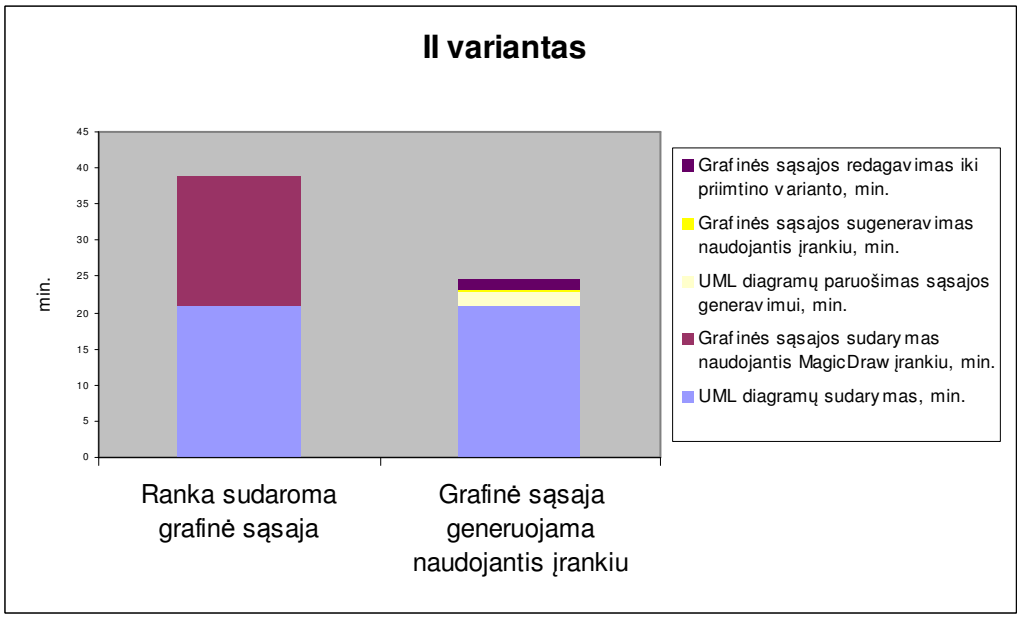
104 pav. Grafinės vartotojo sąsajos sudarymo naudojantis įrankiu procesas ir rezultatai

Kadangi UML diagramas reikia paruošti prieš įkeliant į sistemą, tam sugaištama papildomo laiko, tačiau akivaizdu, kad jis sutaupomas sugeneruojant grafinę sąsają.

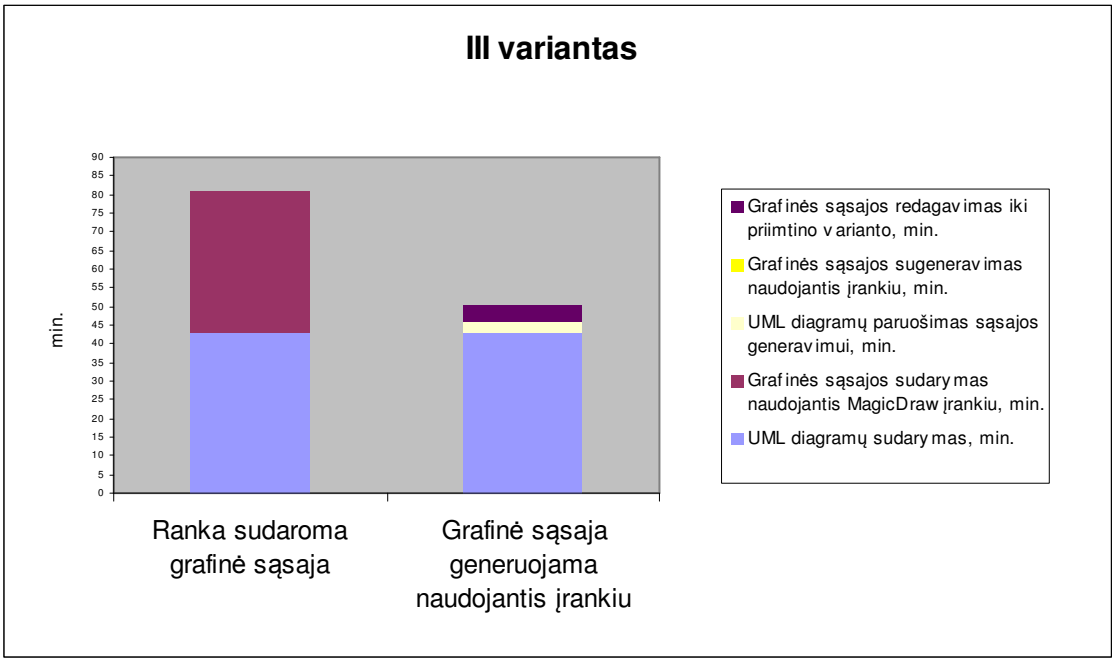
105-109 paveiksluose pateikti laikų pasiskirstymai, kai kiekvienas iš variantų, aprašytų 55 lentelėje buvo atliekamas dviem būdais: sąsaja generuojama naudojant grafinės vartotojo sąsajos generavimo įrankį ir kuriama ranka (sudaroma naudojant *MagicDraw UML* įrankio siūlomas Grafinės vartotojo sąsajos diagramas).



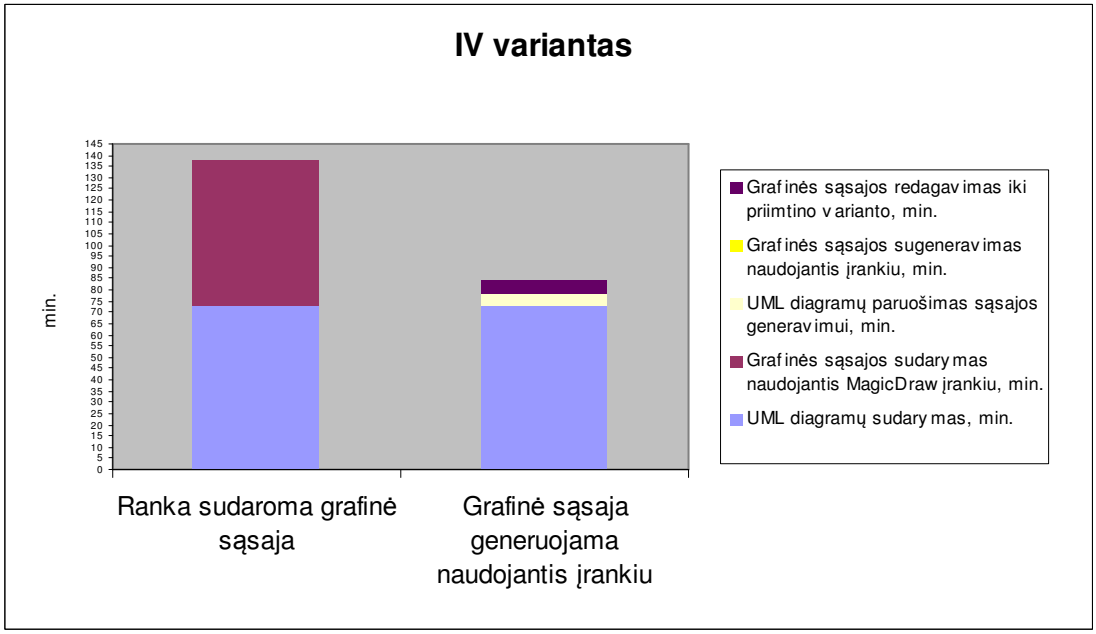
105 pav. I varianto laikų pasiskirstymas



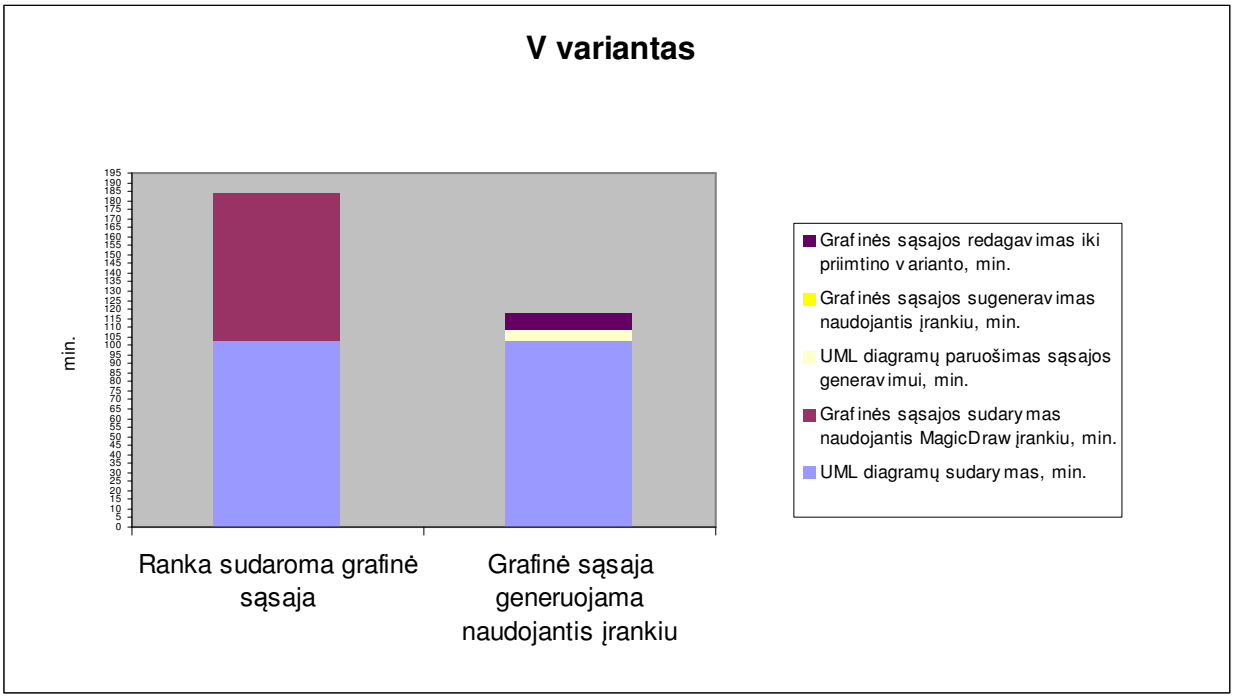
106 pav. II varianto laikų pasiskirstymas



107 pav. III varianto laikų pasiskirstymas



108 pav. IV varianto laikų pasiskirstymas



109 pav. V varianto laikų pasiskirstymas

Be to, sugeneravę grafinę sąsają, gaunami realūs langai su realizuota navigacija, kurios sudarant sąsajos grafinius modelius *MagicDraw UML* įrankyje realizuoti neįmanoma. Grafinės sąsajos generavimo įrankis taip pat pateikia sąsajos prototipo programinį kodą tiek *Java* programavimo kalba, tiek HTML, taigi sąsają toliau tobulinti galima ne tik pačiame įrankyje, bet ir, prireikus, redaguojant kodą sistemos programavimo etape. Šiame etape taip pat nebereikia sąsajos programuoti nuo pradžių, galima naudoti jau sugeneruotą programinį kodą, taigi bendras projekto įvykdymo laikas dar labiau sutrumpėja.

7.3.3 Grafinės vartotojo sąsajos generavimas iš įvairaus sudėtingumo UML diagramų

Grafinės vartotojos sąsajos generavimo įrankis gali būti naudojamas įvairios apimties sąsajoms generuoti. 55 lentelėje pateikti bandymai pagal įvairios apimties panaudojimo atvejų, klasių ir veiklų diagramas. Lentelėje pateikti duomenys iš 7.3.2 skiltyje aprašytų bandymų grafinių sąsajų.

55 lentelė. Bandymų rezultatai pagal įvairios apimties panaudojimo atvejų, klasių ir veiklų diagramas

Nr.	**UML diagramų elementai**			**Grafinė vartotojo sąsaja**		
	Panaudojimo atvejai/iš jų su stereotipu, sk.	**Klasės/atributai, sk.**	**Veiklos/iš jų su stereotipu, sk.**	**Langai, sk.**	**Sukurti elementai iš klasių diagramos, sk.**	**Navigacija**
1.	2/**2**	1/**6**	2/**2**	**2**	**6**	Yra
2.	4/**3**	1/**6**	4/**3**	**3**	**6**	Yra
3.	9/**7**	2/**8**	10/**8**	**7**	**8**	Yra
4.	17/**13**	4/**13**	18/**14**	**13**	**13**	Yra
5.	26/**20**	6/**18**	27/**21**	**20**	**18**	Yra

Iš 55 lentelės, matoma, kad langų skaičius atitinka panaudojimo atvejų su stereotipu *Window* skaičių, visi klasių atributai atvaizduojami languose kaip grafiniai elementai, o pagal veiklos diagramą sukuriama langų navigacija. Į elementų skaičių nebuvo įtraukti automatiškai pagal nutylėjimą sugeneruojami mygtukai, sąrašai ar meniu juosta.

Atliktais bandymais įrodyta, kad pasiektas išsikeltas tikslas – pagreitinti ir palengvinti informacijos sistemų kūrimo procesą, o tai realizuota sukuriant grafinės vartotojo sąsajos prototipo generavimo iš UML diagramų algoritmą ir jį panaudojant sukuriant grafinės sąsajos generavimo įrankį.

7.4 Savybių analizė ir taikymo rekomendacijos

Sukurtas įrankis skirtas generuoti grafinę vartotojo sąsają iš UML diagramų. Norint sugeneruoti grafinę sąsają į įrankį reikia įkelti trijų diagramų informaciją *MagicDraw UML* projekto XML formatu. Reikalingas sąsajai sukurti panaudojimo atvejų, klasių ir veiklos diagramas sistemų projektuotojai bet kuriuo atveju sudaro kurdami sistemos specifikaciją. UML pagalba sumodeliuojami pagrindiniai sistemos procesai, funkcionalumas, numatoma struktūra, klasės, jų atributai, operacijos ir ryšiai. Šie duomenys ir yra naudojami generuojant grafinę vartotojo sąsają.

Įkėlus diagramas į įrankį ir sugeneravus grafinę sąsają, ją galima peržiūrėti dviem būdais: pritaikytą aplikacijai ir internetinei sąsajai.

Grafinę vartotojo sąsają galima generuoti automatiškai, tačiau įrankio pagalba taip pat galima automatizuotai įkelti naujus bei redaguoti jau sukurtus sąsajos langus.

Generuojant grafinę vartotojo sąsają, sugeneruojamas ir jos programinis kodas. Aplikacijai pritaikytos sąsajos kodas išsaugomas *Java* programavimo kalboje, o internetinei sąsajai – HTML kalba.

8. IŠVADOS

1. Vartotojo sąsajos prototipas yra naudingas:
    - reikalavimų analizės ir specifikavimo etape, nes pademonstruoja būsimos sistemos veikimo principus pagal vartotojo pateiktus reikalavimus;
    - realizuojant sistemą taip pat gali būti naudojamas jau sukurtas grafinės sąsajos prototipas.
2. Analizuotuose egzistuojančiuose sprendimuose vartotojo sąsajos generavimas yra sudėtingas ir užima daug laiko aprašant diagramas. Be to, metodai dažniausiai remiasi ne UML diagramomis, o jau su realizacija susietais būsimos sistemos aprašais, o tie kurie naudoja UML diagramas, reikalauja daug papildomos informacijos.
3. Būtų naudinga sukurti vartotojo sąsajos generavimo algoritmą, kuris naudotų minimalų reikalingų diagramų kiekį ir neperkrautų vartotojo reikalavimais įvesti daug papildomos informacijos. Taip pat svarbu, kad papildomos informacijos įvedimas nebūtų labai sudėtingas, priešingu atveju, vartotojui bus paprasčiau kurti vartotojo sąsają rankiniu būdu, negu naudoti pasiūlytą algoritmą.
4. Sistema, generuojanti grafinę vartotojo sąsają iš bet kuriuo atveju projektavimo metu sudaromų UML diagramų sutaupytų laiko, skirto sąsajos prototipo kūrimui, o esant net ir dideliems pakeitimams, sąsają tereikėtų tik redaguoti, o ne sudaryti iš naujo.
5. Siūlomam generavimo metodui būtų reikalinga panaudojimų atvejų diagrama, atvaizduojama į sistemos langus, klasių diagrama, atvaizduojama į grafinės sąsajos elementus, ir veiklos diagrama, aprašanti navigaciją tarp sąsajos langų.
6. Grafinės sąsajos generavimo procese naudojamoms diagramoms ir jų elementams pažymėti šiame darbe sudarytas *MagicDraw UML* įrankio *GUI* profilis, kuriame aprašyti diagramų elementų stereotipai.
7. Kaip įėjimo duomenys algoritmui naudojamas UML metamodelio poaibis, o rezultato metamodelis gali būti dviejų rūšių – aplikacijos arba internetinės grafinės sąsajos.
8. Vartotojo sąsajos generavimo algoritmas realizuotas *Java* programavimo kalba kaip atskira programa, kuri naudoja informaciją, gautą iš *MagicDraw* įrankiu sukurto XML projekto failo. Naudojant šiame darbe sukurtą įrankį gali būti sugeneruojama dviejų tipų grafinė sąsaja – aplikacijos ir internetinė sąsaja. Internetinės grafinės sąsajos generavime naudojami šablonai, kuriuose gali būti realizuotas bet kokio sudėtingumo dizainas ar elementų išdėstymas.

9. Šiame darbe sukurtas įrankis leidžia redaguoti grafinę vartotojo sąsają esant pakeitimams. Į grafinę vartotojo sąsają galima ne tik įterpti bei redaguoti esamus langus, bet ir elementus bei mygtukus.
10. Atlikus eksperimentą, nustatyta, kad sukurtą grafinės vartotojo sąsajos generavimo įrankį verta naudoti įvairaus sudėtingumo sistemų grafinėms sąsajoms generuoti. Įrankis naudingas tuo atveju, kai sistemos projektavimo etape sudaromos UML diagramos.
11. Naudojantis sukurtu įrankiu, grafinė sąsaja gali būti ne tik sudaroma bei redaguojama, bet ir sugeneruojamas aplikacijos (*Java* programavimo kalba) bei internetinės sąsajos (HTML) programinis kodas.

LITERATŪRA

1. Jiang He, I-Ling Yen „Adaptive User Interface Generation for Web Services“ // IEEE International Conference on e-Business Engineering, 2007 ICEBE Hong Kong, p. 536 – 539.
2. Rosado da Cruz A. M., Pascoal de Faria J. „Automatic Generation of User Interfaces from Domain and Use Case Models“ // Sixth International Conference on the Quality of Information and Communications Technology, 2007 QUATIC Lisbon, p. 208 – 212.
3. Blankenhorn K. „A UML Profile for GUI Layout“: magistro darbas. University of Applied Sciences Furtwangen, department of Digital Media. [Furtwangen], 2004. 130 p.
4. Unifiede Modeling Language Superstructure Specification. Version 2.0. Dokumento numeris ptc/03-08-02. OMG, 2003.
5. WSUI Working Group. WSUI Executive White Paper. Epicentric, Inc [interaktyvus]. 2002, [žiūrėta 2009-02-02]. Prieiga per internetą: < http://www.wsui.org/doc/WSUI-wp.pdf>.
6. Aloia N., Concordia C., Paratore M. T. „Automatic GUI Generation for Web Based Information Systems“: tarptautinės konferencijos pranešimų medžiaga iš IADIS International Conference WWW/Internet. Algarve (Portugalija), 2003 lapkritis, p. 161-168.
7. Elkoutbi M., Khriss I., K. Keller R. „User Interface Prototyping using UML Specifications“. 2009, [žiūrėta 2009-10-12]. Prieiga per internetą: <http://www.iro.umontreal.ca/~keller/Suip/bookChapter.pdf>, p. 25.
8. Polak G., Jarosz J. „Automatic Graphical User Interface Form Generation Using Template Haskell“ Department of Computer Science University of Science and Technology, Krokuva (Lenkija) 2008. [žiūrėta 2008-10-11]. Prieiga per internetą: < http://www.cs.nott.ac.uk/~nhn/TFP2006/Papers/01-PolakJarosz-AutomaticGUIFormGenerationUsingTH.pdf >
9. MagicDraw programinio paketo vartotojo instrukcija. Prieiga per internetą: < https://secure.nomagic.com/files/manuals/MagicDraw%20UserManual.pdf>
10. *Oracle Forms Builder* programinio paketo vartotojo instrukcija. Prieiga per internetą: < http://www.oracle.com/technology/documentation/forms.html>

11. *IBM® InfoSphere Master Data Management (MDM) Workbeanch* programinio paketo vartotojo instrukcija. Prieiga per internetą: < http://dl.alphaworks.ibm.com/technologies/wpcw/Script-Workbench-for-IBM-InfoSphere-Master-Data-Managment-Server-for-Product-Information-Management-User-Guide-v1.1.pdf>

Santrumpų ir terminų žodynas

UML (angl. *Unified Modeling Language*) – modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektinių programų dokumentus.

WSDL (angl. *Web Services Description Language*) – interneto paslaugas aprašanti kalba

OCL (angl. *Object Constraint Language*) – deklaratyvi kalba, kuri aprašo UML modelių taisykles.

RUP (angl. *Rational Unified Process* – IBM organizacijos sukurtas programų sistemų kūrimo procesas

XP (angl. *Extreme Programming*) – ribinis programavimas, kurio metu siekiama kuo anksčiau gauti galutinį rezultatą

MDA (angl. *Model-driven architecture*) – modeliais grindžiama architektūra.

DDL (angl. *Data Definition Language*) – kompiuterinė kalba aprašanti duomenų struktūras.

XML (angl. *Extensible Markup Language*) – duomenų struktūrų bei jų turinio aprašomoji kalba.

XUL (angl. *XML User Interface Language*) – kalba žymėmis aprašanti vartotojo sąsają.

DOM (angl. *Document Object Model)* – dokumentų nuskaitymo būdas, kai visa dokumento struktūra nuskaitoma į atmintį ir visi veiksmai atliekami su atmintyje nuskaitytu dokumento elementų medžiu.

CRUD (angl. *Create*, *Read*, *Update*, *Delete*) – skaityti, rašyti, atnaujinti ir trinti.

SAX (angl. *Simple API for XML*) – įvykiais grindžiamas XML dokumento nuskaitymas.

Priedas Nr.1. Vartotojo vadovas

1. Darbo pradžia

Norint pradėti dirbti su grafinės vartotojo sąsajos generavimo įrankiu, reikia ją paleisti vartotojo kompiuteryje. Kaip jau minėta, norint įdiegti sistemą vartotojo kompiuteryje, reikalinga ne mažesnė nei *Java 1.6* versija. Sistemos programiniai failai išsaugomi pasirinktoje kategorijoje, o sistemos darbas pradedamas paleidus *GUI.exe* failą.

2. Sistemos valdymas

Sistemos valdymas vyksta meniu pagalba. Pagrindiniai meniu punktai – *Failas*, *Veiksmai*, *Peržiūra* ir *Apie*.

Failas. Šiame meniu pasirenkamos pagrindinės įrankio funkcijos, tokios kaip naujo projekto sukūrimas, *MagicDraw UML* įrankio XML failo įkėlimas ir pan.

Veiksmai. Šiame meniu sąraše galima pasirinkti pagrindines su projektu susijusias funkcijas – sąsajos generavimo, langų, elementų bei mygtukų įtraukimo bei redagavimo. Taip pat galima sugeneruoti programinį kodą.

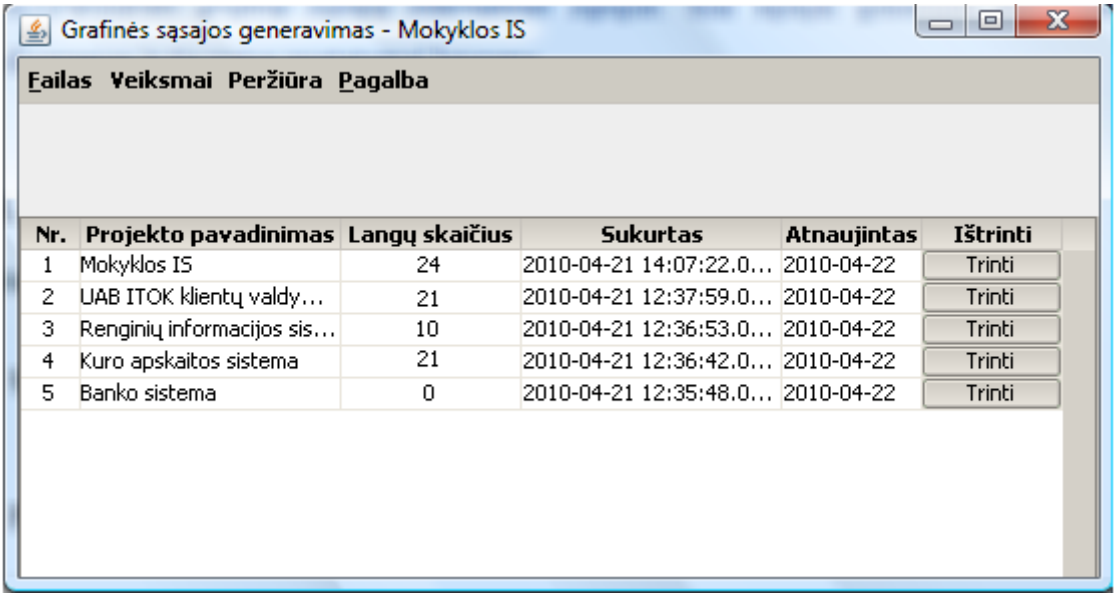
Peržiūra. Šioje skiltyje pasirenkamas būdas, kaip bus peržiūrima grafinė vartotojo sąsaja. Ji gali būti peržiūrima pritaikyta aplikacijai bei internetinei sąsajai.

Pagalba. Šioje skiltyje pateikiama trumpa informacija apie grafinės vartotojo sąsajos generavimo įrankį.

3. Projektų valdymas

Paleidus sistemą, pradiniame sistemos lange (1 pav.) matomas įrankyje jau sukurtų projektų sąrašas bei pagrindiniai jų duomenys: projekto numeris, pavadinimas, sugeneruotų langų skaičius, sukūrimo bei projekto atnaujinimo datos. Kiekvieno projekto eilutėje taip pat yra mygtukas, leidžiantis ištrinti pasirinktą projektą.

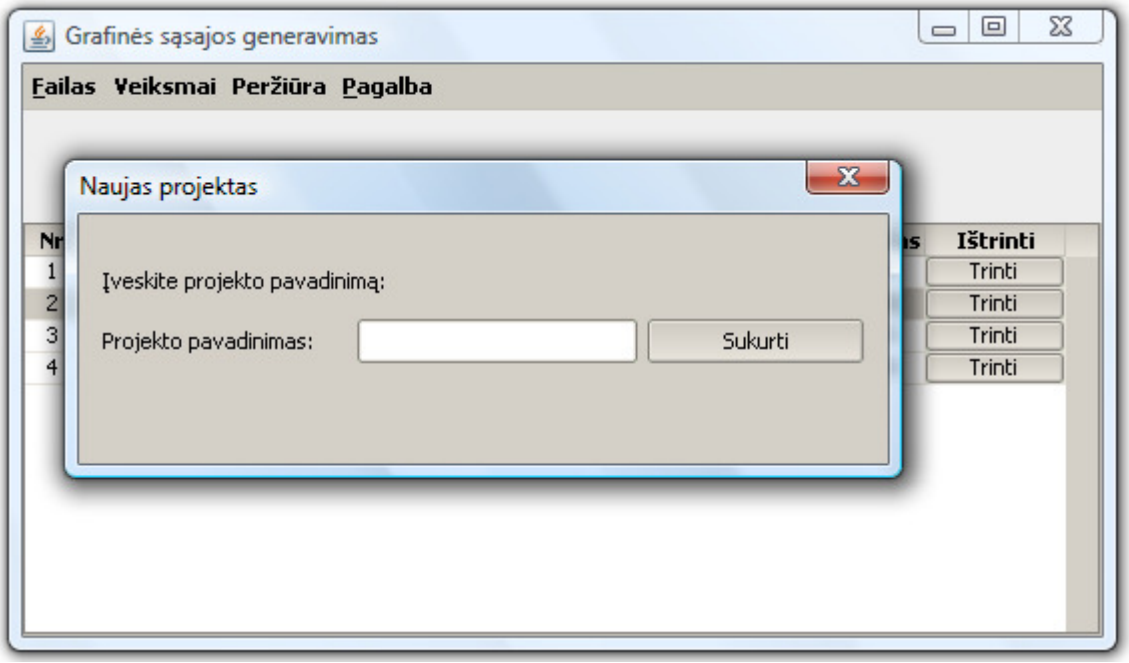
Grafinės vartotojo sąsajos funkcijos pasirenkamos meniu pagalba.



1 pav. Pagrindinis grafinės vartotojo sąsajos generavimo langas

3.1 Projekto sukūrimas

Naujo projekto sukūrimo funkcija iškviečiama meniu pasirinkus *Failas -> Naujas projektas*. Parodomas naujas langas su prašymu įvesti naujo projekto pavadinimą (2 pav.). Įvedus naujo projekto pavadinimą ir paspaudus mygtuką „Sukurti“ sukuriamas naujas projektas, kuris rodomas projektų sąraše.



2 pav. Naujo projekto sukūrimo langas

3.2 Projekto šalinimas

Pasirinktas projektas gali būti pašalinamas pagrindiniame įrankio lange atitinkamoje eilutėje paspaudus mygtuką „Trinti“. Sistema parodys pranešimą „Ar tikrai norite ištrinti projektą?“. Pasirinkus teigiamą atsakymą, projektas ir visi su juo susiję duomenys bus ištrinti.

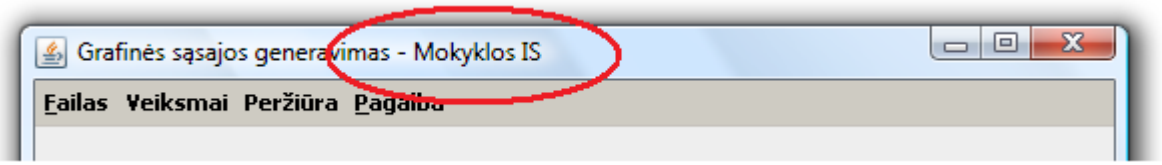
3.3 Projekto uždarymas

Projekto uždarymo funkcija vykdoma pasirenkant meniu *Failas* -> *Uždaryti projektą*. Projekto pavadinimas neberodomas įrankio lango viršuje, deaktyvuojamos tos meniu funkcijos, kurios susiję su projektais, t.y. nebegalima generuoti grafinės vartotojo sąsajos, jos peržiūrėti, kurti ar redaguoti naujų langų ir t.t.

4. Darbas su projektu

4.1 Projekto pasirinkimas

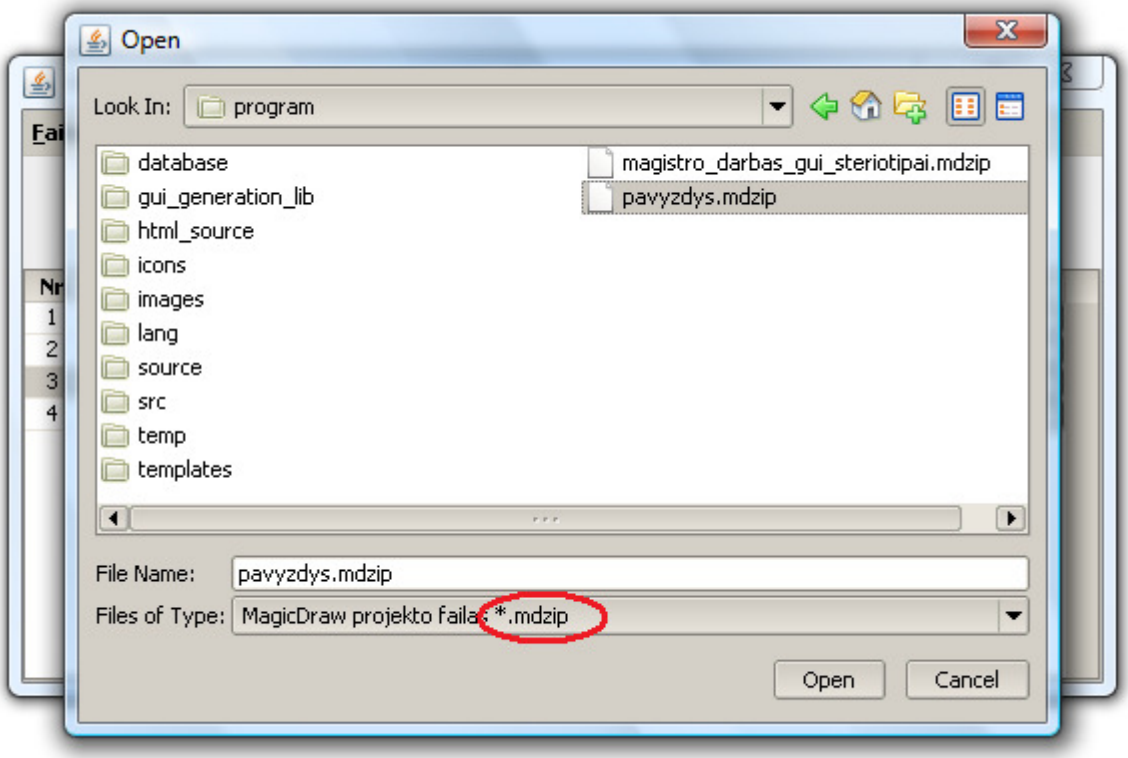
Darbas su konkrečiu projektu pradedamas pasirinkus jį projektų sąraše. Pasirinkus projektą, jo pavadinimas rodomas įrankio pagrindinio lango pavadinime (3 pav.).



3 pav. Pasirinkto projekto pavadinimo rodymas

4.2 MagicDraw projekto failo įkėlimas

MagicDraw UML projekto XML failo įkėlimas vykdomas meniu pasirinkus *Failas* -> *Įkelti MagicDraw failą*. Parodomas naujas langas, kuriame pasirinkamas *MagicDraw UML* projekto failas ir spaudžiamas mygtukas „Open“. Projekto failo formatas turi būti *.mdzip. Kitokio formato failai nėra įkeliami. Įkėlus projekto failą, galima generuoti grafinę vartotojo sąsają.



4 pav. *MagicDraw* projekto failo įkėlimas

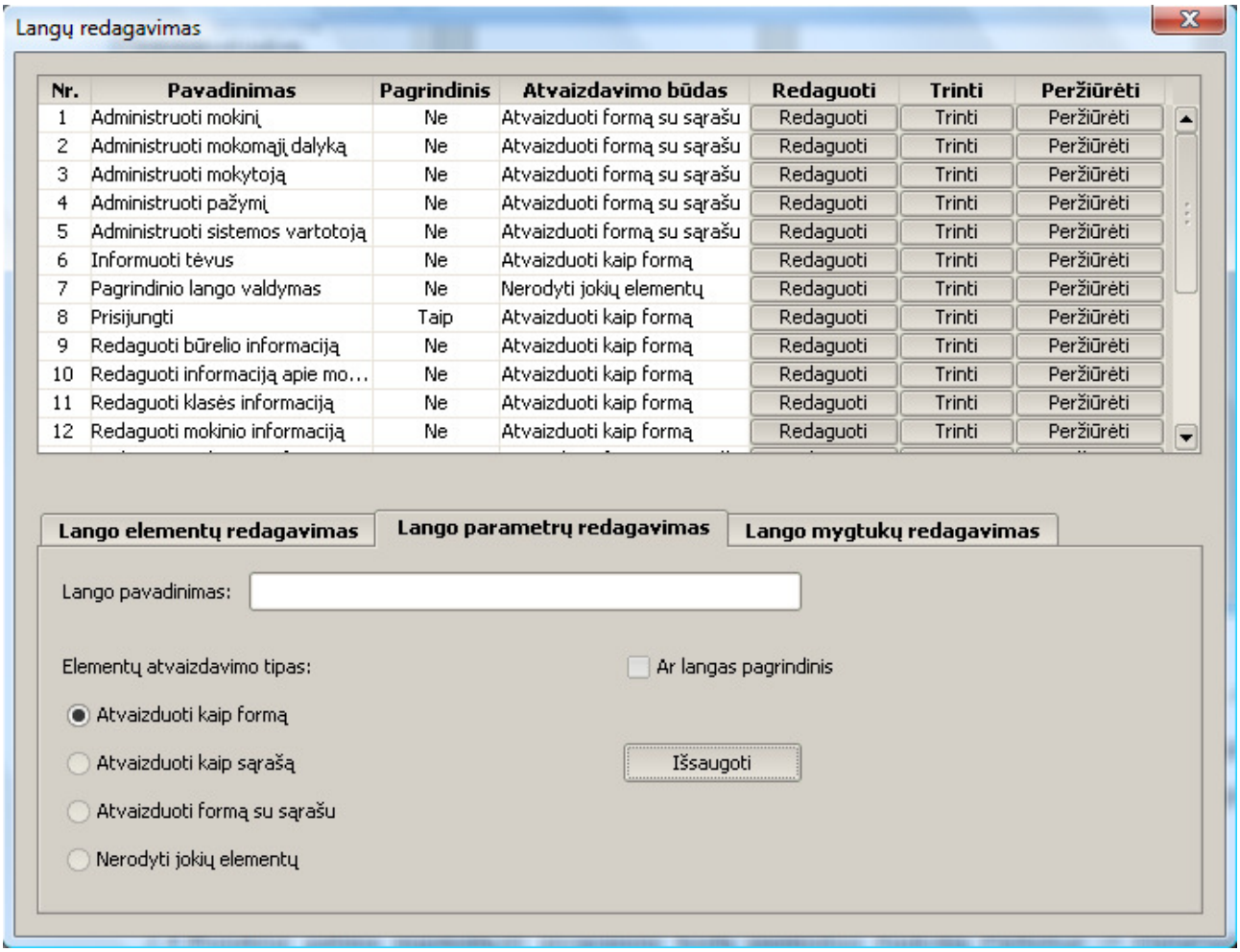
5. Grafinės vartotojo sąsajos generavimas

Grafinė vartotojo sąsaja generuojama pasirinkus projektą iš sąrašo bei pasirinkus meniu funkciją *Veiksmai -> Generuoti*. Grafinę sąsają vienam projektui galima sugeneruoti tik vieną kartą. Toliau langai, jų elementai bei mygtukai gali būti redaguojami bei įtraukiami pasinaudojant tam skirtomis funkcijomis.

Sugeneravus grafinę vartotojo sąsają, ją galima ne tik peržiūrėti dviem formatais (pritaikytą aplikacijai bei internetinei sąsajai), bet ir redaguoti bei kurti naujus langus, jų elementus, mygtukus.

6. Grafinės vartotojo sąsajos redagavimas

Sugeneruotus ar įtrauktus naujus langus galima generuoti pasirinkus funkciją *Veiksmai -> Langų redagavimas*. Grafinės sąsajos redagavimo lange parodomas langų sąrašas su galimybe kiekvieną pasirinktą langą redaguoti (elementus, mygtukus bei lango parametrus), ištrinti ir peržiūrėti. Langų redagavimo aplinka pateikta 5 paveiksle.



5 pav. Langų redagavimo aplinka

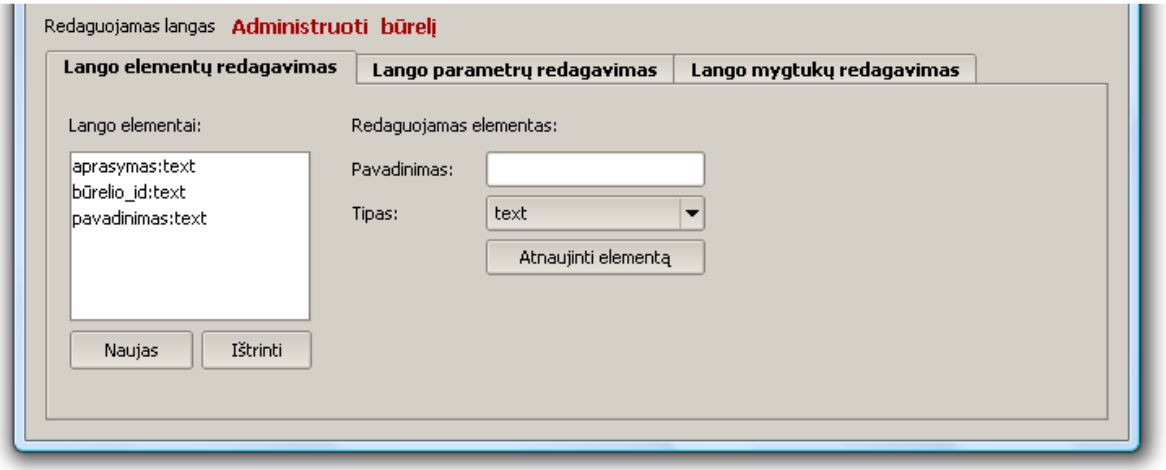
Pasirinkus konkretų langą ir paspaudus mygtuką „Redaguoti“, to lango duomenys, elementai, mygtukai ir t.t. parodomi antroje lango dalyje, kur juos galima redaguoti. Ši lango dalis suskirstyta į tris dalis: lango elementų, lango parametrų ir lango mygtukų redagavimą.

Taip pat nereikalingus langus galima šalinti paspaudus mygtuką „Trinti“. Sistema parodys pranešimą „Ar tikrai ištrinti langą?“. Pasirinkus teigiamą atsakymą, langas ir visa su juo susijusi informacija bus ištrinta.

Lango peržiūra vykdoma paspaudus mygtuką „Peržiūrėti“. Naujame lange parodoma pasirinktas langas, jo elementai, mygtukai.

6.1 Lango elementų redagavimas

Lango elementų redagavimo forma pateikta 6 paveiksle.

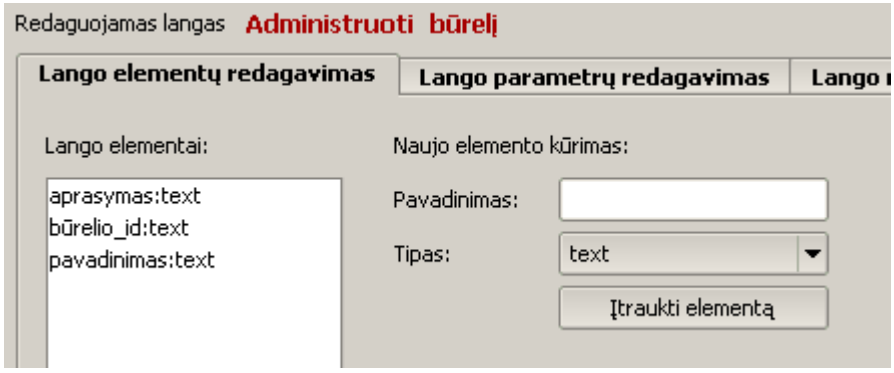


6 pav. Pasirinkto „Administruoti būrelį“ lango elementų redagavimo forma

Lango elementų redagavimo forma suskirstyta į dvi dalis. Kairioje pusėje rodomas jau esamų elementų sąrašas su mygtukais „Naujas“ ir „Trinti“, o dešinėje – pasirinkto iš sąrašo redaguojamo elemento forma arba naujo kuriamo elemento forma (pagal poreikį).

6.1.1 Naujo elemento kūrimas

Naujas pasirinkto lango elementas gali būti sukuriamas pasirinktus mygtuką „Naujas“. Dešinėje esančioje formoje įvedamas elemento pavadinimas ir pasirenkamas jo tipas. Įvedus šiuos duomenis, ir paspaudus mygtuką „Įtraukti elementą“, elementas yra išsaugomas ir rodomas elementų sąraše. Elementas nebus išsaugomas tuo atveju, jei jis neturės pavadinimo.



7 pav. Naujo elemento kūrimo forma

6.1.2 Elemento redagavimas

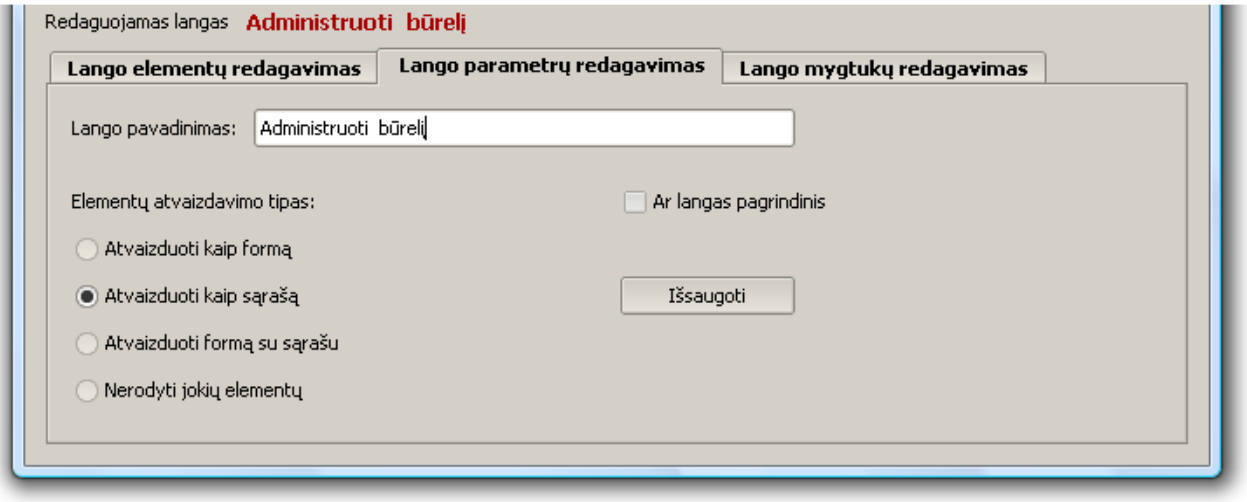
Esant poreikiui, esamus elementus galima redaguoti, t.y. keisti jų pavadinimą bei tipą. Kairėje esančiame elementų sąraše pasirinkus elementą, jo duomenys iškart parodomi dešiniau esančioje formoje. Pakeitus reikiamus duomenis ir paspaudus mygtuką „Atnaujinti elementą“, jo duomenys išsaugomi. Elemento duomenys nebus atnaujinami tuo atveju, jei bus ištrintas jo pavadinimas.

6.1.3 Elemento ištrynimas

Elementą pasirinkus iš sąrašo ir paspaudus mygtuką „Trinti“ jį galima pašalinti. Jo duomenys ištrinami ir jis daugiau nebus atvaizduojamas grafinėje vartotojo sąsajoje.

6.2 Lango parametrų redagavimas

Lango parametrų redagavimo forma pateikta 8 paveiksle.



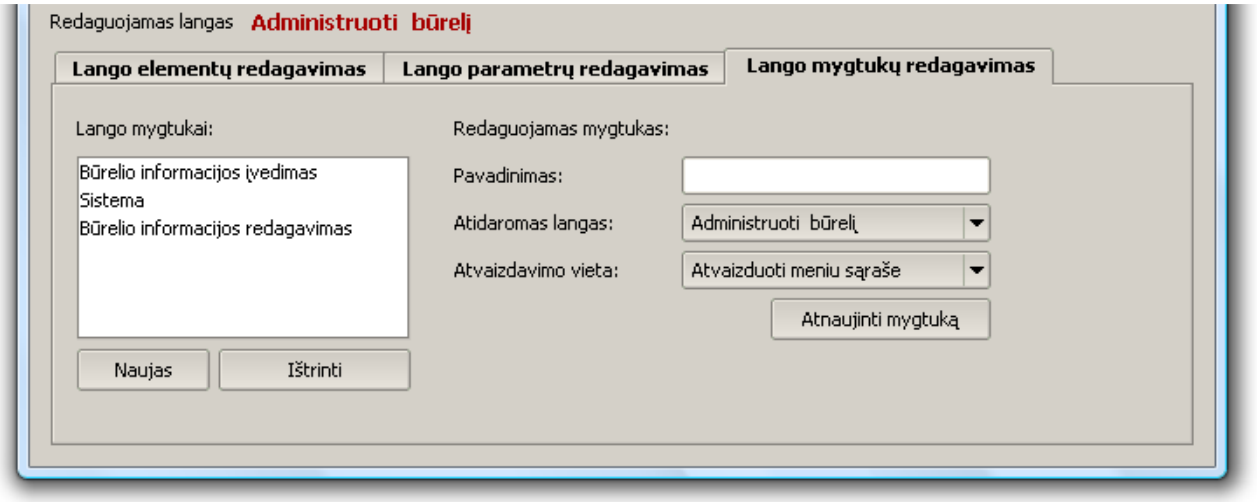
8 pav. Pasirinkto „Administruoti būrelį“ lango parametrų redagavimo forma

Lango parametrų redagavimo formoje rodomi pagrindinai duomenys apie pasirinktą langą – jo pavadinimas, tipas, ir pasirinkimas, ar šis langas yra pagrindinis. Ištrynus lango pavadinimą ir bandant išsaugoti duomenis, sistema parodys klaidos pranešimą, kadangi langas būtinai turi turėti savo pavadinimą.

Lango elementų atvaizdavimo tipai yra keturi – langas gali būti atvaizduojamas kaip forma, sąrašas, forma ir sąrašas ir lange iš viso nerodomi jokie elementai.

6.3 Lango mygtukų redagavimas

Lango mygtukų redagavimo forma pateikta 9 paveiksle.



9 pav. Pasirinkto „Administruoti būrelį“ lango mygtukų redagavimo forma

Lango mygtukų redagavimo forma suskirstyta į dvi dalis. Kairioje pusėje rodomas jau esamų mygtukų sąrašas su mygtukais „Naujas“ ir „Ištrinti“, o dešinėje – pasirinkto iš sąrašo redaguojamo mygtuko forma arba naujo kuriamo mygtuko forma (pagal poreikį).

6.3.1 Naujo mygtuko kūrimas

Naujas pasirinkto lango mygtukas gali būti sukuriamas pasirinktus mygtuką „Naujas“. Dešinėje esančioje formoje įvedamas mygtuko pavadinimas, pasirenkama, kurį langą mygtukas atidarys bei kur turi būti rodomas – formos apačioje ar įtraukiamas į meniu. Įvedus šiuos duomenis, ir paspaudus mygtuką „Įtraukti elementą“, elementas yra išsaugomas ir rodomas elementų sąraše.

Mygtukas nebus išsaugomas tuo atveju, jei jis neturės pavadinimo.

6.3.2 Mygtuko redagavimas

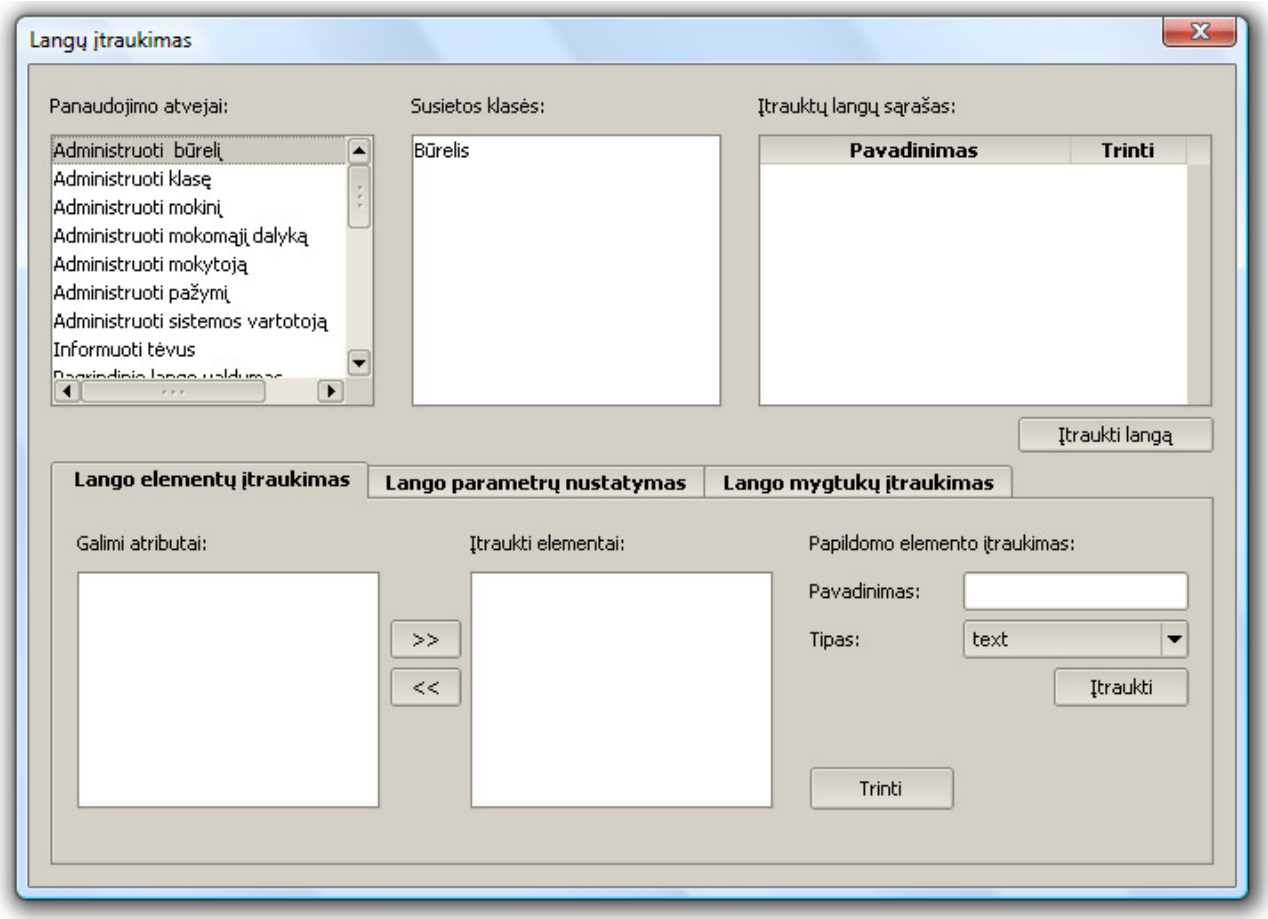
Esant poreikiui, esamus mygtukus galima redaguoti, t.y. keisti jų pavadinimą, atidaromą langą ar vietą. Kairėje esančiame elementų sąraše pasirinkus mygtuką, jo duomenys iškart parodomi dešiniau esančioje formoje. Pakeitus reikiamus duomenis ir paspaudus mygtuką „Atnaujinti mygtuką“, duomenys išsaugomi. Mygtuko duomenys nebus atnaujinami tuo atveju, jei bus ištrintas jo pavadinimas.

6.3.3 Mygtuko šalinimas

Mygtuką pasirinkus iš sąrašo ir paspaudus mygtuką „Ištrinti“ jį galima pašalinti. Jo duomenys ištrinami ir jis daugiau nebus atvaizduojamas grafinėje vartotojo sąsajoje.

7. Naujų langų įtraukimas

Naujus langus į projektą galima įtraukti pasinaudojus funkcija *Veiksmai -> Naujų langų įtraukimas*. Būtini įvesti duomenys – lango pavadinimas ir bent vienas elementas. Langų įtraukimo aplinka pateikta 10 paveiksle.



10 pav. Langų įtraukimo aplinka

Norint įtraukti langą, pirmiausia reikia pasirinkti panaudojimo atvejį iš panaudojimo atvejų sąrašo. Pagal tai, koks pasirinktas panaudojimo atvejis, dešiniau esančiame lange parodoma su juo susieta klasė, kurios atributai gali būti įtraukiami į langą. Į vieną langą galima įtraukti atributus iš neriboto kiekio klasių. Elementui turi būti įvedamas pavadinimas bei parenkamas tipas.

Langų įtraukimo aplinka, panašiai kaip ir Langų redagavimo aplinka, suskirstyta į tris dalis – lango elementų, lango mygtukų įtraukimą ir lango parametrų nustatymą. Lango parametrų nustatymo ir lango mygtukų įtraukimo formos identiškos lango parametrų ir mygtukų redagavimo formoms.

Lango išsaugojimas vykdomas paspaudus mygtuką „Įtraukti langą“. Norint, kad langas būtų išsaugotas, būtina įvesti lango pavadinimą ir parinkti bent vieną elementą.

8. Programinio kodo generavimas

Projektui galima sugeneruoti programinį kodą pasirinkus funkciją *Veiksmai -> Generuoti programinį kodą*. Įvykdžius šią funkciją, parodomas pranešimas apie sėkmingą kodo sugeneravimą. Aplikacijos grafinės sąsajos programinis kodas išsaugomas „source“ direktorijoje, o internetinės grafinės sąsajos programinis kodas – „html_source“ direktorijoje.

Šiose direktorijose programinis kodas išsaugomas pagal projekto pavadinimą.

9. Peržiūra

Šioje skiltyje pasirenkamas būdas, kaip bus peržiūrima grafinė vartotojo sąsaja. Ji gali būti peržiūrima pritaikyta aplikacijai bei internetinei sąsajai.

9.1 Aplikacijos grafinės sąsajos peržiūra

Peržiūra -> Peržiūrėti aplikacijos grafinę sąsają parodo grafinę sąsają, kuri pritaikyta aplikacijai. Langai pradedami rodyti nuo pagrindinio sistemos lango.

9.2 Internetinės grafinės sąsajos peržiūra

Peržiūra -> Peržiūrėti internetinę grafinę sąsają parodo grafinę sąsają, kuri pritaikyta internetui. Langai pradedami rodyti nuo pagrindinio sistemos lango.

10. Informacija apie projektą

Funkcija *Pagalba -> Apie* parodo pagrindinę informaciją apie grafinės vartotojo sąsajos generavimo įrankį.

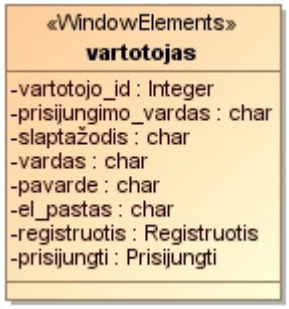
11. Darbo pabaiga

Funkcija *Failas -> Išeiti* uždaro grafinės vartotojo sąsajos generavimo įrankį.

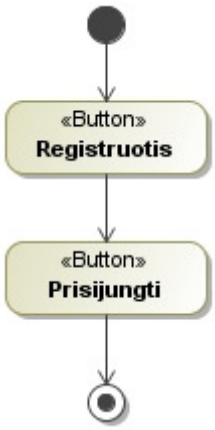
Priedas Nr. 2. Eksperimento duomenys ir rezultatai

4.1 Kai sistemos grafinė sąsaja kuriama su 2 panaudojimo atvejais, 1 klase ir 2 veiklomis

1 pav. I eksperimento bandymo panaudojimo atvejų diagrama



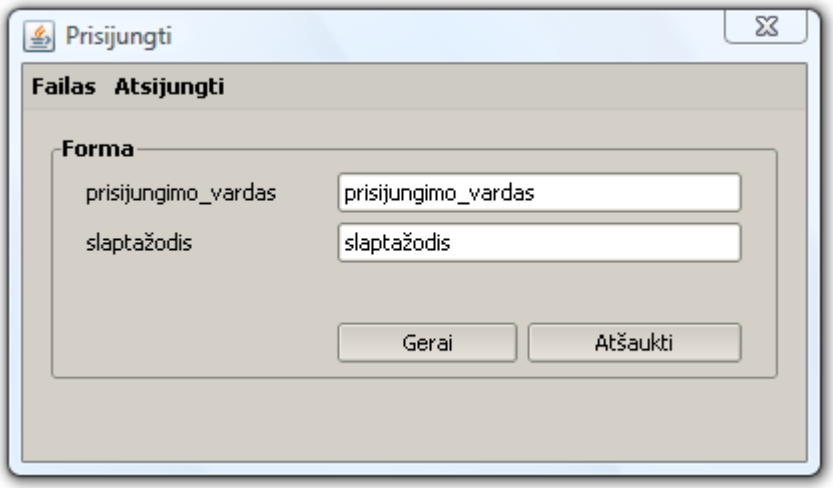
2 pav. I eksperimento bandymo klasių diagrama



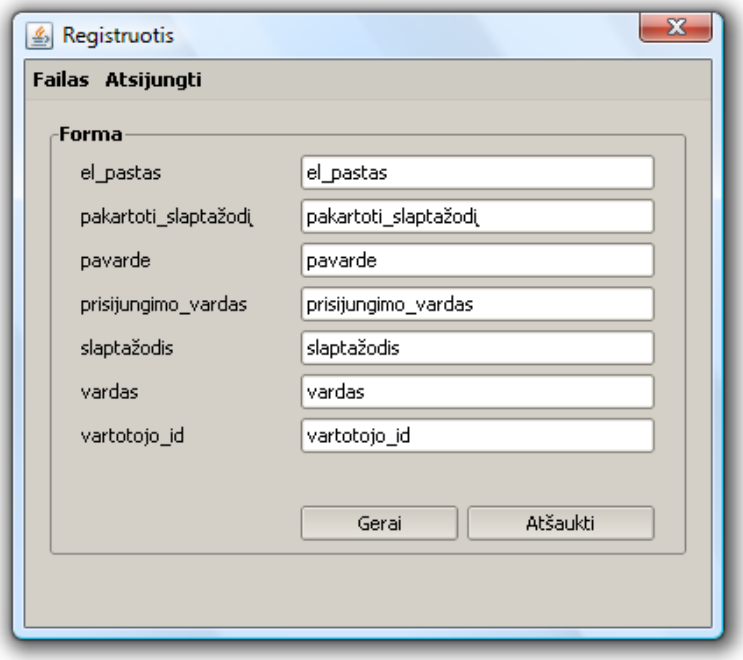
3 pav. I eksperimento bandymo veiklos diagrama

4 pav. Prisijungimo lango projektas

5 pav. Registracijos lango projektas

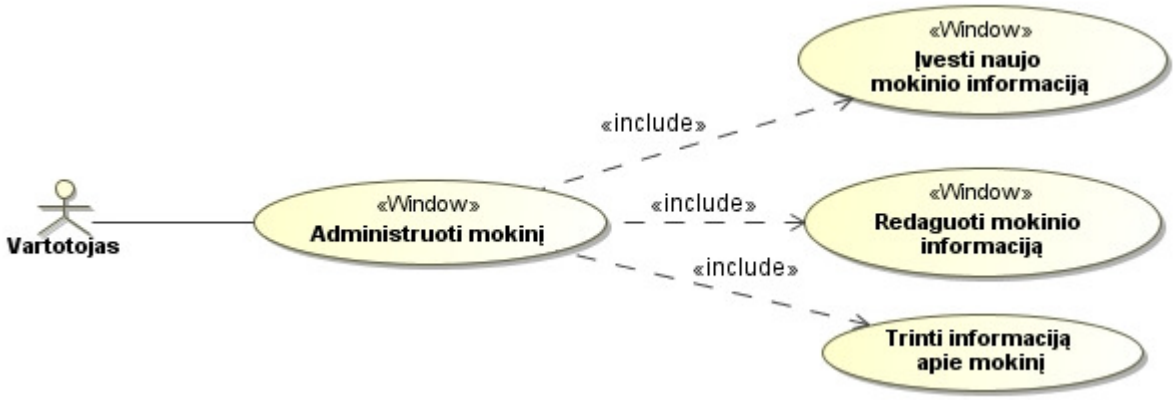


6 pav. Sugeneruotas prisijungimo langas

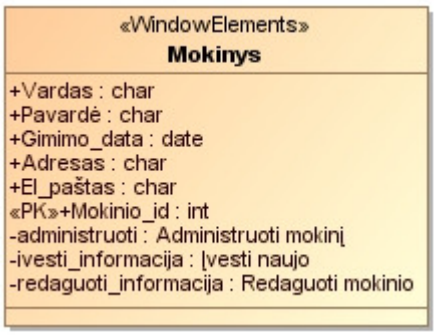


7 pav. Sugeneruotas registracijos langas

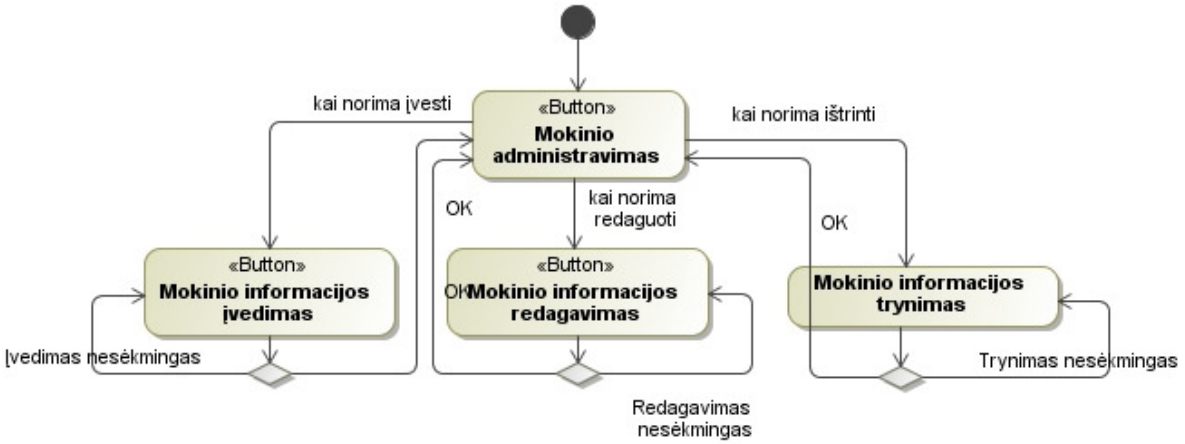
4.2 Kai sistemos grafinė sąsaja kuriama su 4 panaudojimo atvejais, 1 klase ir 4 veiklomis



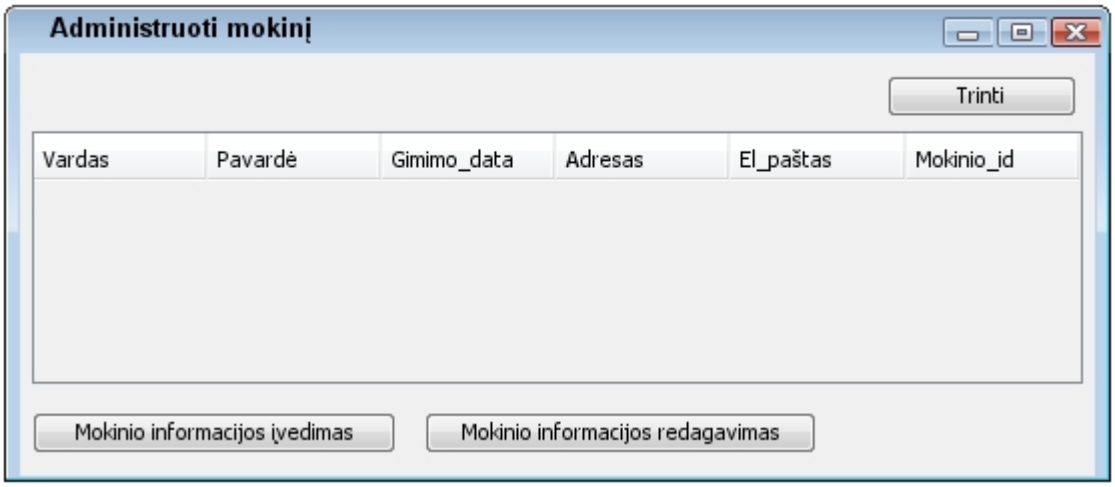
8 pav. II eksperimento bandymo panaudojimo atvejų diagrama



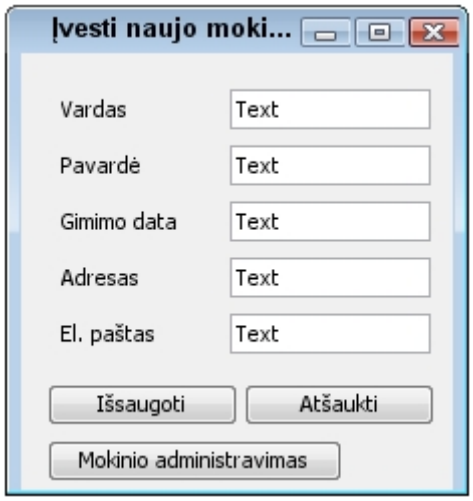
9 pav. II eksperimento bandymo klasių diagrama



10 pav. II eksperimento bandymo veiklos diagrama

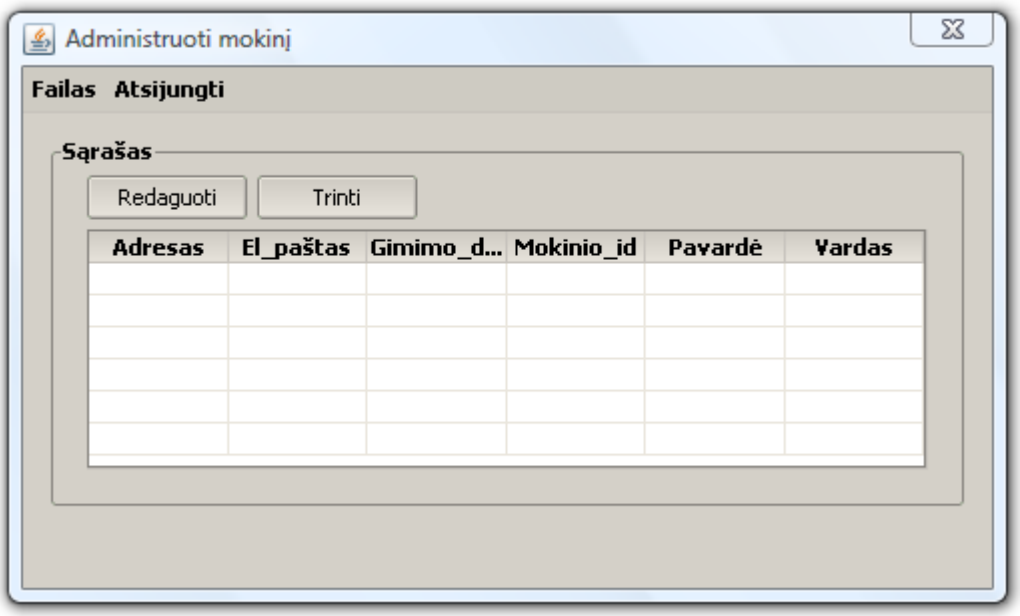


11 pav. Mokinių administravimo lango projektas

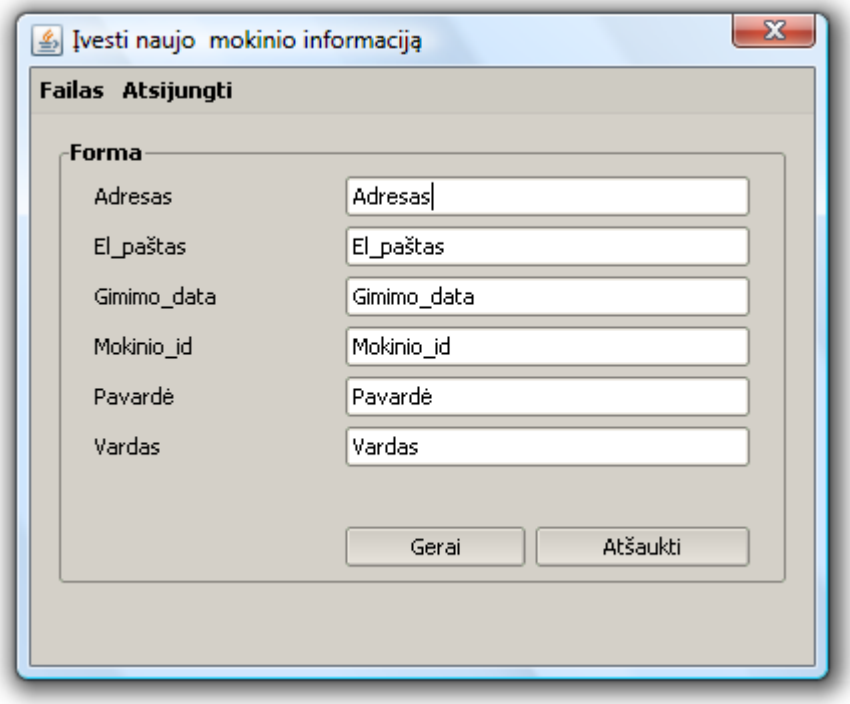


12 pav. Naujo mokinio įvedimo lango projektas

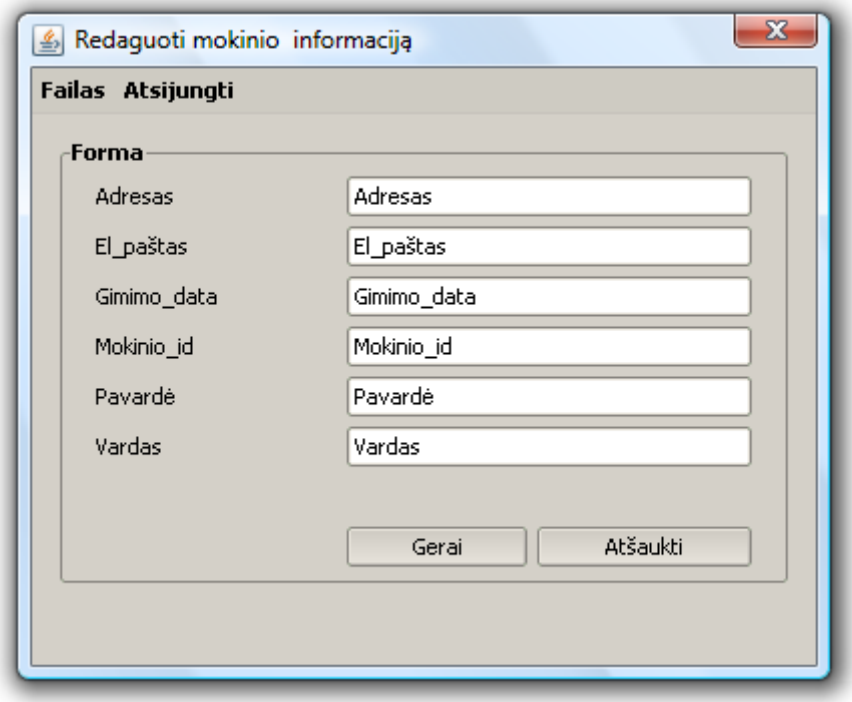
13 pav. Mokinio redagavimo lango projektas



14 pav. Sugeneruotas mokinių administravimo langas

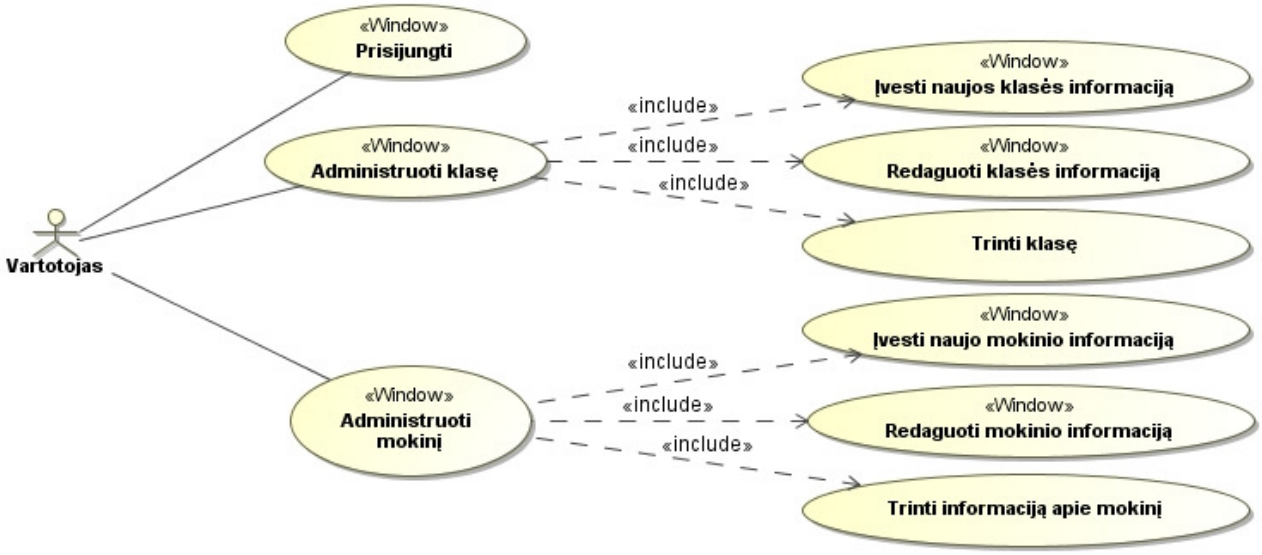


15 pav. Sugeneruotas mokinių įvedimo langas

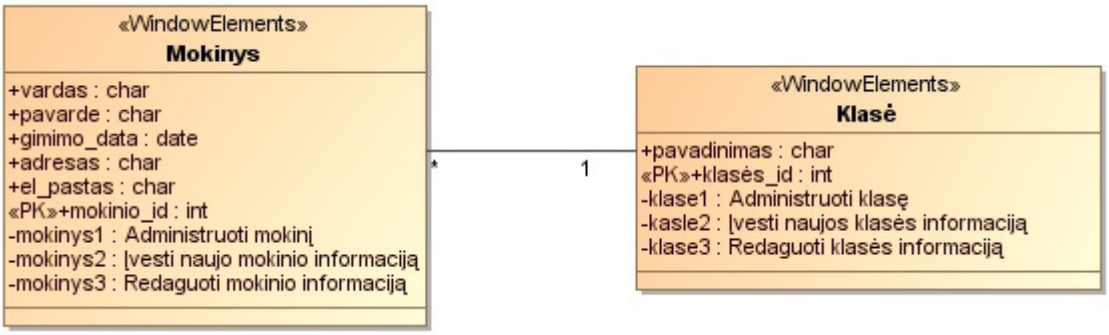


16 pav. Sugeneruotas mokinių redagavimo langas

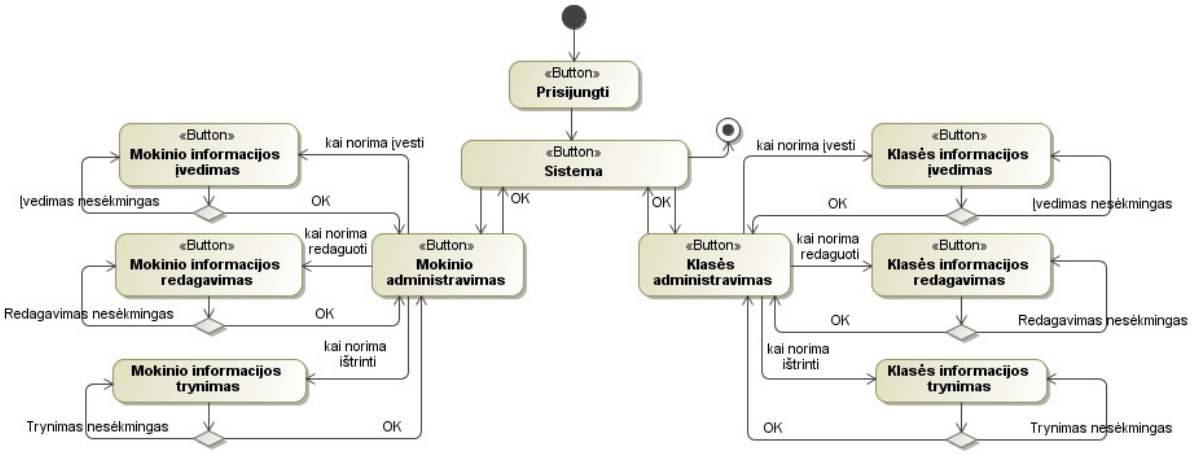
4.3 Kai sistemos grafinė sąsaja kuriama su 9 panaudojimo atvejais, 2 klase ir 10 veiklomis



17 pav. III eksperimento bandymo panaudojimo atvejų diagrama



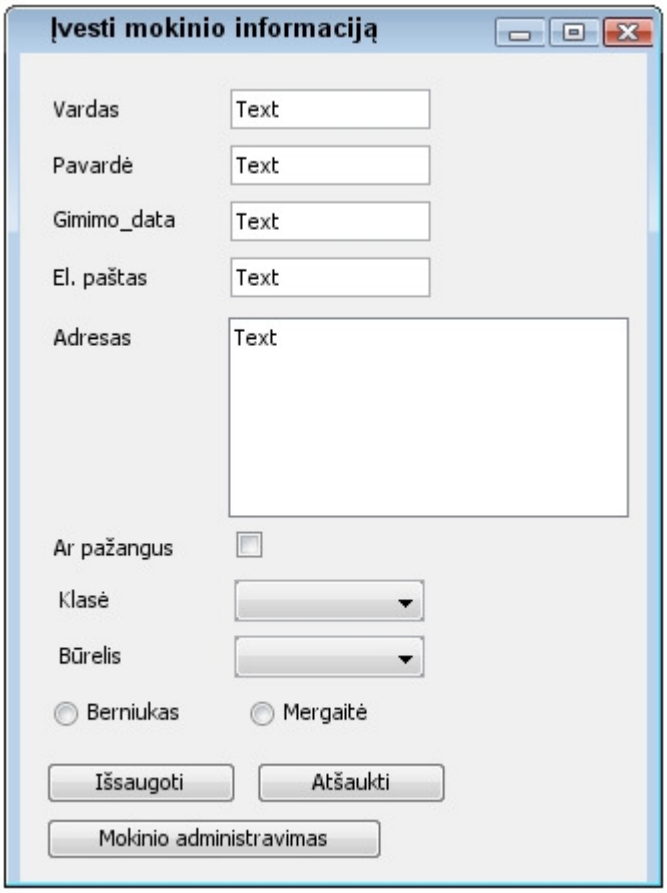
18 pav. II eksperimento bandymo klasių diagrama



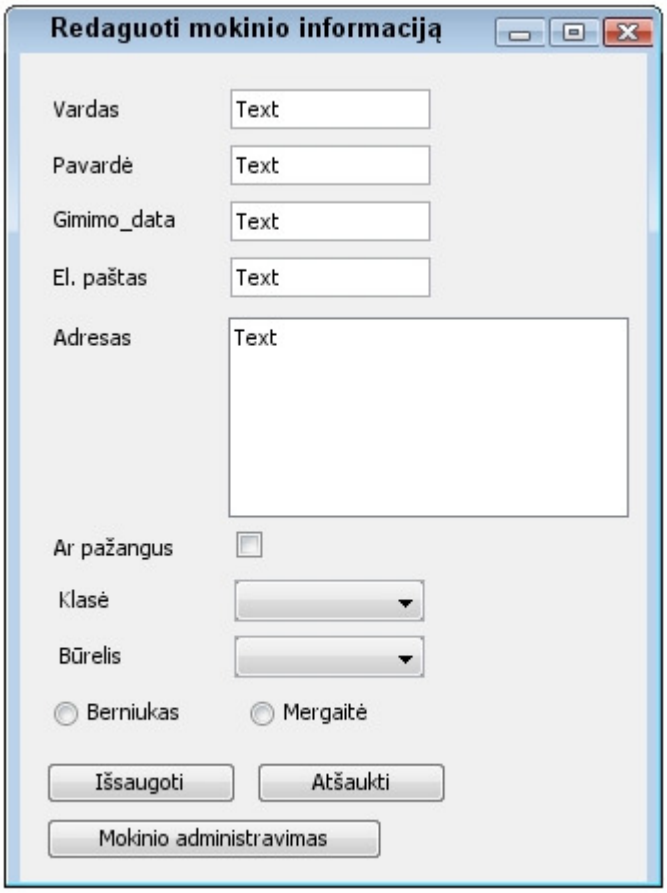
19 pav. II eksperimento bandymo veiklos diagrama

20 pav. Prisijungimo lango projektas

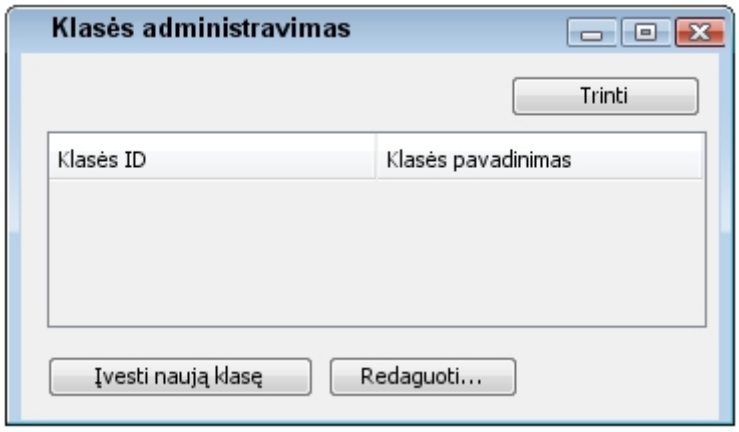
21 pav. Mokinių administravimo lango projektas



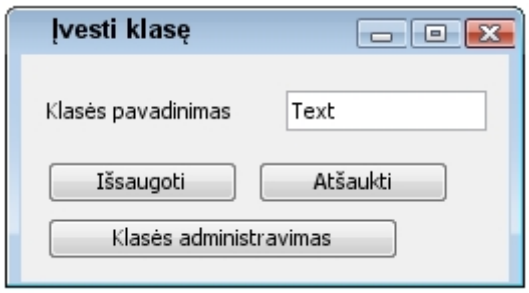
22 pav. Mokinio įvedimo lango projektas



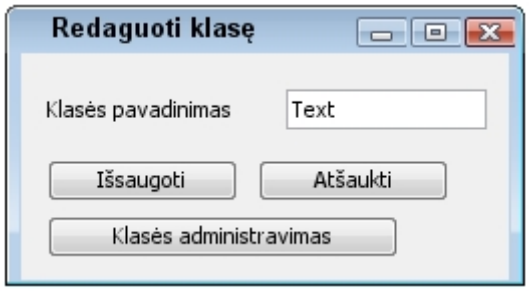
23 pav. Mokinio redagavimo lango projektas



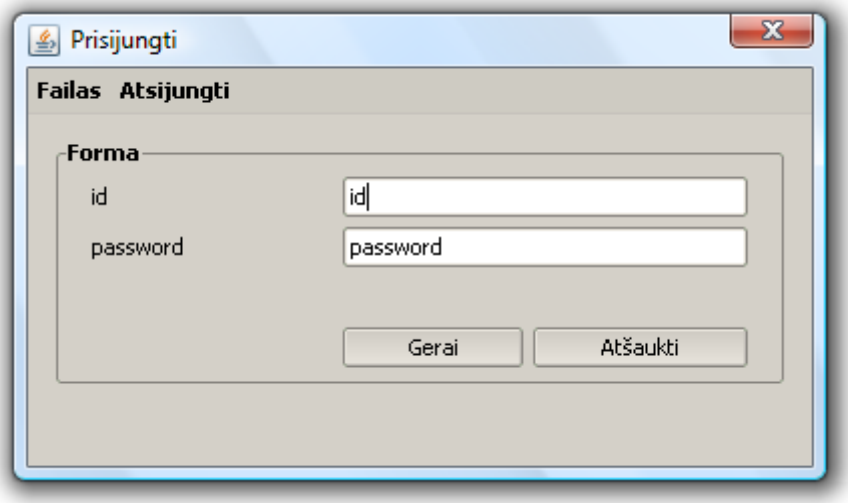
24 pav. Klasių administravimo lango projektas



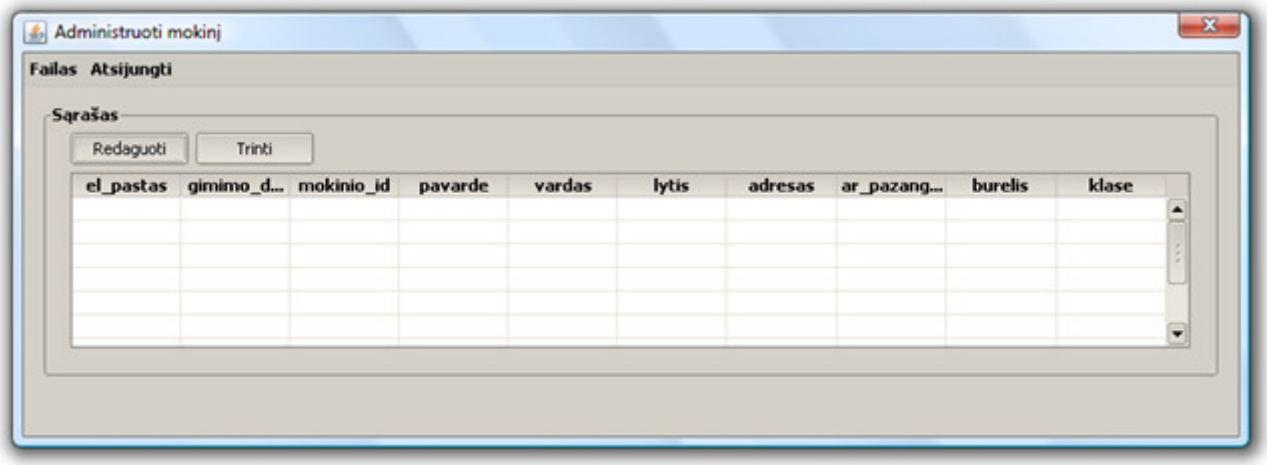
25 pav. Klasės įvedimo lango projektas



26 pav. Klasės redagavimo lango projektas



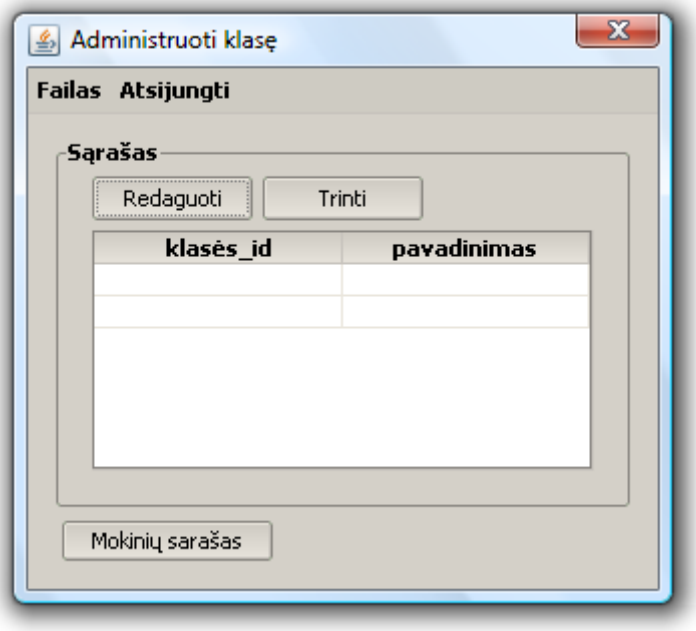
27 pav. Prisijungimo langas



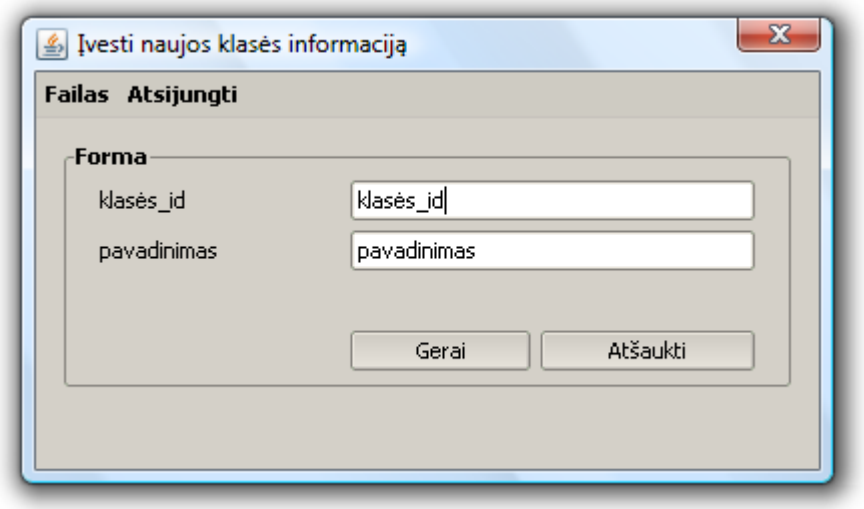
28 pav. Mokinių administravimo langas

29 pav. Mokinio informacijos įvedimo langas

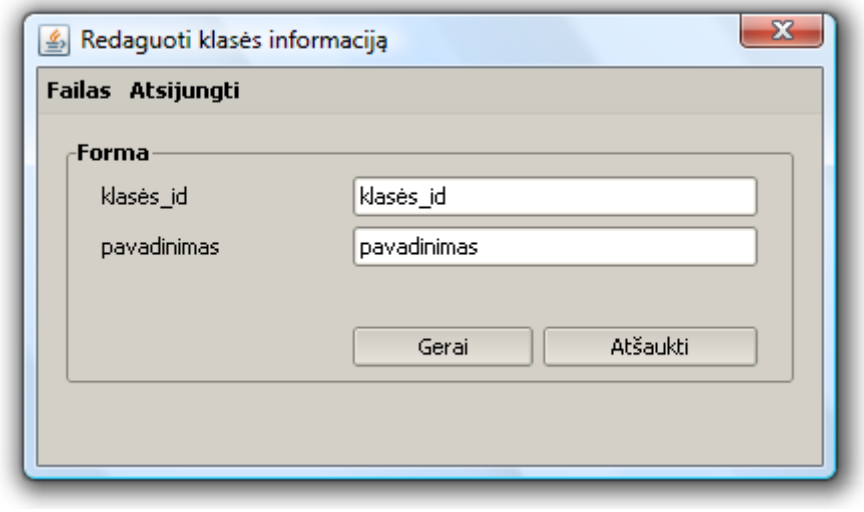
30 pav. Mokinio informacijos redagavimo langas



31 pav. Klasių administravimo langas

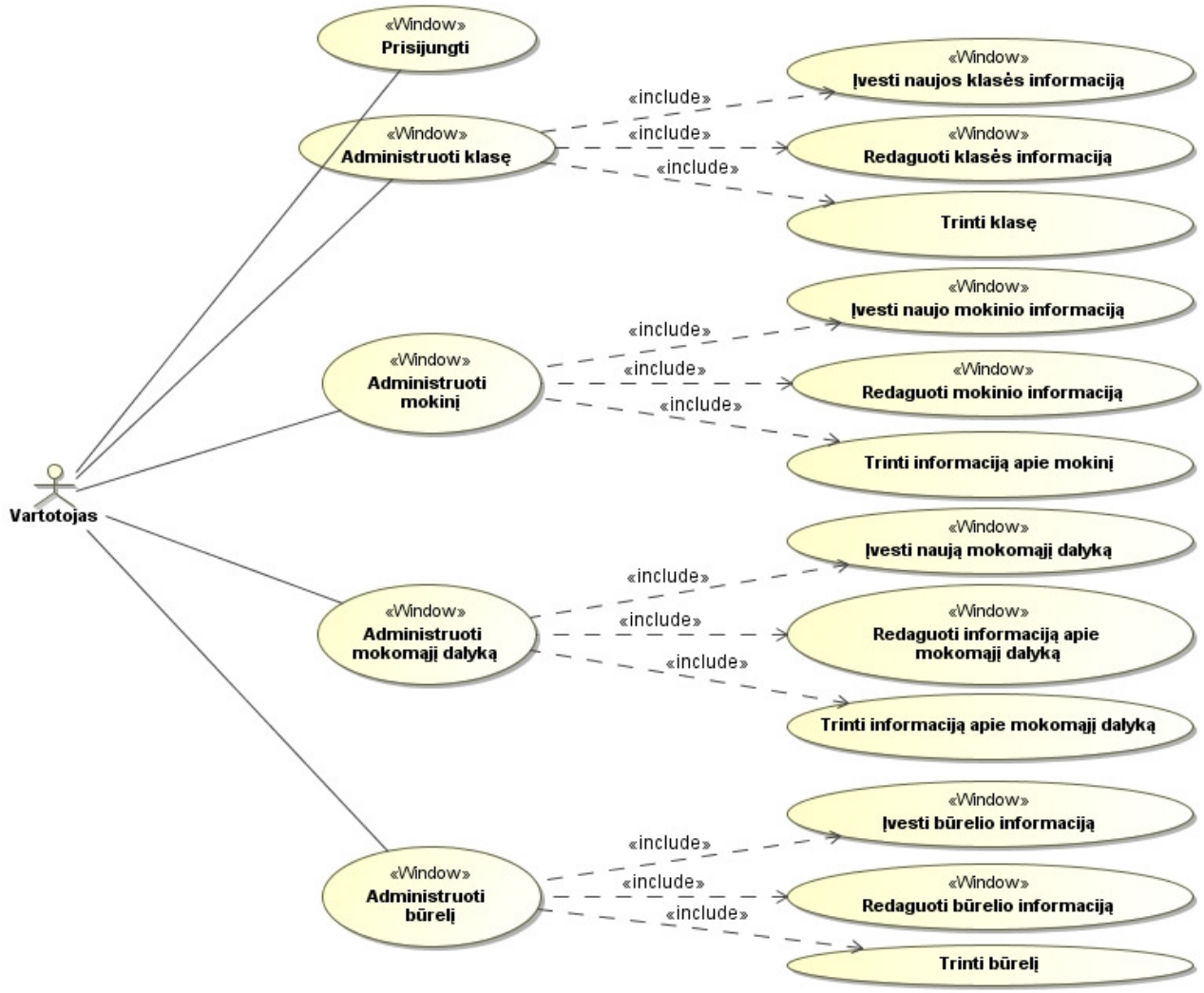


32 pav. Klasės informacijos įvedimo langas

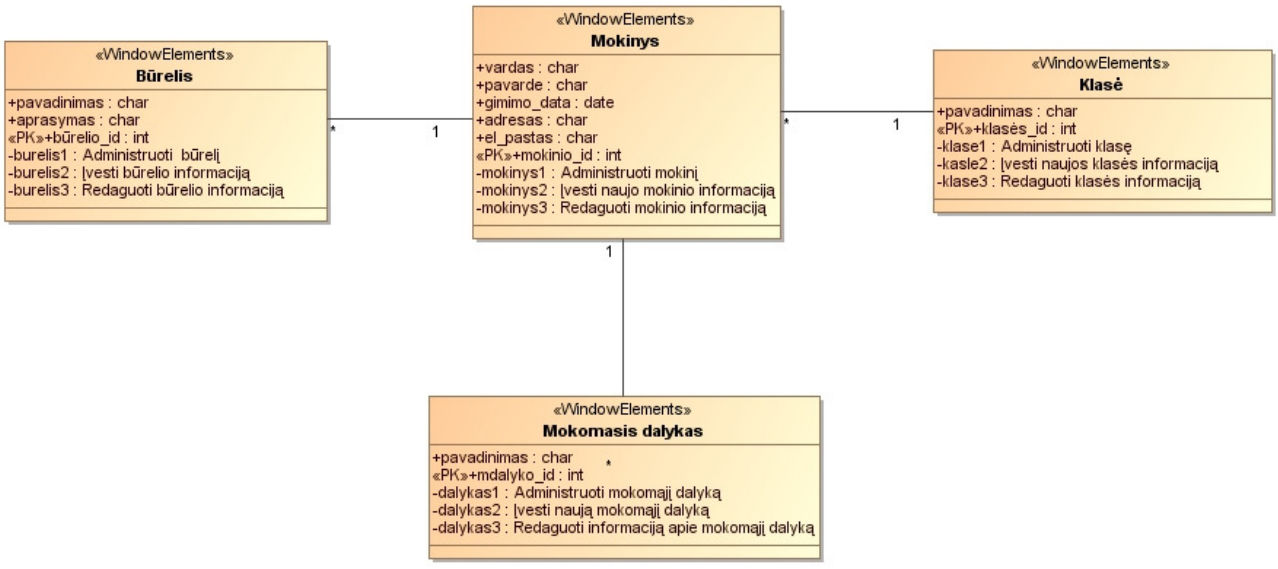


33 pav. Klasės informacijos redagavimo langas

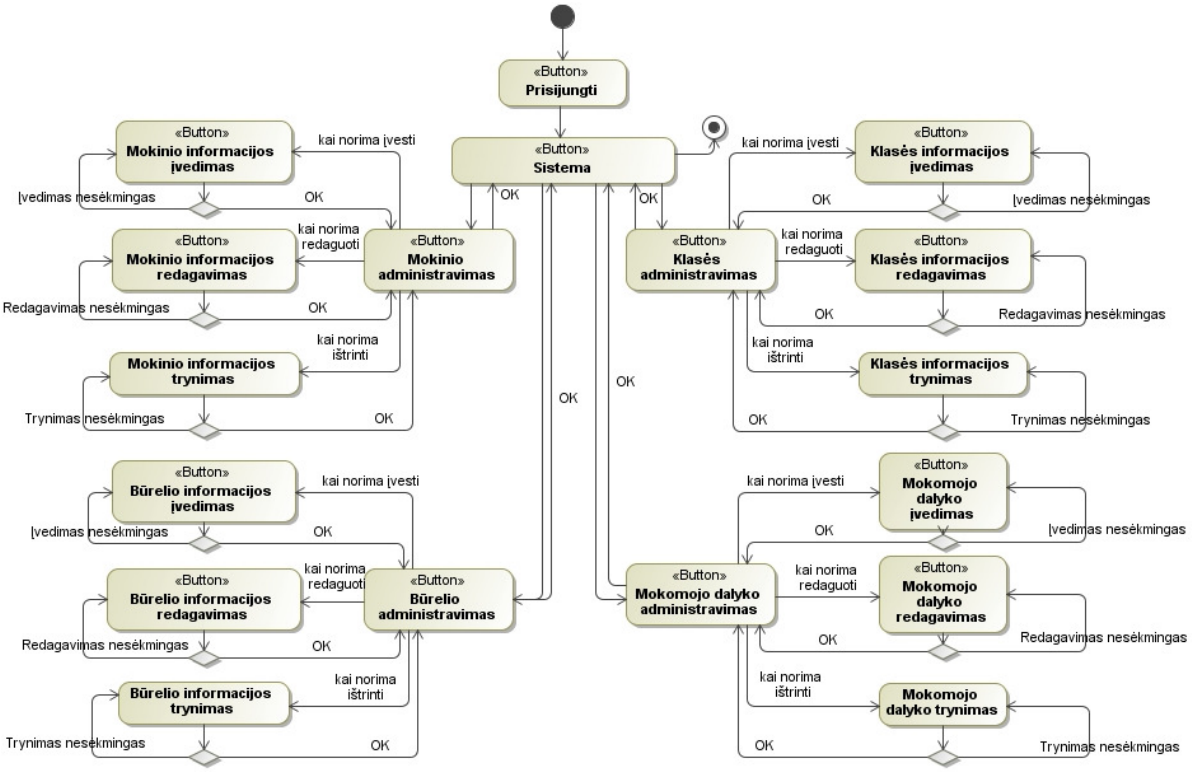
4.4 Kai sistemos grafinė sąsaja kuriama su 13 panaudojimo atvejų, 4 klasėmis ir 14 veiklų



34 pav. IV eksperimento bandymo panaudojimo atvejų diagrama

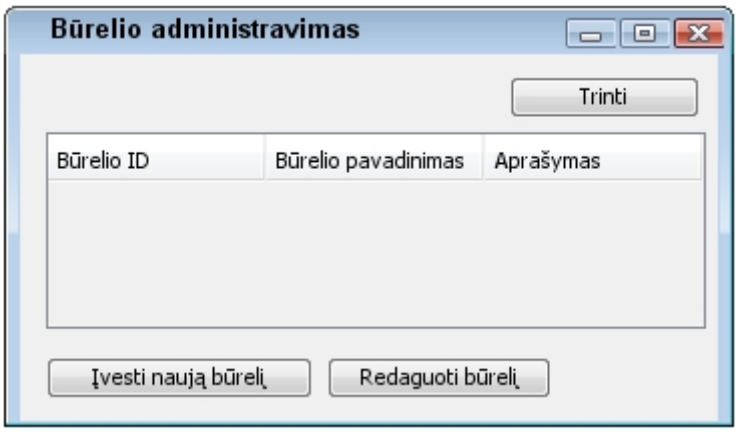


35 pav. IV eksperimento bandymo klasių diagrama

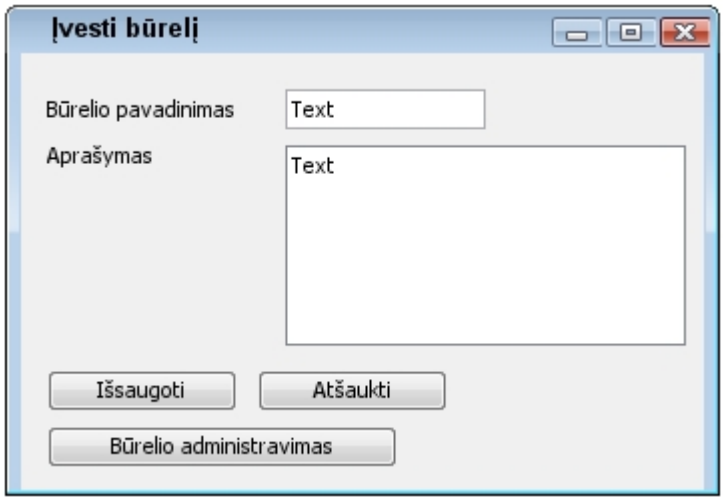


36 pav. IV eksperimento bandymo veiklos diagrama

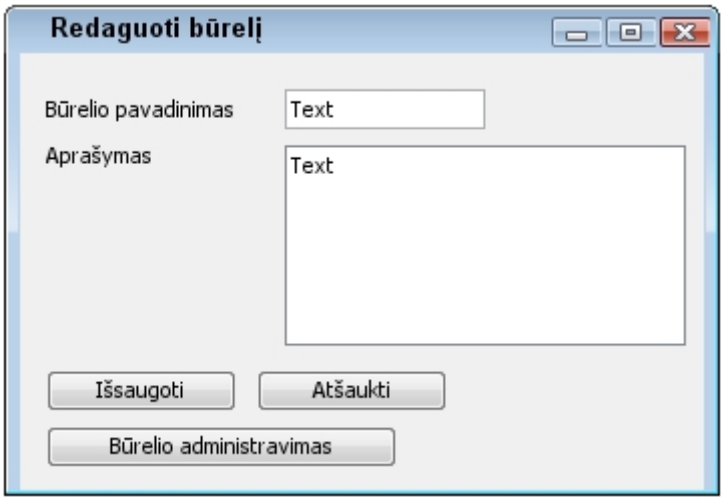
Mokinio administravimo, įvedimo ir redagavimo projektiniai langai pavaizduoti 21 - 23 (sugeneruoti – 28 - 30) šio priedo paveiksluose; Klasės administravimo, įvedimo ir redagavimo projektiniai langai pavaizduoti 24 - 26 paveiksluose (sugeneruoti – 31 - 33).



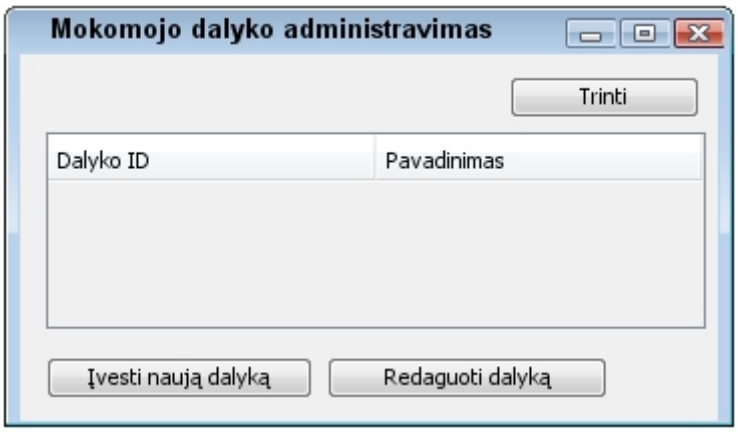
37 pav. Būrelių administravimo lango projektas



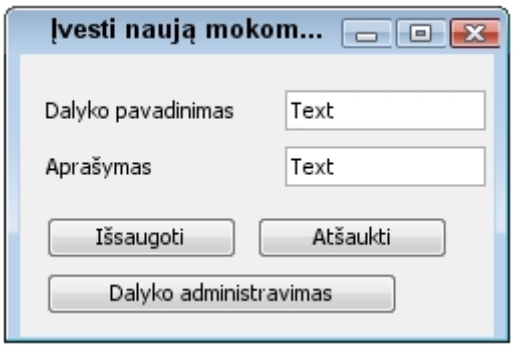
38 pav. Naujo būrelio įvedimo lango projektas



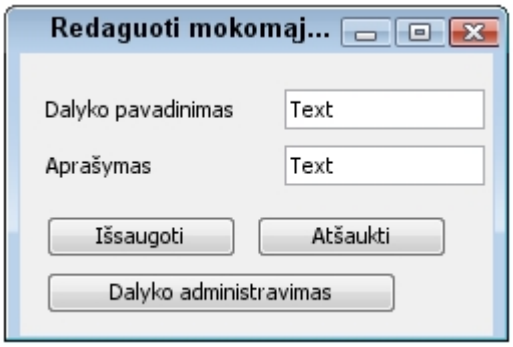
39 pav. Būrelio redagavimo lango projektas



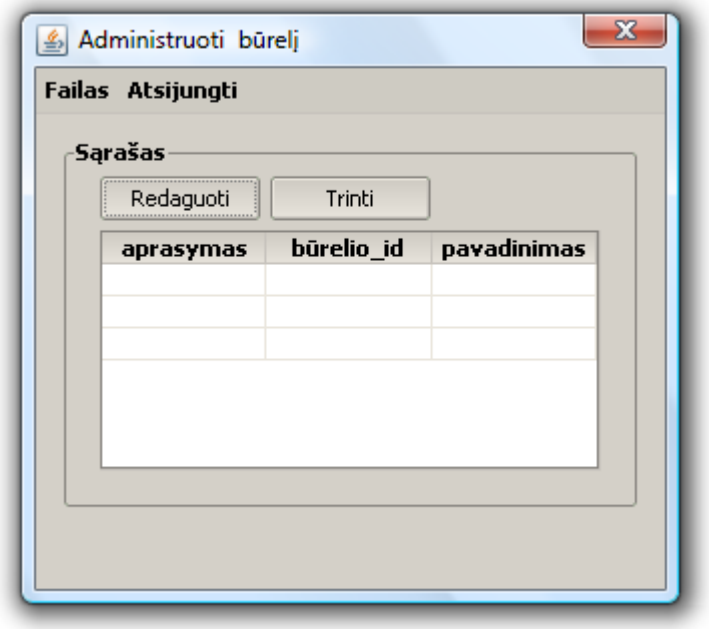
40 pav. Mokomųjų dalykų administravimo lango projektas



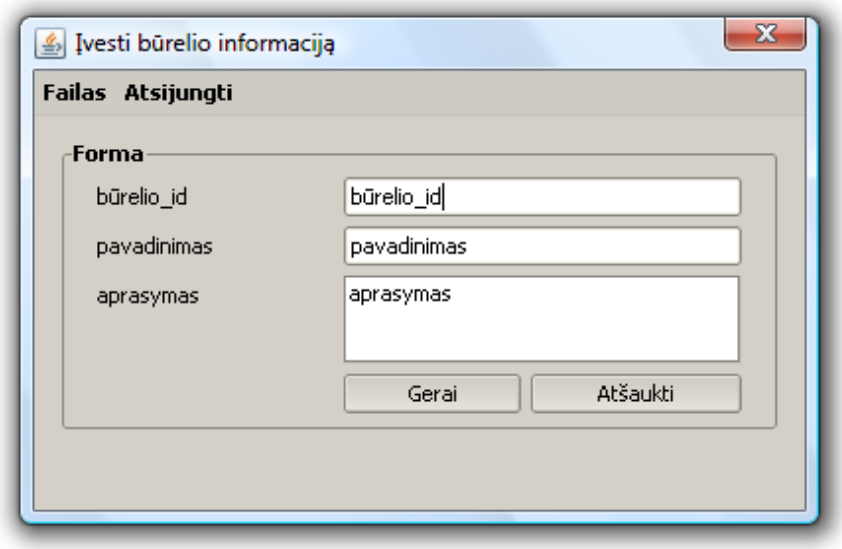
41 pav. Naujo mokomojo dalyko įvedimo lango projektas



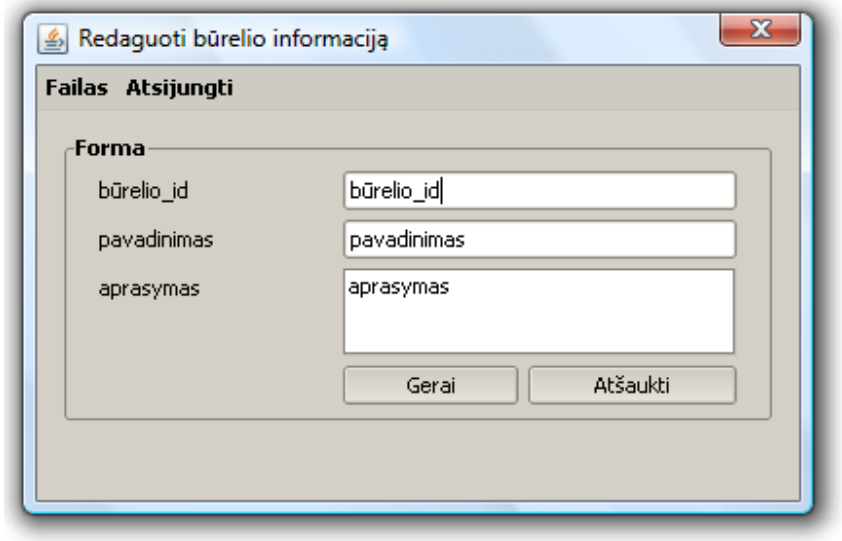
42 pav. Mokomojo dalyko redagavimo lango projektas



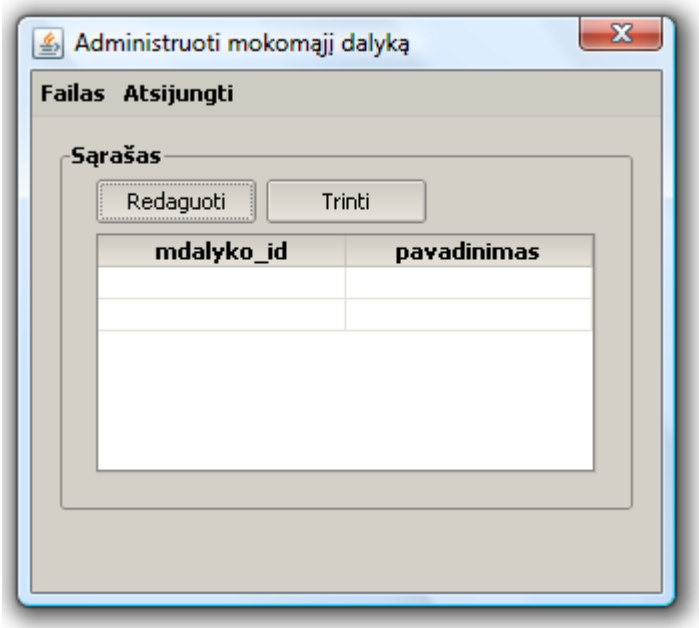
43 pav. Būrelių administravimo langas



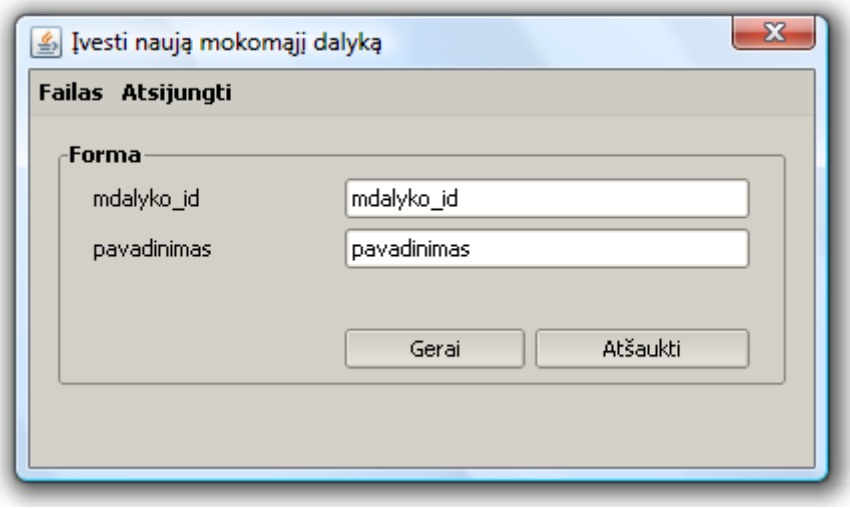
44 pav. Būrelio informacijos įvedimo langas



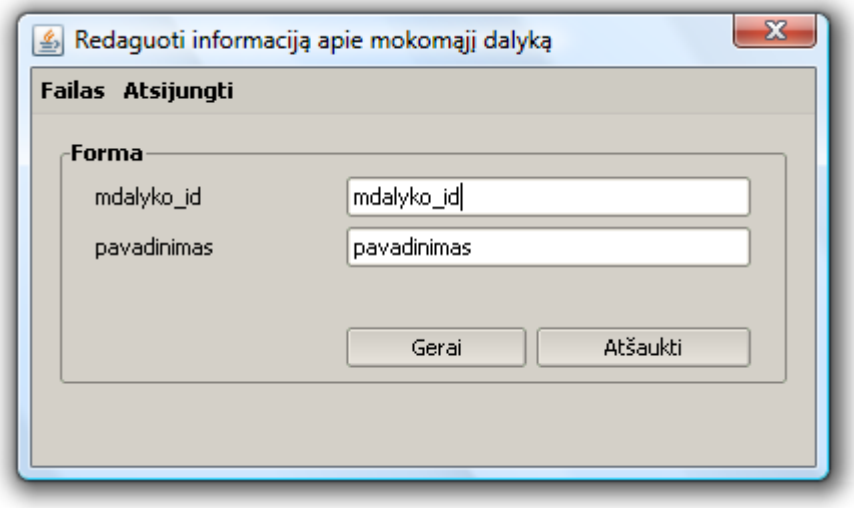
45 pav. Būrelio informacijos redagavimo langas



46 pav. Mokomųjų dalykų administravimo langas



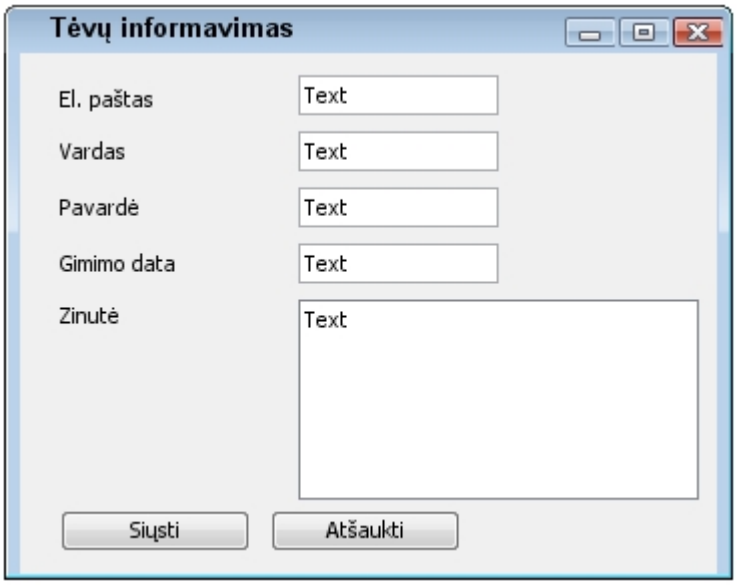
47 pav. Mokomojo dalyko informacijos įvedimo langas



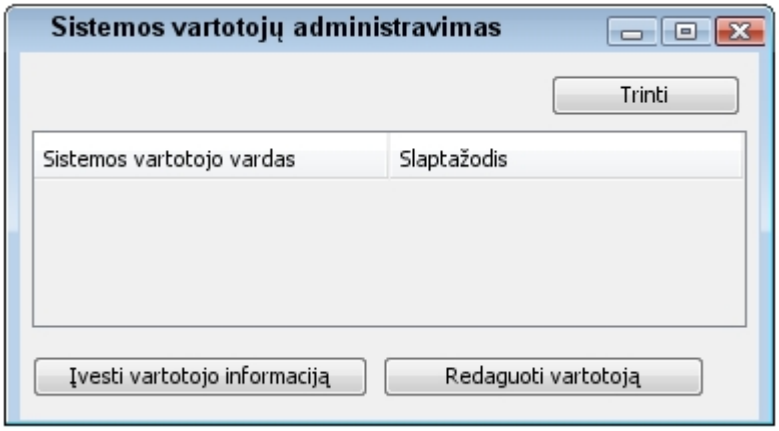
48 pav. Mokomojo dalyko informacijos redagavimo langas

4.5 Kai sistemos grafinė sąsaja kuriama su 4 panaudojimo atvejais, 1 klase ir 4 veiklomis

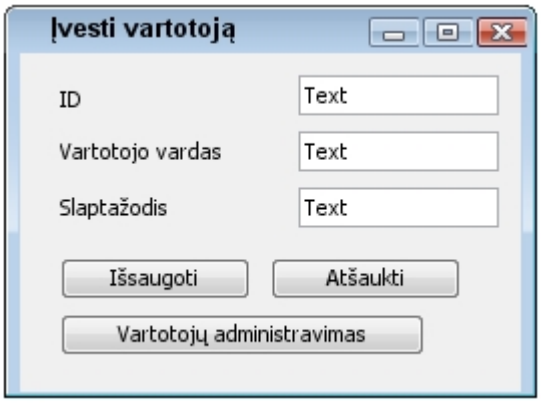
Diagramos, kurių informacija buvo naudojama grafinei vartotojo sąsajai generuoti pateiktos 89 - 91 paveiksluose.



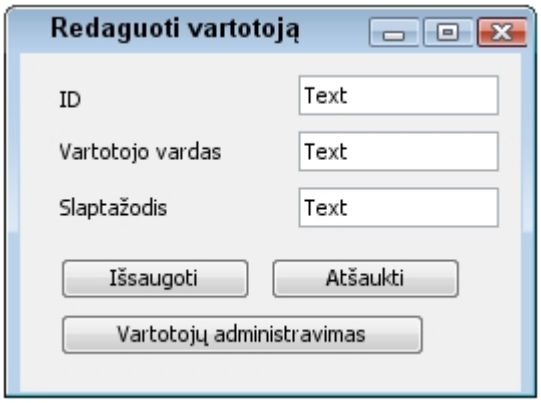
49 pav. Tėvų informavimo lango projektas



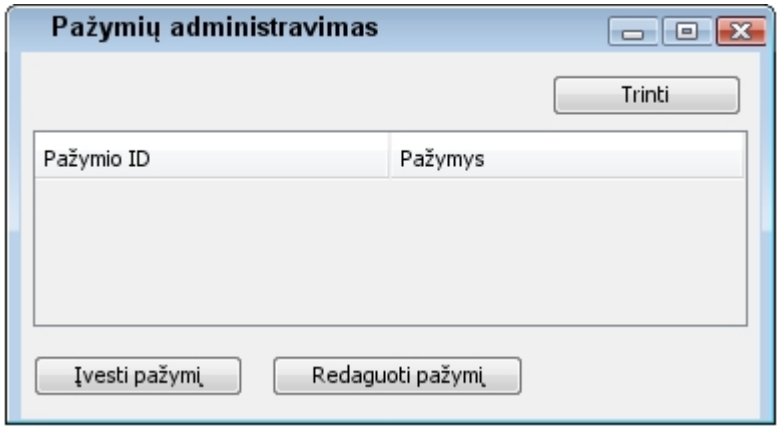
50 pav. Sistemos vartotojų administravimo lango projektas



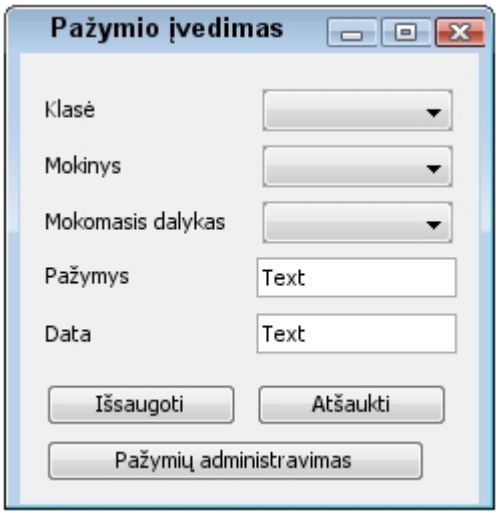
51 pav. Sistemos vartotojo įvedimo lango projektas



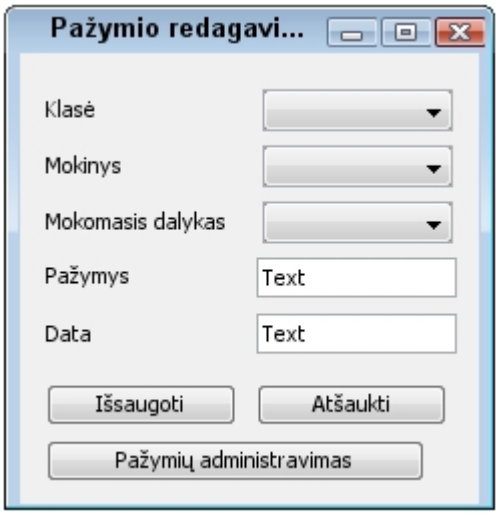
52 pav. Sistemos vartotojo redagavimo lango projektas



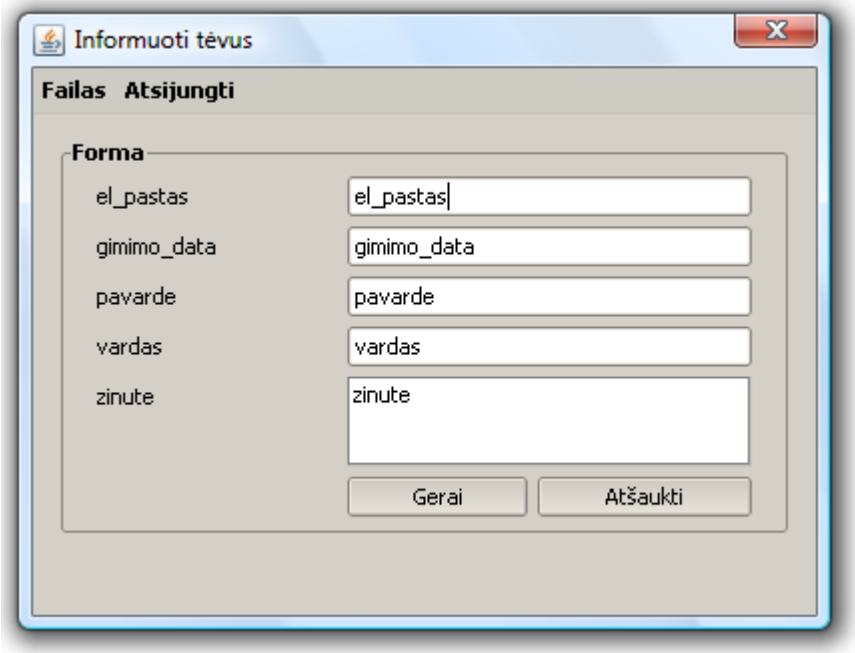
53 pav. Pažymių administravimo lango projektas



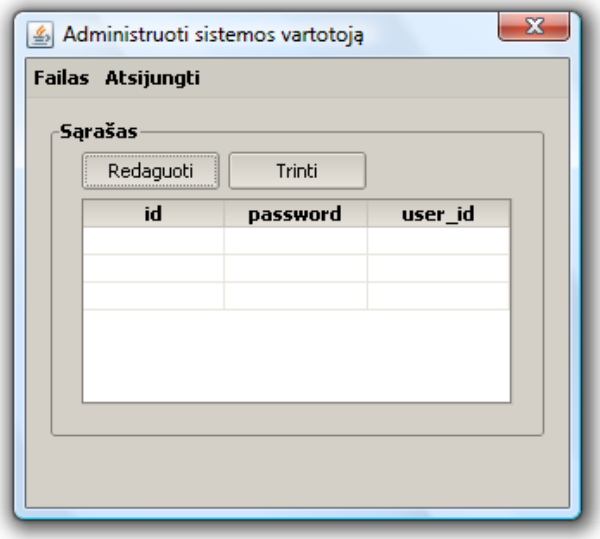
54 pav. Pažymių įvedimo lango projektas



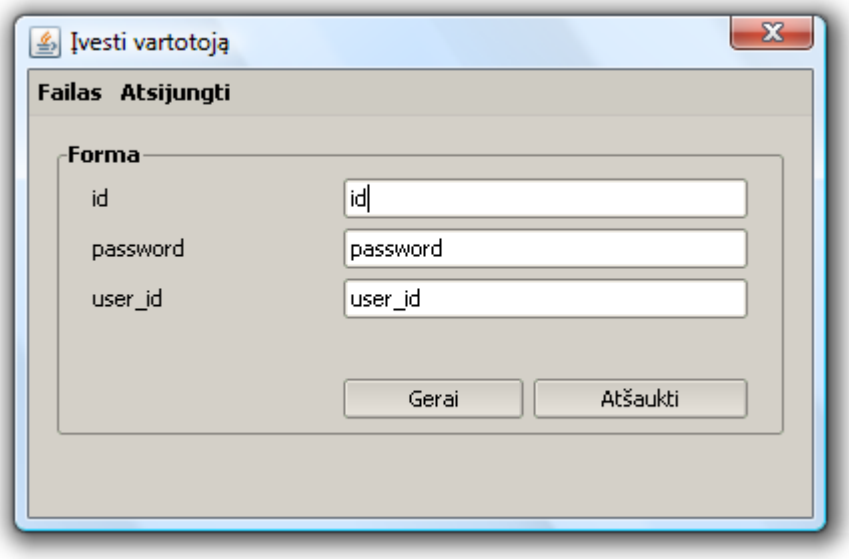
55 pav. Pažymių redagavimo lango projektas



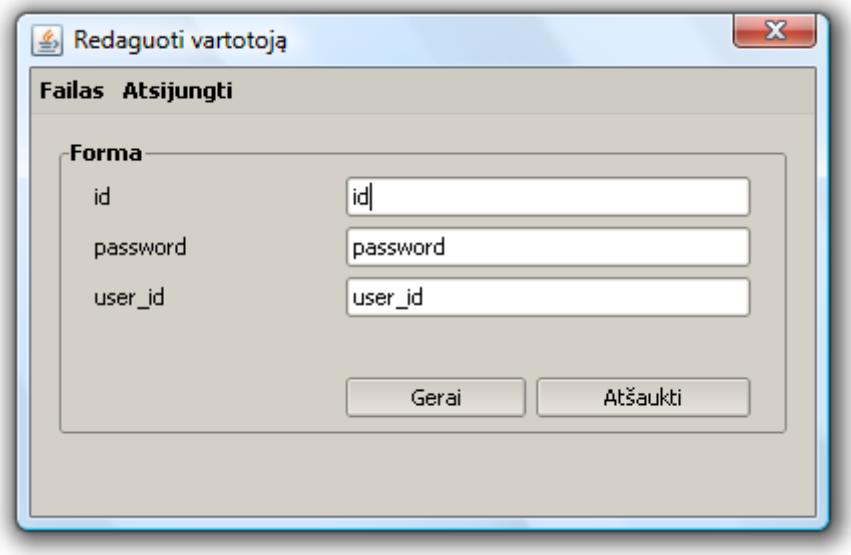
56 pav. Tėvų informavimo langas



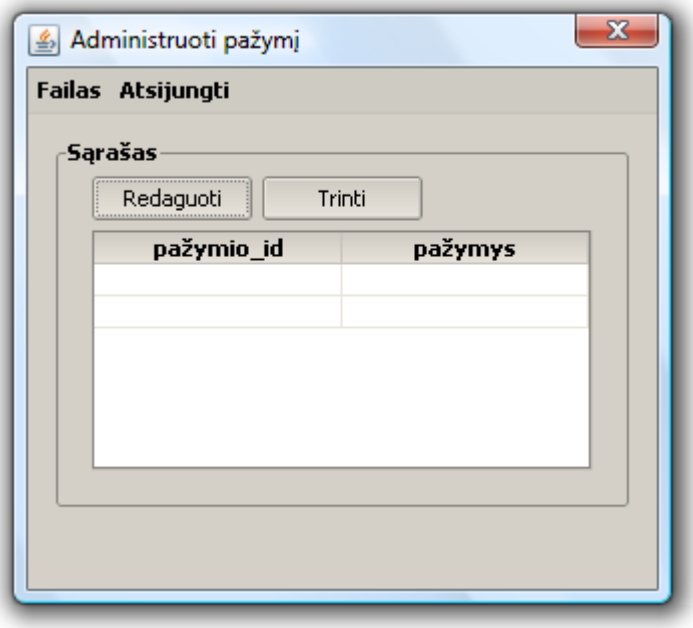
57 pav. Sistemos vartotojų administravimo langas



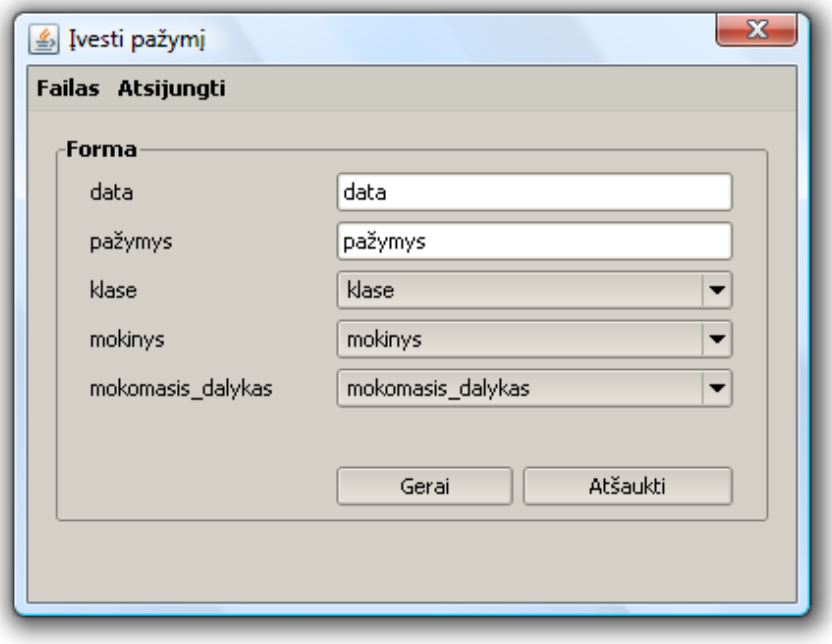
58 pav. Sistemos vartotojo įvedimo langas



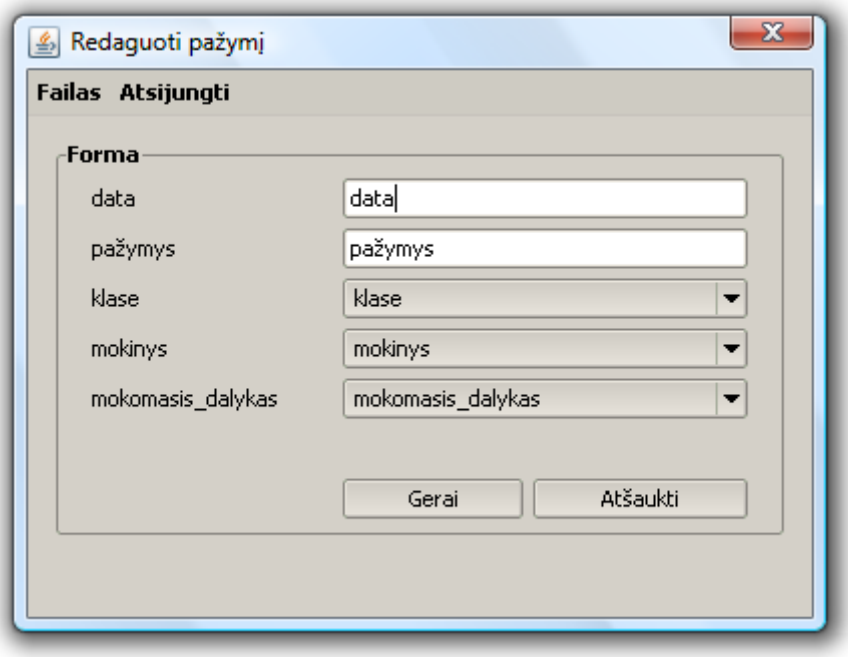
59 pav. Sistemos vartotojo redagavimo langas



60 pav. Pažymių administravimo langas



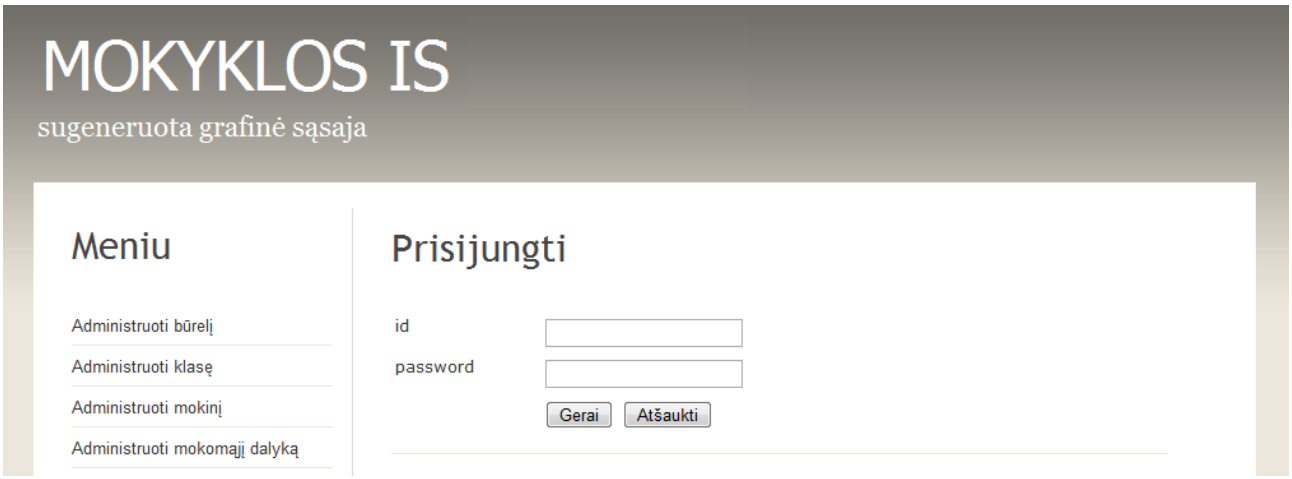
61 pav. Pažymio įvedimo langas



62 pav. Pažymio redagavimo langas

Prisijungimo lango projektas pavaizduotas – 29 paveiksle; Mokinio administravimo, įvedimo ir redagavimo projektiniai langai pavaizduoti 21 - 23 (sugeneruoti – 28 - 30) šio priedo paveiksluose; Klasės administravimo, įvedimo ir redagavimo projektiniai langai pavaizduoti 24 - 26 paveiksluose (sugeneruoti – 31 - 33); Būrelio administravimo, įvedimo ir redagavimo projektiniai langai pavaizduoti 37 - 39 paveiksluose (sugeneruoti – 43 - 45); Mokomojo dalyko administravimo, įvedimo ir redagavimo projektiniai langai pavaizduoti 33-35 paveiksluose (sugeneruoti – 46 - 48).

Priedas Nr. 3. Sugeneruoti internetinės grafinės vartotojo sąsajos langai eksperimentinei Mokyklos informacijos sistemai



1 pav. Prisijungimo langas

2 pav. Mokomųjų dalykų administravimo langas

3 pav. Mokomojo dalyko informacijos įvedimo langas

4 pav. Mokomojo dalyko informacijos redagavimo langas

5 pav. Mokinių administravimo langas

6 pav. Mokinio informacijos įvedimo langas

7 pav. Mokinio informacijos redagavimo langas

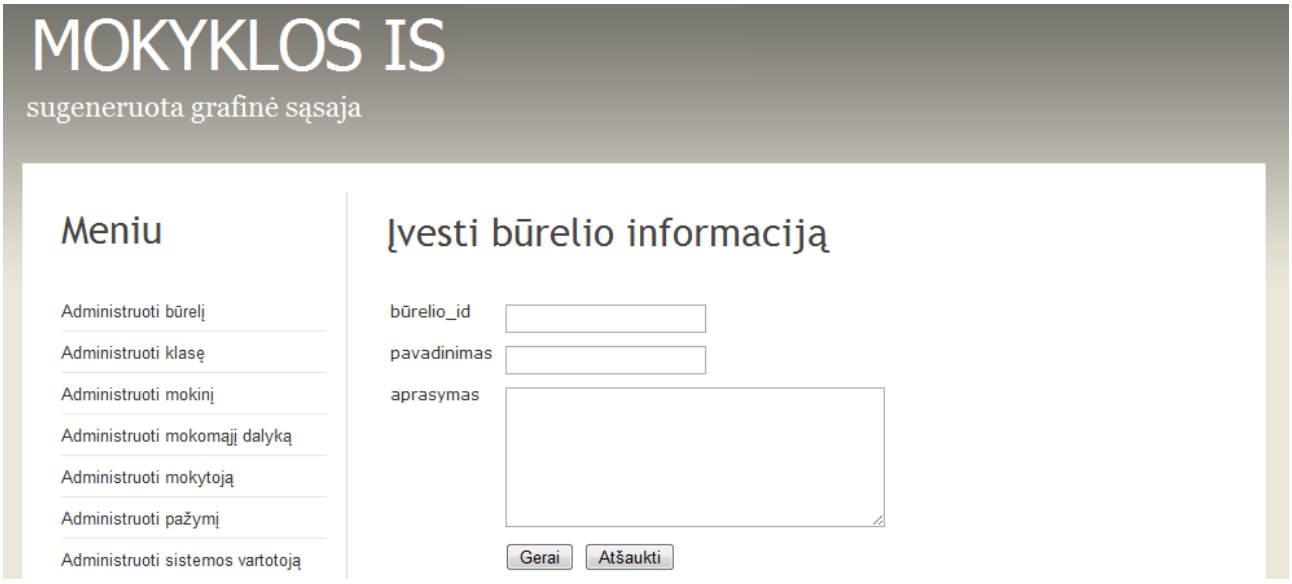
8 pav. Tėvų informavimo langas

9 pav. Sistemos vartotojų administravimo langas

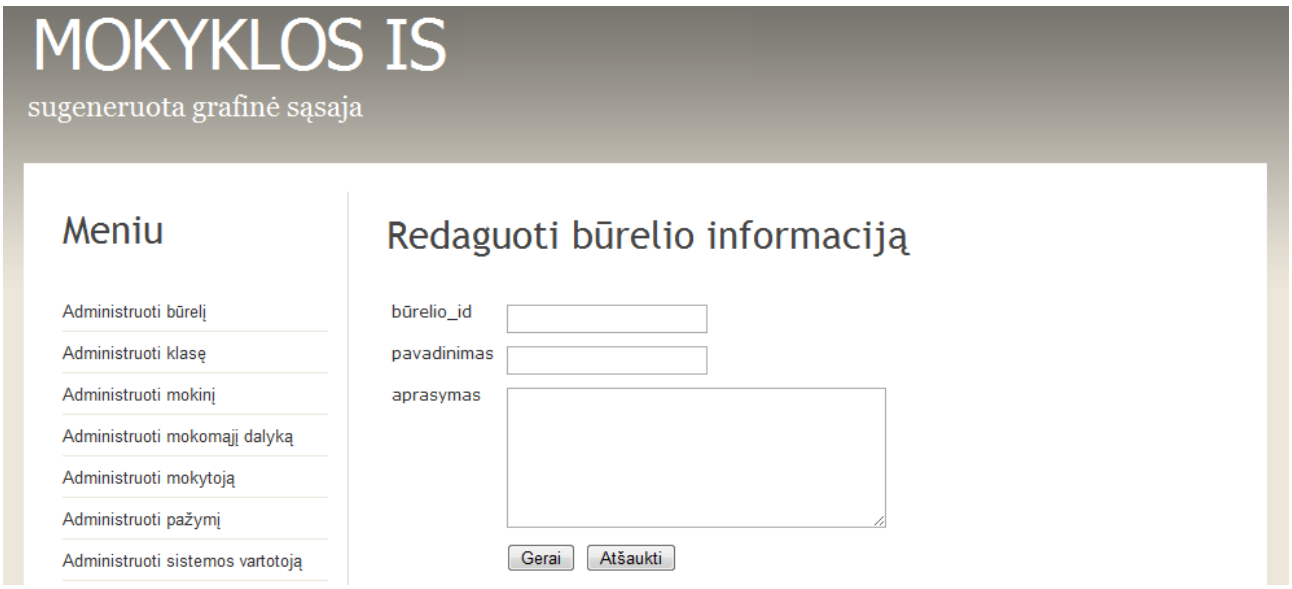
10 pav. Sistemos vartotojo įvedimo langas

11 pav. Sistemos vartotojo redagavimo langas

12 pav. Būrelių administravimo langas



13 pav. Būrelio informacijos įvedimo langas

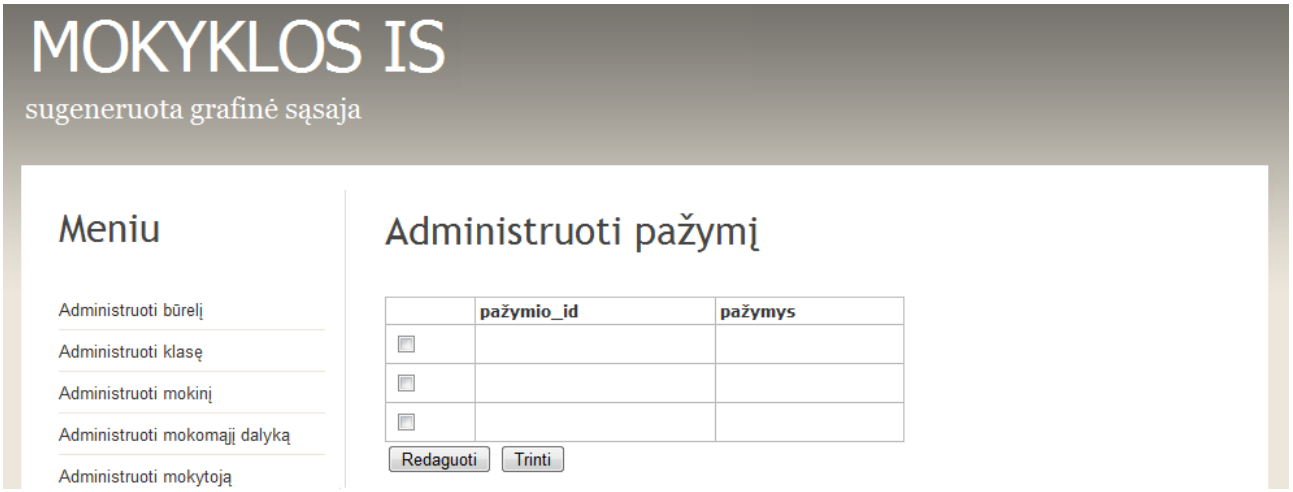


14 pav. Būrelio informacijos redagavimo langas

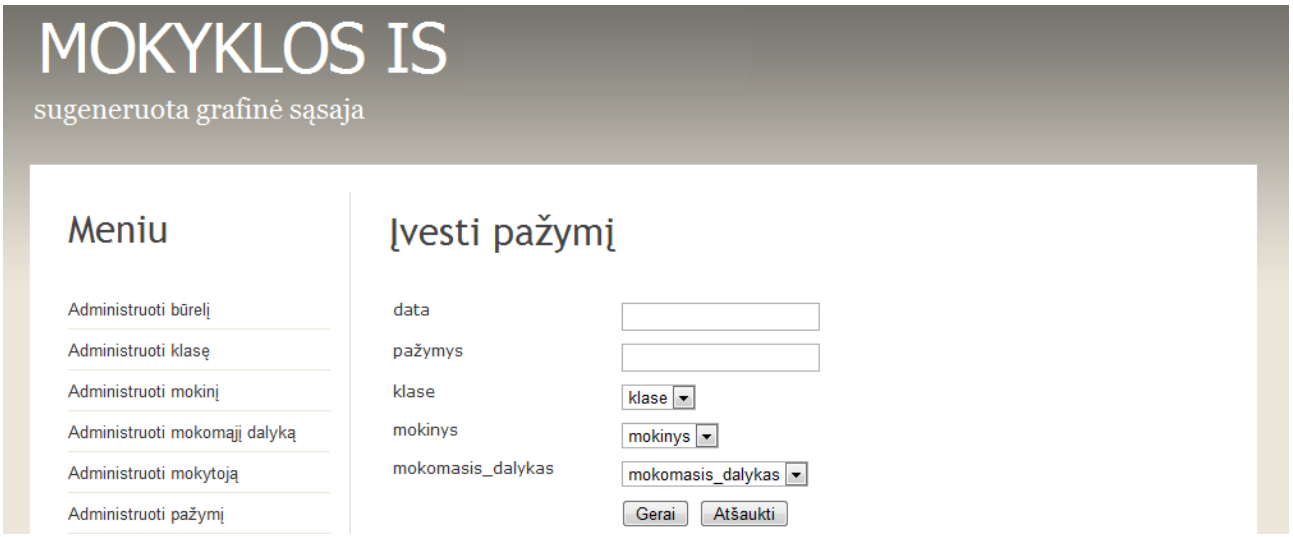
15 pav. Klasių administravimo langas

16 pav. Klasės informacijos įvedimo langas

17 pav. Klasės informacijos redagavimo langas



18 pav. Pažymių administravimo langas



19 pav. Pažymio įvedimo langas

20 pav. Pažymio redagavimo langas

Priedas Nr. 4. Programinio kodo pavyzdžiai

Java programinis kodas „Mokinio administravimo“ langui:

```
import info.clearthought.layout.TableLayout;
import javax.swing.*;

import java.awt.FlowLayout;
import java.awt.Frame;
import java.util.Iterator;

/**
 * Used tamplate variables: columnsCount
 *
            windowType
 *
            variables
 *
            initGUIText
 *
 */
public class NewJDialog extends javax.swing.JDialog {

            private int _elementsCount = 7;
            private int _rowHeight = 20;
            private int _columnWidth = 80;
            private JMenuBar jMainMenu;
            private JPanel jFormPanel;
            private JPanel jListPanel;
            private JButton jButton1;
            private JButton jButton2;
            private JButton jButton3;
            private JButton jButton4;
            private int _position;
            private double[] _rows;
            private double[] _columns;
            private JMenu jFileMenu;
            private JMenu jExitMenu;
            private JMenuItem jExit;
            private int _windowType = 3;
            private ButtonGroup buttonGroup;
            private JPanel jButtonsPanel;
            private JTextField jTextField0;

            private JLabel jLabel0;

            private JCheckBox jCheckBox1;

            private JLabel jLabel1;

            private JTextField jTextField2;

            private JLabel jLabel2;

            private JTextField jTextField3;

            private JLabel jLabel3;

            private JTextField jTextField4;

            private JLabel jLabel4;

            private JTextField jTextField5;

            private JLabel jLabel5;

            private JTextField jTextField6;

            private JLabel jLabel6;
```

```
            /**
            * Auto-generated main method to display this JDialog
            */
            public static void main(String[] args) {
                        JFrame frame = new JFrame();
                        NewJDialog inst = new NewJDialog(frame);
                        inst.setVisible(true);
            }

            public NewJDialog(JFrame frame) {
                        super(frame);
                        initGUI();
            }

            private void initGUI() {
                        try {
                                    double[] formColumns = new double[]{ 7.0,
TableLayout.FILL, TableLayout.FILL, 22.0, 7.0};

                                    if (_windowType == 3) {
                                                formColumns = new double[]{ 7.0,
TableLayout.FILL, TableLayout.FILL, 10.0,

            TableLayout.FILL, TableLayout.FILL, 22.0, 7.0 };
                                    }

                                    TableLayout thisLayout = new TableLayout(new
double[][] {
                                                                { 7.0, TableLayout.FILL,
TableLayout.FILL,

            TableLayout.FILL, TableLayout.FILL, 7.0 },

            formColumns });
                                    thisLayout.setHGap(5);
                                    thisLayout.setVGap(5);
                                    getContentPane().setLayout(thisLayout);
                                    initFormSize();
                                    //Menu
                                    {
                                                jMainMenu = new JMenuBar();
                                                setJMenuBar(jMainMenu);

                                                jFileMenu = new JMenu("Failas");

            jFileMenu.getAccessibleContext().setAccessibleDescription(
                                    "Failas");
                                                jMainMenu.add(jFileMenu);

                                                jExitMenu = new JMenu("Atsijungti");

            jExitMenu.getAccessibleContext().setAccessibleDescription(
                                    "Atsijungti");
                                                jMainMenu.add(jExitMenu);

                                                jExit = new JMenuItem("Baigti
darbą");

            jExit.getAccessibleContext().setAccessibleDescription(
                                    "Baigti darbą");
                                                jExit.addActionListener(new
GenCloseDialog(this));
                                                jExitMenu.add(jExit);
                                    }

                                    //generate form panel
                                    {
                                                jFormPanel = new JPanel();
```

```
                                                    TableLayout jPanel1Layout = new
TableLayout(new double[][] {
                                                                        { 10.0,
120.0, 90.0, 107.0 },
                                                                        _rows });
                                                    jPanel1Layout.setHGap(5);
                                                    jPanel1Layout.setVGap(5);
                                                    jFormPanel.setLayout(jPanel1Layout);

            jFormPanel.setBorder(BorderFactory.createTitledBorder("Forma"));
                                                    {
                                                                jButton1 = new
JButton();
                                                                jFormPanel.add(jButton1,
"2, "+(_rows.length-1));

            jButton1.setText("Išsaugoti");
                                                    }
                                                    {
                                                                jButton4 = new
JButton();
                                                                jFormPanel.add(jButton4,
"3, "+(_rows.length-1));

            jButton4.setText("Atšaukti");

            jButton4.addActionListener(new GenCloseDialog(this));
                                                    }
                                                    {
                                                                buttonGroup = new
ButtonGroup();
                                                    }
                                                    //All rows between label and button
are filled with window elements
                                        }
                                        {
                                                    jButtonsPanel = new JPanel();
                                                    jButtonsPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 5, 0));
                                        }

                                        //generate list panel
                                        {
                                                    jListPanel = new JPanel();
                                                    TableLayout jPanel2Layout = new
TableLayout(new double[][] {
                                                                        _columns,
                                                                        { 22.0,
30.0, 40.0,

            40.0, 5.0 } });
                                                    jPanel2Layout.setHGap(5);
                                                    jPanel2Layout.setVGap(5);
                                                    jListPanel.setLayout(jPanel2Layout);

            jListPanel.setBorder(BorderFactory.createTitledBorder("Sąrašas"));
                                                    {
                                                                jButton3 = new
JButton();
                                                                jListPanel.add(jButton3,
"1, 0");

            jButton3.setText("Redaguoti");
                                                    }
                                                    {
                                                                jButton2 = new
JButton();
                                                                jListPanel.add(jButton2,
"2, 0");
```

```
            jButton2.setText("Trinti");
                                    }
                                }

                                if (_windowType == GlobalStatus.SHOW_FORM) {
                                    getContentPane().add(jFormPanel, "1,
1, 4, 2");
                                    getContentPane().add(jButtonsPanel,
"1, 3, 4, 3");
                                }

                                if (_windowType == GlobalStatus.SHOW_LIST) {
                                    getContentPane().add(jListPanel, "1,
1, 4, 2");
                                    getContentPane().add(jButtonsPanel,
"1, 3, 4, 3");
                                }

                                if (_windowType == GlobalStatus.SHOW_FORM_LIST)
{
                                    getContentPane().add(jFormPanel, "1,
1, 4, 2");
                                    getContentPane().add(jListPanel, "1,
4, 4, 5");
                                    getContentPane().add(jButtonsPanel,
"1, 6, 4, 6");
                                }

                                if (_windowType == GlobalStatus.SHOW_NONE) {
                                    setSize(400,300);
                                } else {
                                    pack();
                                }


            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

{
                                jTextField0 = new JTextField();
                                jLabel0 = new JLabel();
                                jLabel0.setText("el_pastas");
                                jTextField0.setText("el_pastas");
                                jFormPanel.add(jLabel0, "1, 1");
                                jFormPanel.add(jTextField0, "2, 1 ,3 , 1 ");
}


{
                                jCheckBox1 = new JCheckBox();
                                jLabel1 = new JLabel();
                                jLabel1.setText("lytis");
                                jCheckBox1.setText("lytis");
                                jFormPanel.add(jCheckBox1, "1, 2");
                                jFormPanel.add(jLabel1, "2, 2 ,3 , 2 ");
}


{
                                jTextField2 = new JTextField();
                                jLabel2 = new JLabel();
                                jLabel2.setText("adresas");
                                jTextField2.setText("adresas");
                                jFormPanel.add(jLabel2, "1, 3");
                                jFormPanel.add(jTextField2, "2, 3 ,3 , 3 ");
}


{
                                jTextField3 = new JTextField();
```

```
                                    jLabel3 = new JLabel();
                                    jLabel3.setText("gimimo_data");
                                    jTextField3.setText("gimimo_data");
                                    jFormPanel.add(jLabel3, "1, 4");
                                    jFormPanel.add(jTextField3, "2, 4 ,3 , 4 ");
}


{
                                    jTextField4 = new JTextField();
                                    jLabel4 = new JLabel();
                                    jLabel4.setText("pavarde");
                                    jTextField4.setText("pavarde");
                                    jFormPanel.add(jLabel4, "1, 5");
                                    jFormPanel.add(jTextField4, "2, 5 ,3 , 5 ");
}


{
                                    jTextField5 = new JTextField();
                                    jLabel5 = new JLabel();
                                    jLabel5.setText("vardas");
                                    jTextField5.setText("vardas");
                                    jFormPanel.add(jLabel5, "1, 6");
                                    jFormPanel.add(jTextField5, "2, 6 ,3 , 6 ");
}


{
                                    jTextField6 = new JTextField();
                                    jLabel6 = new JLabel();
                                    jLabel6.setText("mokinio_id");
                                    jTextField6.setText("mokinio_id");
                                    jFormPanel.add(jLabel6, "1, 7");
                                    jFormPanel.add(jTextField6, "2, 7 ,3 , 7 ");
}

DefaultTableModel model = new DefaultTableModel();
String[] columnNames = new String[]{
"el_pastas","lytis","adresas","gimimo_data","pavarde","vardas","mokinio_id",};
int dataSize = 10;
int index = 0;
Object[][] data = new Object[_elementsCount][dataSize];

for (int i = 0; i < _elementsCount; i++) {
for (int j = i; j < dataSize; j++) {
data[i][j] = "";
}
}

model.setDataVector(data, columnNames);

JTable table = new JTable(model);
table.setFillsViewportHeight(true);
JScrollPane scrollTable = new JScrollPane( table );

jListPanel.add(scrollTable,"1, 1, "+(_columns.length-1)+", 3");

                        } catch (Exception e) {
                                    e.printStackTrace();
                        }
            }

            private void initFormSize() {
                        int add = 2;
                        int columnAdd = 4;
                        int columnsCount = _elementsCount < columnAdd ? columnAdd :
_elementsCount;

                        _rows = new double[_elementsCount+add];
```

```
                        _columns = new double[columnsCount];

                        for (int i = 0; i < _elementsCount+add; i++) {
                                    _rows[i] = _rowHeight;
                        }

                        _columns[0] = 10;
                        for (int i = 1; i < columnsCount; i++) {
                                    _columns[i] = _columnWidth;
                        }

                        _position = 0;
            }

}
```

Java programinis kodas „Naujo mokinio informacijos įvedimo“ langui:

```
import info.clearthought.layout.TableLayout;
import javax.swing.*;

import java.awt.FlowLayout;
import java.awt.Frame;
import java.util.Iterator;

/**
 * Used tamplate variables: columnsCount
 *
            windowType
 *
            variables
 *
            initGUIText
 *
 */
public class NewJDialog extends javax.swing.JDialog {

            private int _elementsCount = 7;
            private int _rowHeight = 20;
            private int _columnWidth = 80;
            private JMenuBar jMainMenu;
            private JPanel jFormPanel;
            private JPanel jListPanel;
            private JButton jButton1;
            private JButton jButton2;
            private JButton jButton3;
            private JButton jButton4;
            private int _position;
            private double[] _rows;
            private double[] _columns;
            private JMenu jFileMenu;
            private JMenu jExitMenu;
            private JMenuItem jExit;
            private int _windowType = 3;
            private ButtonGroup buttonGroup;
            private JPanel jButtonsPanel;
            private JTextField jTextField0;
            private JLabel jLabel0;
            private JTextField jTextField1;
            private JLabel jLabel1;
            private JCheckBox jCheckBox2;
            private JLabel jLabel2;
            private JTextField jTextField3;
            private JLabel jLabel3;
            private JTextField jTextField4;
            private JLabel jLabel4;
            private JTextField jTextField5;
            private JLabel jLabel5;
            private JTextField jTextField6;
            private JLabel jLabel6;
```

```
            /**
            * Auto-generated main method to display this JDialog
            */
            public static void main(String[] args) {
                        JFrame frame = new JFrame();
                        NewJDialog inst = new NewJDialog(frame);
                        inst.setVisible(true);
            }

            public NewJDialog(JFrame frame) {
                        super(frame);
                        initGUI();
            }

            private void initGUI() {
                        try {
                                    double[] formColumns = new double[]{ 7.0,
TableLayout.FILL, TableLayout.FILL, 22.0, 7.0};

                                    if (_windowType == 3) {
                                                formColumns = new double[]{ 7.0,
TableLayout.FILL, TableLayout.FILL, 10.0,

            TableLayout.FILL, TableLayout.FILL, 22.0, 7.0 };
                                    }

                                    TableLayout thisLayout = new TableLayout(new
double[][] {
                                                                { 7.0, TableLayout.FILL,
TableLayout.FILL,

            TableLayout.FILL, TableLayout.FILL, 7.0 },

            formColumns });
                                    thisLayout.setHGap(5);
                                    thisLayout.setVGap(5);
                                    getContentPane().setLayout(thisLayout);
                                    initFormSize();
                                    //Menu
                                    {
                                                jMainMenu = new JMenuBar();
                                                setJMenuBar(jMainMenu);

                                                jFileMenu = new JMenu("Failas");

            jFileMenu.getAccessibleContext().setAccessibleDescription(
                                    "Failas");
                                                jMainMenu.add(jFileMenu);

                                                jExitMenu = new JMenu("Atsijungti");

            jExitMenu.getAccessibleContext().setAccessibleDescription(
                                    "Atsijungti");
                                                jMainMenu.add(jExitMenu);

                                                jExit = new JMenuItem("Baigti
darbą");

            jExit.getAccessibleContext().setAccessibleDescription(
                                    "Baigti darbą");
                                                jExit.addActionListener(new
GenCloseDialog(this));
                                                jExitMenu.add(jExit);
                                    }

                                    //generate form panel
                                    {
                                                jFormPanel = new JPanel();
```

```
                                                    TableLayout jPanel1Layout = new
TableLayout(new double[][] {
                                                                { 10.0,
120.0, 90.0, 107.0 },
                                                                _rows });
                                                    jPanel1Layout.setHGap(5);
                                                    jPanel1Layout.setVGap(5);
                                                    jFormPanel.setLayout(jPanel1Layout);

            jFormPanel.setBorder(BorderFactory.createTitledBorder("Forma"));
                                                    {
                                                                jButton1 = new
JButton();
                                                                jFormPanel.add(jButton1,
"2, "+(_rows.length-1));

            jButton1.setText("Išsaugoti");
                                                    }
                                                    {
                                                                jButton4 = new
JButton();
                                                                jFormPanel.add(jButton4,
"3, "+(_rows.length-1));

            jButton4.setText("Atšaukti");

            jButton4.addActionListener(new GenCloseDialog(this));
                                                    }
                                                    {
                                                                buttonGroup = new
ButtonGroup();
                                                    }
                                                    //All rows between label and button
are filled with window elements
                                        }
                                        {
                                                    jButtonsPanel = new JPanel();
                                                    jButtonsPanel.setLayout(new
FlowLayout(FlowLayout.LEFT, 5, 0));
                                        }

                                        //generate list panel
                                        {
                                                    jListPanel = new JPanel();
                                                    TableLayout jPanel2Layout = new
TableLayout(new double[][] {
                                                                _columns,
                                                                { 22.0,
30.0, 40.0,

            40.0, 5.0 } });
                                                    jPanel2Layout.setHGap(5);
                                                    jPanel2Layout.setVGap(5);
                                                    jListPanel.setLayout(jPanel2Layout);

            jListPanel.setBorder(BorderFactory.createTitledBorder("Sąrašas"));
                                                    {
                                                                jButton3 = new
JButton();
                                                                jListPanel.add(jButton3,
"1, 0");

            jButton3.setText("Redaguoti");
                                                    }
                                                    {
                                                                jButton2 = new
JButton();
                                                                jListPanel.add(jButton2,
"2, 0");
```

```
            jButton2.setText("Trinti");
                                    }
                                }

                                if (_windowType == GlobalStatus.SHOW_FORM) {
                                    getContentPane().add(jFormPanel, "1,
1, 4, 2");
                                    getContentPane().add(jButtonsPanel,
"1, 3, 4, 3");
                                }

                                if (_windowType == GlobalStatus.SHOW_LIST) {
                                    getContentPane().add(jListPanel, "1,
1, 4, 2");
                                    getContentPane().add(jButtonsPanel,
"1, 3, 4, 3");
                                }

                                if (_windowType == GlobalStatus.SHOW_FORM_LIST)
{
                                    getContentPane().add(jFormPanel, "1,
1, 4, 2");
                                    getContentPane().add(jListPanel, "1,
4, 4, 5");
                                    getContentPane().add(jButtonsPanel,
"1, 6, 4, 6");
                                }

                                if (_windowType == GlobalStatus.SHOW_NONE) {
                                    setSize(400,300);
                                } else {
                                    pack();
                                }


            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

{
                                jTextField0 = new JTextField();
                                jLabel0 = new JLabel();
                                jLabel0.setText("gimimo_data");
                                jTextField0.setText("gimimo_data");
                                jFormPanel.add(jLabel0, "1, 1");
                                jFormPanel.add(jTextField0, "2, 1 ,3 , 1 ");
}


{
                                jTextField1 = new JTextField();
                                jLabel1 = new JLabel();
                                jLabel1.setText("vardas");
                                jTextField1.setText("vardas");
                                jFormPanel.add(jLabel1, "1, 2");
                                jFormPanel.add(jTextField1, "2, 2 ,3 , 2 ");
}


{
                                jCheckBox2 = new JCheckBox();
                                jLabel2 = new JLabel();
                                jLabel2.setText("lytis");
                                jCheckBox2.setText("lytis");
                                jFormPanel.add(jCheckBox2, "1, 3");
                                jFormPanel.add(jLabel2, "2, 3 ,3 , 3 ");
}


{
                                jTextField3 = new JTextField();
```

```
                                    jLabel3 = new JLabel();
                                    jLabel3.setText("mokinio_id");
                                    jTextField3.setText("mokinio_id");
                                    jFormPanel.add(jLabel3, "1, 4");
                                    jFormPanel.add(jTextField3, "2, 4 ,3 , 4 ");
}


{
                                    jTextField4 = new JTextField();
                                    jLabel4 = new JLabel();
                                    jLabel4.setText("pavarde");
                                    jTextField4.setText("pavarde");
                                    jFormPanel.add(jLabel4, "1, 5");
                                    jFormPanel.add(jTextField4, "2, 5 ,3 , 5 ");
}


{
                                    jTextField5 = new JTextField();
                                    jLabel5 = new JLabel();
                                    jLabel5.setText("adresas");
                                    jTextField5.setText("adresas");
                                    jFormPanel.add(jLabel5, "1, 6");
                                    jFormPanel.add(jTextField5, "2, 6 ,3 , 6 ");
}


{
                                    jTextField6 = new JTextField();
                                    jLabel6 = new JLabel();
                                    jLabel6.setText("el_pastas");
                                    jTextField6.setText("el_pastas");
                                    jFormPanel.add(jLabel6, "1, 7");
                                    jFormPanel.add(jTextField6, "2, 7 ,3 , 7 ");
}

DefaultTableModel model = new DefaultTableModel();
String[] columnNames = new String[]{
"gimimo_data","vardas","lytis","mokinio_id","pavarde","adresas","el_pastas",};
int dataSize = 10;
int index = 0;
Object[][] data = new Object[_elementsCount][dataSize];

for (int i = 0; i < _elementsCount; i++) {
for (int j = i; j < dataSize; j++) {
data[i][j] = "";
}
}

model.setDataVector(data, columnNames);

JTable table = new JTable(model);
table.setFillsViewportHeight(true);
JScrollPane scrollTable = new JScrollPane( table );

jListPanel.add(scrollTable,"1, 1, "+(_columns.length-1)+", 3");

                        } catch (Exception e) {
                                    e.printStackTrace();
                        }
            }

            private void initFormSize() {
                        int add = 2;
                        int columnAdd = 4;
                        int columnsCount = _elementsCount < columnAdd ? columnAdd :
_elementsCount;

                        _rows = new double[_elementsCount+add];
```

```
                        _columns = new double[columnsCount];

                        for (int i = 0; i < _elementsCount+add; i++) {
                                    _rows[i] = _rowHeight;
                        }

                        _columns[0] = 10;
                        for (int i = 1; i < columnsCount; i++) {
                                    _columns[i] = _columnWidth;
                        }

                        _position = 0;
            }

}
```

HTML programinis kodas „Mokinio administravimo“ langui:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Mokyklos IS</title>
<meta name="keywords" content="" />
<meta name="Hanging" content="" />
<link href="styles/default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<div id="wrapper">
  <div id="header-wrapper">
    <div id="header">
      <div id="logo">
        <h1><a href="#"><span>Mokyklos IS</span></a></h1>
        <p>sugeneruota grafinė sąsaja</p>
      </div>
    </div>
  </div>
  <div id="page">
  <div id="page-bgtop">
   <div id="page-bgbtm">
   <div id="sidebar1" class="sidebar">
      <ul>
        <li>
          <h2>Meniu</h2>
          <ul>
                        <li><a href="Administruoti__bureli.html">Administruoti
būrelį</a></li>
            <li><a href="Administruoti_klase.html">Administruoti klasę</a></li>
            <li><a href="Administruoti_mokini.html">Administruoti mokinį</a></li>
            <li><a href="Administruoti_mokomaji_dalyka.html">Administruoti mokomąjį
dalyką</a></li>
            <li><a href="Administruoti_mokomaji_dalyka.html">Administruoti mokomąjį
dalyką</a></li>
            <li><a href="Administruoti_mokytoja.html">Administruoti
mokytoją</a></li>
            <li><a href="Administruoti_pazymi.html">Administruoti pažymį</a></li>
            <li><a href="Administruoti_sistemos_vartotoja.html">Administruoti
sistemos vartotoją</a></li>
            <li><a href="Informuoti_tevus.html">Informuoti tėvus</a></li>
            <li><a href="Pagrindinio_lango_valdymas.html">Pagrindinio lango
valdymas</a></li>
            <li><a href="Prisijungti.html">Prisijungti</a></li>
            <li><a href="Redaguoti_burelio_informacija.html">Redaguoti būrelio
informaciją</a></li>
            <li><a href="Redaguoti_informacija_apie_mokomaji_dalyka.html">Redaguoti
informaciją apie mokomąjį dalyką</a></li>
            <li><a href="Redaguoti_klases_informacija.html">Redaguoti klasės
informaciją</a></li>
```

```
            <li><a href="Redaguoti_mokinio_informacija.html">Redaguoti mokinio
informaciją</a></li>
            <li><a href="Redaguoti_mokytojo_informacija.html">Redaguoti mokytojo
informaciją</a></li>
            <li><a href="Redaguoti_pazymi.html">Redaguoti pažymį</a></li>
            <li><a href="Redaguoti_vartotoja.html">Redaguoti vartotoją</a></li>
            <li><a href="test.html">test</a></li>
            <li><a href="Ivesti_burelio_informacija.html">Įvesti būrelio
informaciją</a></li>
            <li><a href="Ivesti_mokytojo_informacija.html">Įvesti mokytojo
informaciją</a></li>
            <li><a href="Ivesti_naujo_mokinio_informacija.html">Įvesti naujo
mokinio informaciją</a></li>
            <li><a href="Ivesti_naujos_klases_informacija.html">Įvesti naujos
klasės informaciją</a></li>
            <li><a href="Ivesti_nauja_mokomaji_dalyka.html">Įvesti naują mokomąjį
dalyką</a></li>
            <li><a href="Ivesti_pazymi.html">Įvesti pažymį</a></li>
            <li><a href="Ivesti_vartotoja.html">Įvesti vartotoją</a></li>

</ul>

      </ul>
    </div>
    <div id="content">
      <div class="post">
        <h1 class="title"><a href="#">Administruoti mokinį</a></h1>
                                    <div><input type="button" name="buton"
value="Redaguoti" onclick="" />
<input type="button" name="buton" value="Trinti" onclick="" />
</div>
<table cellpadding="0" cellspacing="0" border="0" id="tableList">
            <tr>
                        <th>adresas</th>
            <th>el_pastas</th>
            <th>gimimo_data</th>
            <th>mokinio_id</th>
            <th>pavarde</th>
            <th>vardas</th>
            <th>lytis</th>
            <th>klasė</th>

</tr>
</table>

      </div>
    </div>
    <div style="clear: both;"> </div>
  </div>
</div>
</div>
</div>
<div id="footer-wrapper">
  <div id="footer">
    <p class="copyright">&copy;  2010 Visos teisės saugomos.</p>
  </div>
</div>
</body>
</html>
```

HTML programinis kodas „Naujo mokinio informacijos įvedimo“ langui

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Mokyklos IS</title>
<meta name="keywords" content="" />
```

```
<meta name="Hanging" content="" />
<link href="styles/default.css" rel="stylesheet" type="text/css" media="screen" />
</head>
<body>
<div id="wrapper">
  <div id="header-wrapper">
    <div id="header">
      <div id="logo">
        <h1><a href="#"><span>Mokyklos IS</span></a></h1>
        <p>sugeneruota grafinė sąsaja</p>
      </div>
    </div>
  </div>
  <div id="page">
  <div id="page-bgtop">
   <div id="page-bgbtm">
   <div id="sidebar1" class="sidebar">
      <ul>
        <li>
          <h2>Meniu</h2>
          <ul>
                        <li><a href="Administruoti__bureli.html">Administruoti
būrelį</a></li>
            <li><a href="Administruoti_klase.html">Administruoti klasę</a></li>
            <li><a href="Administruoti_mokini.html">Administruoti mokinį</a></li>
            <li><a href="Administruoti_mokomaji_dalyka.html">Administruoti mokomąjį
dalyką</a></li>
            <li><a href="Administruoti_mokomaji_dalyka.html">Administruoti mokomąjį
dalyką</a></li>
            <li><a href="Administruoti_mokytoja.html">Administruoti
mokytoją</a></li>
            <li><a href="Administruoti_pazymi.html">Administruoti pažymį</a></li>
            <li><a href="Administruoti_sistemos_vartotoja.html">Administruoti
sistemos vartotoją</a></li>
            <li><a href="Informuoti_tevus.html">Informuoti tėvus</a></li>
            <li><a href="Pagrindinio_lango_valdymas.html">Pagrindinio lango
valdymas</a></li>
            <li><a href="Prisijungti.html">Prisijungti</a></li>
            <li><a href="Redaguoti_burelio_informacija.html">Redaguoti būrelio
informaciją</a></li>
            <li><a href="Redaguoti_informacija_apie_mokomaji_dalyka.html">Redaguoti
informaciją apie mokomąjį dalyką</a></li>
            <li><a href="Redaguoti_klases_informacija.html">Redaguoti klasės
informaciją</a></li>
            <li><a href="Redaguoti_mokinio_informacija.html">Redaguoti mokinio
informaciją</a></li>
            <li><a href="Redaguoti_mokytojo_informacija.html">Redaguoti mokytojo
informaciją</a></li>
            <li><a href="Redaguoti_pazymi.html">Redaguoti pažymį</a></li>
            <li><a href="Redaguoti_vartotoja.html">Redaguoti vartotoją</a></li>
            <li><a href="test.html">test</a></li>
            <li><a href="Ivesti_burelio_informacija.html">Įvesti būrelio
informaciją</a></li>
            <li><a href="Ivesti_mokytojo_informacija.html">Įvesti mokytojo
informaciją</a></li>
            <li><a href="Ivesti_naujo_mokinio_informacija.html">Įvesti naujo
mokinio informaciją</a></li>
            <li><a href="Ivesti_naujos_klases_informacija.html">Įvesti naujos
klasės informaciją</a></li>
            <li><a href="Ivesti_nauja_mokomaji_dalyka.html">Įvesti naują mokomąjį
dalyką</a></li>
            <li><a href="Ivesti_pazymi.html">Įvesti pažymį</a></li>
            <li><a href="Ivesti_vartotoja.html">Įvesti vartotoją</a></li>

</ul>

      </ul>
    </div>
    <div id="content">
      <div class="post">
```

```
        <h1 class="title"><a href="#">Įvesti naujo mokinio informaciją</a></h1>
                                    <table cellpadding="0" cellspacing="0"
border="0" >
            <tr>
                        <td>el_pastas</td>
            <td><input type="text" name="text_box[]" value="" />
</td>

</tr>
<tr>
                        <td>gimimo_data</td>
            <td><input type="text" name="text_box[]" value="" />
</td>

</tr>
<tr>
                        <td>mokinio_id</td>
            <td><input type="text" name="text_box[]" value="" />
</td>

</tr>
<tr>
                        <td>pavarde</td>
            <td><input type="text" name="text_box[]" value="" />
</td>

</tr>
<tr>
                        <td>vardas</td>
            <td><input type="text" name="text_box[]" value="" />
</td>

</tr>
<tr>
                        <td>berniukas</td>
            <td><input type="radio" name="radio[berniukas]" value="1" />
</td>

</tr>
<tr>
                        <td>mergaite</td>
            <td><input type="radio" name="radio[mergaite]" value="1" />
</td>

</tr>
<tr>
                        <td>adresas</td>
            <td><textarea cols="16" rows="3" name="adresas">adresas</textarea>
</td>

</tr>
<tr>
                        <td>klasė</td>
            <td><select name="klasė">
            <option value="0">klasė</option>
</select>
</td>

</tr>
<tr>
                        <td></td>
            <td><input type="button" name="buton" value="Atšaukti" onclick="" />
<input type="button" name="buton" value="Išsaugoti" onclick="" />
</td>

</tr>

</table>

      </div>
```

```
    </div>
    <div style="clear: both;"> </div>
  </div>
</div>
</div>
</div>
<div id="footer-wrapper">
  <div id="footer">
    <p class="copyright">&copy; 2010 Visos teisės saugomos.</p>
  </div>
</div>
</body>
</html>
```

Priedas Nr. 5. Darbų pasidalinimas

Lina Jankauskaitė	Tadas Karlonas
1. ĮVADAS	
2. ANALIZĖS DALIS 2.1 Analizės tikslas 2.2 Tyrimo sritis, objektas ir problema 2.7 Problemos sprendimo metodų literatūros šaltiniuose analizė	
2.4 Vartotojų analizė 2.5 Vartotojo sąsaja ir jos struktūra *2.5.3 Vartotojo sąsajos prototipo sudarymas* 2.6 UML modeliavimo kalba *2.6.1 UML diagramų tipai* 2.7 Problemos sprendimo metodų literatūros šaltiniuose analizė 2.8 GUI generuojančių įrankių analizė *2.8.1 MagicDraw UML* 2.9 Siekiamos sistemos apibrėžimas 2.11 Kompiuterizuojamos funkcijos 2.14 Rezultato kokybės kriterijai	2.3 Aplinkos analizė 2.5 Vartotojo sąsaja ir jos struktūra *2.5.1 Aplikacijos grafinės sąsajos elementai* *2.5.2 Internetinės grafinės sąsajos elementai* *2.6.2 UML diagramų ryšys su GUI* 2.7 Problemos sprendimo metodų literatūros šaltiniuose analizė 2.8 GUI generuojančių įrankių analizė *2.8.2 Oracle Designer* *2.8.3 Provision Workbench* 2.10 Darbo tikslas ir siekiami privalumai 2.12 Reikalavimai duomenims 2.13 Nefunkciniai reikalavimai ir apribojimai
3. REIKALAVIMŲ ANALIZĖ IR SPECIFIKACIJA	
3.1 Reikalavimų specifikacija 3.3 Reikalavimų analizė 3.4 Reikalavimų analizės apibendrinimas	3.1 Reikalavimų specifikacija 3.2 Dalykinės srities modelis 3.3 Reikalavimų analizė
4. ALGORITMO APRAŠAS	
4.1 UML diagramos ir jų sąsaja su GUI 4.3 Naudojamų UML diagramų metamodelis 4.6 UML metaklasių transformacija į internetinės grafinės vartotojo sąsajos metaklases 4.7 Algoritmas grafinės vartotojo sąsajos generavimui *4.7.2 Algoritmas internetinės grafinės vartotojo sąsajos generavimui*	4.2 Apribojimai diagramoms 4.4 Vartotojo sąsajos metamodeliai 4.5 UML metaklasių transformacija į aplikacijos grafinės vartotojo sąsajos metaklases *4.7.1 Algoritmas aplikacijos grafinės vartotojo sąsajos generavimui* 4.8 GUI atvaizdavimo programiniu kodu algoritmas

<table>
<tr><td colspan="2">5. GRAFINĖS VARTOTOJO SĄSAJOS GENERAVIMO IŠ UML DIAGRAMŲ ĮRANKIO PROJEKTAS</td></tr>
<tr><td>5.1 Grafinės vartotojo sąsajos generavimo metodo pagrindimas ir esmės išdėstymas
5.3 Detalus projektas</td><td>5.2 Sistemos architektūra
5.4 Duomenų bazės schema
5.5 Realizacijos modelis</td></tr>
<tr><td colspan="2">6. REALIZACIJA</td></tr>
<tr><td>6.1 Diagramų susiejimas MagicDraw pakete
6.3 Diagramų stereotipai ir MagicDraw profilio GUI naudojimas
6.5 Sistemos veikimo aprašymas
6.6 Testavimo modelis bei duomenys</td><td>6.2 MagicDraw XML dokumento nuskaitymas
6.2.1 XML dokumentų nuskaitymo būdai
6.4 Klasių atributų tipų ir grafinių elementų tipų atitikmenys
6.5 Sistemos veikimo aprašymas
6.7 Sukurto grafinės sąsajos generavimo algoritmo realizacijos apibendrinimas</td></tr>
<tr><td colspan="2">7. EKSPERIMENTINIS SISTEMOS TYRIMAS</td></tr>
<tr><td>7.1 Įvedami į grafinės sąsajos generavimo įrankį duomenys
7.2 Automatinis grafinės sąsajos generavimas
7.2.2 Grafinės sąsajos peržiūra internetinei sąsajai
7.3 Kokybės kriterijų įvertinimas
7.3.2 Grafinės vartotojo sąsajos generavimo laiko įvertinimas
7.4 Savybių analizė ir taikymo rekomendacijos</td><td>7.2 Automatinis grafinės sąsajos generavimas
7.2.1 Aplikacijos grafinės vartotojo sąsajos peržiūra ir redagavimas
7.3 Kokybės kriterijų įvertinimas
7.3.1 Grafinės vartotojo sąsajos sugeneravime naudojami grafiniai elementai
7.3.3 Grafinės vartotojo sąsajos generavimas iš įvairaus sudėtingumo UML diagramų</td></tr>
<tr><td colspan="2">8. IŠVADOS</td></tr>
<tr><td colspan="2">Santrumpų ir terminų žodynas</td></tr>
<tr><td colspan="2">Priedas Nr.1. Vartotojo vadovas</td></tr>
<tr><td colspan="2">Priedas Nr. 2. Eksperimento duomenys ir rezultatai</td></tr>
<tr><td colspan="2">Priedas Nr. 3. Sugeneruoti internetinės grafinės vartotojo sąsajos langai eksperimentinei Mokyklos informacijos sistemai</td></tr>
<tr><td colspan="2">Priedas Nr. 4. Straipsnis, pristatytas 15-oje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos (IVUS 2010)“</td></tr>
</table>

Priedas Nr. 6. Straipsnis, pristatytas 15-oje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos (IVUS 2010)“

Grafinės vartotojo sąsajos generavimas iš UML diagramų

Tadas Karlonas, Lina Jankauskaitė
Informacijos sistemų katedra, Informatikos fakultetas
KAUNO TECHNOLOGIJOS UNIVERSITETAS
Kaunas, Lietuva

Reziume - Vartotojo sąsaja yra svarbi programinės įrangos dalis, kuri reikalinga norint perduoti informaciją iš vartotojo sistemai bei vykdyti jos funkcijas. Daugeliu atveju, iš UML kalba aprašytos sistemos specifikacijos būtų galima automatiškai ar automatizuotai generuoti grafinę vartotojo sąsają – pradinį būsimos sistemos prototipą. Šiame darbe siūlomas sprendimas remiasi trimis UML diagramomis (panaudojimo atvejų, klasių ir veiklų) ir jose aprašytos informacijos transformavimu į grafinės vartotojo sąsajos elementus, kurie atvaizduojami grafinės sąsajos languose.
Raktažodžiai: UML, grafinė sąsaja, metamodeliai

1. Įvadas

Kuriant informacines sistemas (IS), dažnai siekiama kuo anksčiau pademonstruoti vartotojui būsimos sistemos prototipą. Toks prototipas kūrimo proceso pradžioje dažniausiai būna tik grafinė vartotojo sąsaja, tačiau jai sukurti reikalingos papildomos priemonės ir papildomas sistemos projektuotojų darbas. Sistemos projektuotojai tuo pačiu turi kurti ir sistemos reikalavimų specifikaciją bei projekto modelį, o jiems aprašyti dažnai naudojama UML (angl. *Unified Modeling Language*) kalba. Daugeliu atveju, iš UML kalba aprašytos sistemos specifikacijos būtų galima automatiškai ar automatizuotai generuoti grafinę vartotojo sąsają – pradinį būsimos sistemos prototipą.

IS kūrimo procesai padeda sistemų inžinieriams identifikuoti kompiuterizuojamus panaudojimo atvejus, išanalizuoti verslo problemas ir jas spręsti sukuriant informacinę sistemą. Kuriant tokią sistemą, labai svarbu surinkti, specifikuoti ir išanalizuoti užsakovo pateikiamus reikalavimus. Funkciniams reikalavimams aprašyti dažniausiai naudojami įvairūs UML diagramų tipai (panaudojimo atvejų, klasių, veiklos, sekos ir kt. diagramos). Šiose diagramose aprašytą informaciją ir būtų galima panaudoti grafinės vartotojo sąsajos prototipo sudarymui.

2. *Vartotojo sąsajos generavimo būdai ir įrankiai*

Vartotojo sąsaja yra svarbi programinės įrangos dalis, kuri reikalinga norint perduoti informaciją iš vartotojo sistemai bei vykdyti jos funkcijas. Tai vienintelė sistemos dalis, kurią mato vartotojas ir kurios pagalba jis valdo sistemą. Vartotojo sąsajos prototipo sukūrimas yra naudingas šiuose IS kūrimo etapuose:

- reikalavimų analizės ir specifikavimo - pademonstruoja būsimos sistemos veikimo principus pagal vartotojo pateiktus reikalavimus;
- projektavimo ir realizavimo – realizuojant sistemą naudojamas jau sukurtas grafinės sąsajos prototipas, be to, prototipas yra susietas su projekto modeliais ir jų realizacija;
- palaikymo – specifikuotas ir aprašytas prototipas, susietas su realizacija, palengvina tolimesnį sistemos tobulinimą.

Vartotojo sąsajos sudėtines dalis galima aprašyti metamodeliu, kuriame fiksuojama, kokie elementai sudaro vartotojo sąsają ir kaip jie siejasi tarpusavyje. Vienas tokių vartotojų sąsajos metamodelių pavyzdžių pateikiamas [1]. Čia aprašomos vartotojo sąsajos saugumo problemos, o sąsajos metamodelis padeda susieti vartotojo sąsajos elementus su saugumą užtikrinančiais elementais. Šiame darbe, generuojant vartotojo sąsajos prototipą, saugumo problema nėra nagrinėjama, bet [1] pateiktas metamodelis naudingas kaip formalus grafinės sąsajos struktūros aprašas. Grafiniai elementai išskaidomi į tam tikras grupes pagal jų tipus ir atliekamus veiksmus. Šių elementų elgsena aprašoma priskiriant veiksmus.

Šiame darbe vartotojo sąsajos prototipo sudarymui reikalingą informaciją siūloma surinkti iš UML diagramų, kurios sudaromos IS kūrimo metu. Sistemų projektuotojai UML pagalba sumodeliuoja pagrindinius sistemos procesus, funkcionalumą, numatomą struktūrą, klases, jų atributus, operacijas ir ryšius.UML – tai vieninga modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektinių programų dokumentus [2]. Ji aprūpina sintaksiniais ženklais, kurie apibūdina visus pagrindinius sistemos vaizdus, naudodama įvairias diagramų rūšis. UML 2.0 [3] apibrėžiama trylika diagramų (išskaidytų į dvi didesnes šakas: vaizduojančias sistemos elgseną ir struktūrą), kurios drauge sudaro bendrą sistemos vaizdą su reikalingais aspektais. Šiame straipsnyje siūlomas sprendimas – vartotojo sąsajos generavimas, naudojant informaciją, aprašytą UML panaudojimo atvejų, klasių ir veiklos diagramose.

a. Vartotojo sąsajos sudarymo įrankiai

Šiuo metu yra sukurta keletas įrankių, kurie palengvina grafinės vartotojo sąsajos sudarymą. Vienas iš jų – *MagicDraw 16.6.*, kuriame siūlomas specifinis diagramų tipas – vartotojo sąsajos modeliavimo diagrama. Šios diagramos [4] pagalba galima sudaryti vartotojo sąsajos prototipą, naudojantis pagrindiniais sąsajų elementais: mygtukais, meniu, sąrašais, tekstiniais laukais, lentelėmis, pasirinkimo elementais, nuorodų, slaptažodžių laukais ir t.t. bei susieti ją su atitinkamomis klasėmis tame pačiame projekte. Nors ši diagrama ir palengvina darbą sistemų projektuotojams, tačiau pagrindinis jos trūkumas – iš atskirų elementų vartotojo sąsajos diagrama yra suprojektuojama rankiniu būdu, taigi reikalauja papildomų projektuotojo darbo sąnaudų, be to, gali būti išsaugoma ir peržiūrima tik kaip paveikslėlis arba interneto sąsajos prezentacija.

Oracle Forms Builder [5] įrankis leidžia generuoti vartotojo sąsają. O*racle Forms Builder* siejasi su sukurta *Oracle* (ar kita) duomenų baze ir *The Forms Runtime* įrankio pagalba bei eile grafinių elementų leidžia sukurti formas, kurios parodo, įveda ar redaguoja duomenis toje duomenų bazėje. Vartotojo sąsajos formos yra kuriamos trigerių pagrindu juos modifikuojant, taigi dirbant šiuo įrankiu reikalingos ne tik projektavimo, bet ir programavimo žinios.

The User Interface Generator yra *IBM® InfoSphere Master Data Management (MDM) Workbench* komponentas [6], kurio dėka galima sudaryti vartotojo, tikslų bei užduočių diagramas, kurios reikalingos norint sugeneruoti grafinę vartotojo sąsają. Sukūrus šias diagramas, jas galima panaudoti internetinei sąsajai sukurti naudojantis papildomu *The User Interface Generator* įrankiu. Nei vienas iš aptartųjų įrankių nesuteikia galimybės generuoti vartotojo sąsajos prototipo iš UML diagramų, kurios sudaromos IS kūrimo metu.

b. Vartotojo sąsajos generavimo metodai

Vartotojo sąsajos generavimui siūlomi įvairūs sprendimai. J. He, I-L. Yen siūlomas sprendimas skirtas būtent tinklo paslaugų sistemoms. Tinklo paslaugos (angl. *web services*) gali būti iškviečiamos viena kitos arba vartotojo. Šių iškvietimų pobūdis yra skirtingas. Tam, kad vartotojas galėtų iškviesti tinklo paslaugas, jis turi matyti grafinę vartotojo sąsają. Jai generuoti autoriai siūlo naudoti [7] WSDL (angl. *Web Service Description Language*) aprašymo kalbą kartu su papildomais grafinės sąsajos aprašymais .

Vartotojo interfeisą taip pat siūloma generuoti ir naudojant XUL (angl. *XML-based User-interface Language*), XML (angl. *eXtensible Markup Language*) ir Gecko biblioteką [8].

Sudarius XUL ir XML failus, jie yra nuskaitomi ir įrašomi į atmintį. Kiekvienam XUL elementui yra priskiriamas tam tikras dizaino elementas.

[9] siūlo grafinę vartotojo sąsają generuoti iš šių UML diagramų: panaudojimo atvejų, klasių, bendradarbiavimo ir būsenų. Papildomi apribojimai kuriamai sistemai aprašomi OCL (angl. *Object Constraint Language*) kalba [9]. Klasių diagramą autorius siūlo naudoti sistemos struktūros aprašymui ir ryšių tarp sistemos klasių identifikavimui. Panaudojimo atvejų diagrama yra susijusi su sistemos vartotojų ir sistemos sąveika, todėl šia diagrama aprašomas sistemos kontekstas, funkcionalumas ir elgsena. Bendradarbiavimo diagrama naudojama identifikuoti kiekvieno panaudojimo atvejo vykdymo scenarijui. Šią funkciją taip pat gali atlikti ir sekų diagrama. Būsenų diagrama parodo objektų būsenų seką įvairiuose vykdymo etapuose. Ji gali būti sujungta su objektų klasėmis, kurių būsenos kinta dinamiškai.
Polak G., Jarosz J. siūlo vartotojo grafinę sąsają generuoti naudojant aprašytus šablonus. Išsirinkus atitinkamus parametrus yra sugeneruojamas standartinio stiliaus grafinis interfeisas [10].
Taigi, egzistuojantys vartotojo sąsajos generavimo metodai dažniausiai:
remiasi ne UML diagramomis, o jau su realizacija susietais būsimos sistemos aprašais ([7], [8], [10]);
naudoja UML diagramas, bet reikalauja daug papildomos informacijos (pvz. OCL aprašų [9]), kurią kartais sudėtingiau aprašyti, negu sukurti pačią grafinę sąsają.

3. Grafinės vartotojo sąsajos generavimo algoritmas

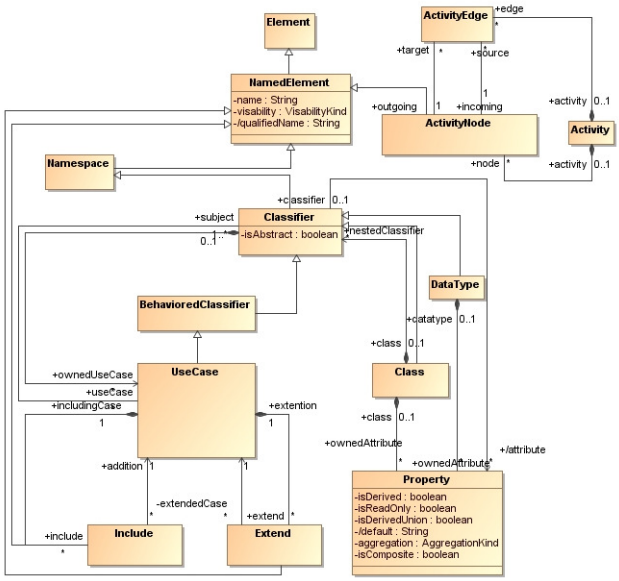
Šiame darbe siūlomas sprendimas remiasi trimis UML diagramomis (panaudojimo atvejų, klasių ir veiklų) ir jose aprašytos informacijos transformavimu į grafinės vartotojo sąsajos elementus. Pradiniai darbe siūlomo algoritmo duomenys – tai UML diagramos, o rezultatas – grafinės vartotojo sąsajos elementai, iš kurių galima sudaryti sąsajos vaizdą. Pradiniams duomenims aprašyti naudojamas UML diagramų metamodelis, o rezultatui - grafinės vartotojo sąsajos metamodelis. UML diagramų metaklasės transformuojamos į langų metaklases, ir galiausiai jos atvaizduojamos grafiškai sudarant sąsajos langus su diagramose numatytais elementais (paveikslas I).

Paveikslas I. Vartotojo sąsajos generavimo pagrindiniai etapai

a. Pradiniai algoritmo duomenys – naudojamų UML diagramų metamodelis

Remiantis UML 2.0 metamodelio aprašymu sudarytas metamodelis diagramoms, kurios naudojamos kuriant grafinę vartotojo sąsają.

UML metamodelį sudaro trys didesnės elementų kategorijos [4]: klasifikatoriai (angl. *classifier*), įvykiai (angl. *events*) ir elgsenos (angl. *behaviors*).Kadangi UML 2.0 metamodelis yra universalus ir gali būti naudojami įvairiausiose situacijose, II paveiksle pavaizduotos atrinktos UML metamodelio panaudojimo atvejų, klasių ir veiklų klasės bei klasės, susijusios su šiomis diagramomis. Tai yra duomenys, kurie naudojami vartotojo sąsajos generavimui.



Paveikslas II. UML metamodelis

Pagrindinės klasės, kurios naudojamos pasiūlytame algoritme yra *UseCase*, *Class*, *Property*, *Activity*, *ActivityNode*, *ActivityEdge*. II paveiksle pavaizduotų ir algoritme naudojamų metaklasių detalesnis aprašas pateiktas I lentelėje.

Lentelė I. Duomenys, įvedami į sistemą ir jų paaiškinimas

Metaklasė	**Atributas**	**Paaiškinimas**
UseCase	name: string - panaudojimo atvejo pavadinimas	Panaudojimo atvejis aprašo sistemos įvykių specifikaciją. Nurodoma, ką sistema gali atlikti ir kokie aktoriai dalyvauja dirbant su sistema.
Class	name: string - klasės pavadinimas	*Class* apibūdina objektų, turinčių panašias ypatybes, rinkinį. Tai tam tikras klasifikatorius su atributais, priklausomybėmis ir operacijomis.
Property	/default: string - atributo reikšmė	*Property* atspindi klasės atributus arba atributų rinkinius.
DataType	datatype: string - atributo tipas	Tai yra tipas, pagal kurį identifikuojama tam tikro atributo reikšmė. Naudojamas programuojant duomenų tipo atpažinimui.
ActivityNode	name: string – veiklos pavadinimas	Abstrakti klasė, numatanti veiklų žingsnius.
ActivityEdge	source: string - esamos veiklos identifikacinis numeris target: string - veiklos, į kurią nukreipiama esama veikla, identifikacinis numeris	Abstrakti klasė, aprašanti dviejų veiklų tarpusavio ryšį.
Activity	Nėra.	Parametrizuoto elgesio specifikacija, aprašanti veiklą.

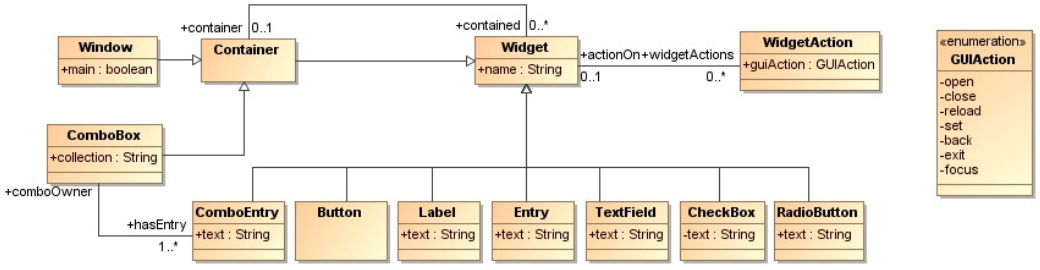
Iš panaudojimo atvejų metaklasių išsaugomas panaudojimo atvejo pavadinimas, kuris generuojant grafinę vartotojo sąsają virs lango pavadinimu. Kiekvienas panaudojimo atvejis – tai naujas langas grafinėje vartotojo sąsajoje. Klasės sistemoje apibūdina langų elementus, taigi reikia išsaugoti klasių pavadinimus, taip pat klasių atributų pavadinimus ir tipus. Veiklų informacija, atspindinti langų navigaciją, - tai jų pavadinimai ir tarpusavio ryšiai. Išsaugomi duomenys aprašyti II lentelėje.
Kadangi veiklos diagrama nurodo grafinės vartotojo sąsajos langų navigaciją, kai, pavyzdžiui, panaudojimo atvejų diagramoje yra tik vienas panaudojimo atvejis, ji nėra būtina. Visais kitais atvejais, kai yra keletas panaudojimo atvejų ir grafinėje vartotojo sąsajoje reikia sukurti keletą langų, veiklos diagramoje būtina atvaizduoti langų tarpusavio susiejimą bei atidarymo eiliškumą.

b. Grafinės vartotojo sąsajos metamodelis

Šiame darbe grafinės sąsajos elementams aprašyti naudojamas [1] pasiūlytas grafinės sąsajos metamodelis (paveikslas III). Grafinė vartotojo sąsaja išskaidoma į atskirus grafinius elementus, kurie atvaizduojami konteineryje. Šis elementas taip pat yra priskiriamas prie grafinių elementų. Prie jų taip pat priskiriami mygtukai, įrašai, informacijos įrašas, tekstiniai laukai, pasirinkimų elementai, sąrašo įrašas. Pats grafinės sąsajos langas ir informacijos sąrašas nėra laikomi grafiniais elementais dėl savo savybės turėti savo elementus.
Grafinės sąsajos metaklasių detalesnis aprašas pateiktas II lentelėje.

Lentelė II. Grafinės vartotojo sąsajos metamodelio klasių aprašas

Pavadinimas	**Aprašymas**
Window	Elementas, galintis turėti kitus grafinius elementus. Langai atvaizduojami medžio hierarchija.
Container	Abstraktus elemento tipas. Gali turėti kitus elementus.
Widget	Grafinis elementas.
WidgetAction	Aprašo trigerių inicijuotus veiksmus.
ComboBox	Grafinis elementas, turintis *ComboEntry*.
ComboEntry	Grafinis elementas, atspindintis *ComboBox* elemento egzempliorius.
Button	Grafinis elementas, inicijuojantis veiksmą.
Label	Grafinis elementas, naudojamas informacijos pateikimui. Šios informacijos vartotojas keisti negali.
Entry	Informacijos įrašas.
TextField	Tekstinis laukas.
CheckBox	Grafinis elementas, skirtas sprendimo pasirinkimui. Galimi keli pasirinkimai.
RadioButton	Grafinis elementas, skirtas sprendimo pasirinkimui. Galimas tik vienas pasirinkimas.



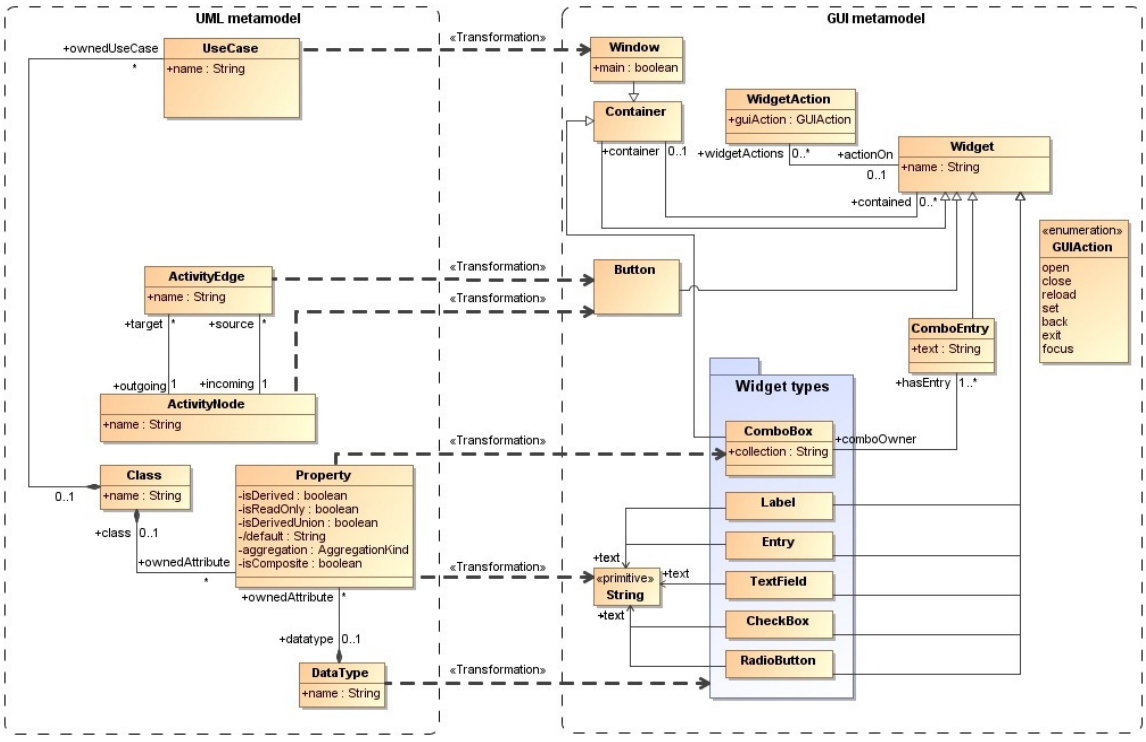
Paveikslas III. Grafinės vartotojo sąsajos metamodelis [1]

4. Grafinės vartotojo sąsajos generavimo algoritmo aprašymas

Transformavimo iš UML diagramų į vartotojo sąsajos prototipą algoritmo principai pateikiami IV paveiksle. Algoritmas susideda iš keleto žingsnių. Pirmiausiai transformuojami panaudojimo atvejai. Kiekviena panaudojimo atvejo metaklasė *UseCase* atitinka grafinės sąsajos metaklasę *Window*. Lango pavadinimas (metaklasės *Window* atributas *name*) gaunamas iš metaklasės *UseCase* atributo *name*.

Sekantis žingsnis – transformuoti klasių diagramos elementus – klases (metaklasė *Class*) ir jų atributus (metaklasė *Property*). Klasių (*Class*) atributai (*Property*) transformuojami į atitinkamus lango elementus (kuriuos apibendrina metaklasė *Widget*) pagal jų tipą. Šiuos lango elementus atitinka metaklasės *Label*, *Entry*, *TextField*, *CheckBox*, *RadioButton*, *ComboBox*. Atributo vardas (metaklasės *Widget* atributas *name*) gaunamas iš *Property* metaklasės atributo *default*, o tipas nustatomas iš *DataType* metaklasės. Tokiu būdu patikrinus visas klases, tų klasių ir jų atributai yra transformuojami į GUI elementų metaklases. Ryšys tarp klasės ir panaudojimo atvejo nustato, kurie atributai kuriam langui priklauso. Transformuojamas *Class – UseCase* ryšys į ryšį tarp *Window – Widget* metaklasių.

Veiklos diagramų elementai – veiklos (metaklasė *ActivityNode*) transformuojamos į grafinės sąsajos elementus ir atvaizduojami mygtukais (metaklasė *Button*). Mygtuko pavadinimas (klasės *Button* atributas *name*) gaunamas iš *AcivityNode* atributo *name*. Navigavimo planas, kuris nurodo kurį sekantį langą atidaryti, gaunamas iš veiklos ryšių (metaklasė *ActivityEdge*). Ryšiai gaunami iš klasės *ActivityEdge* atributų *source* ir *target*. Kiekviena veikla (klasė *AcitivityNode*) yra susieta su panaudojimo atveju.



Paveikslas IV. UML metamodelio klasių transformacija į GUI metamodelio elementus

Atlikus diagramų metaklasių transformavimą į langų metaklases, pastarosios gali būti atvaizduojamos į vartotojo sąsajos elementus, taip sudarant grafinį vartotojo sąsajos prototipo vaizdą.

V paveiksle pateikiamas algoritmo veikimą demonstruojantis pavyzdys su keliais panaudojimo atvejais, dviem klasėm („Klasė“ ir „Mokinys“) ir keletu veiklų. Diagramos yra

sudarytos MagicDraw įrankiu ir tarpusavyje susietos. Iš informacijos, pateiktos kairioje paveikslo pusėje, laikydamiesi pasiūlyto algoritmo principų, gauname vartotojo sąsają, pavaizduotą V paveikslo dešinėje pusėje. Tarp sukurtų sąsajos langų galima naviguoti mygtukų ir meniu pagalba.

Paveikslas V. Transformacija iš diagramų į programos langus

5. Algoritmo realizacija

Šiame darbe pasiūlyto algoritmo realizacija susieta su CASE įrankiu *MagicDraw*. Projektuotojui siūloma sistemą aprašančias diagramas sudaryti *MagicDraw* įrankiu. Šis įrankis diagramų informaciją išsaugo XML formatu. *MagicDraw* pakete sukurto sistemos projekto XML failas yra įkeliamas į šiame darbe realizuotą grafinės vartotojo sąsajos generavimo sistemą.

Diagramų informacijai nuskaityti naudojamas SAX (angl. *Simple API for XML*) – įvykiais grindžiamas XML dokumento nuskaitymas. Šiuo būdu nuskaitant dokumentą funkcijos yra iškviečiamos prieš pradedant ir baigiant skaityti dokumentą, taip pat prieš pradedant ir baigiant skaityti elementą. SAX metodu reikiama informacija iš dokumento nuskaitoma neįkeliant viso dokumento į atmintį.

XML dokumente informacija imama iš šių elementų: *packagedElement, node, supplier, client, edge, ownedAttribute* ir *referenceExtension*. Elemento tipo nustatymui naudojamas atributas *xmi:type*.

Jeigu elemento pavadinimas atitinka *packagedElement,* tikrinamas jo tipas. Pagal tai parenkamas atliekamas veiksmas. Elementų tipai ir su jais susiję veiksmai pateikiami III lentelėje.

Lentelė III. *packagedElement* Elemento tipai ir jų atliekamų veiksmų aprašas

***packagedElement* tipas**	**Atliekamas veiksmas**
uml:UseCase	Išsaugomas panaudojimo atvejis į masyvą.
uml:Class	Sukuriamas klasės objektas, į kurį bus surašyti atributai ir klasės pavadinimas.
uml:Activity	Nurodoma programai, kad toliau reikės saugoti veiklos diagramos elementus *node*.
uml:Usage	Nurodoma programai, kad toliau reikės išsaugoti veiklos ryšius su panaudojimo atveju. Šiame pakete yra saugojami veiklos ir panaudojimo atvejo ryšiai.

Elemento *node* tipai ir su jais susiję veiksmai pateikiami IV lentelėje.

Lentelė IV. *node* Elemento tipai ir jų atliekamų veiksmų aprašas

***node* tipas**	**Atliekamas veiksmas**
uml:CallBehaviorAction	Išsaugomas veiklos pavadinimas.

uml:InitialNode	Išsaugomas veiklos diagramos pradžios elementas.
uml:ActivityFinalNode	Išsaugomas veiklos diagramos pabaigos elementas.

Pagal elemento pavadinimą atliekami veiksmai pateikiami V lentelėje.

Lentelė V. Elementai ir jų atliekamų veiksmų aprašas

Elemento pavadinimas	**Atliekamas veiksmas**
supplier arba client	Išsaugomas ryšys tarp veiklos ir panaudojimo atvejo.
edge	Išsaugomas veiklos diagramos ryšys tarp elementų.
ownedAttribute	Pridedamas klasės atributas prie jau sukurto klasės objekto.
referenceExtension	Išsaugomas sukurto atributo tipas.

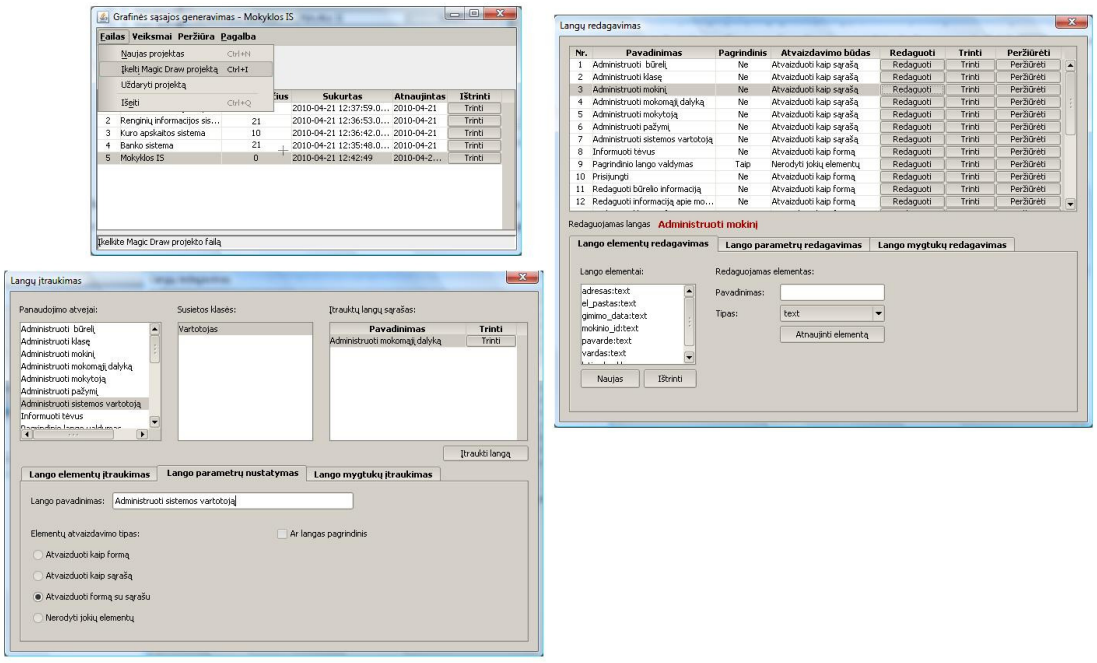
Baigus failo nuskaitymą, atliekamas kiekvieno elemento filtravimas pagal stereotipus ir grąžinami tik tie elementai, kuriems buvo priskirti atitinkami stereotipai. Panaudojimo atvejus žymi «*Window*», klases - «*WindowAttribute*», o veiklas - «*Button*» stereotipai. Atlikus filtravimą duomenys įgauna UML metamodelio duomenų struktūrą.

Naudojantis IV skyriuje aprašytu algoritmu atliekama UML metaklasių transformacija į grafinės sąsajos metaklases, kurios, sukūrus sąsajos langus, atvaizduojamos grafiniais elementais.

Iš pradžių yra sukuriami vartotojo sąsajos dialogai, atitinkantys nuskaitytus grafinės sąsajos langus. Į kiekvieną dialogą yra sukeliami lango elementai tikrinant jo tipą. Kiekviena programavimo kalba skirtingai aprašo grafinius elementus, todėl yra būtinas grafinės sąsajos elemento tikrinimas pagal tipą. Atvaizdavus grafinius elementus sąsajos lange, įkeliami mygtukai. Kiekvienam mygtukui yra nustatomas veiksmas, nurodantis, kurį sekantį langą atidaryti.

Vartotojo sąsajos generavimo algoritmas realizuotas *Java* programavimo kalba kaip atskira programa, kuri naudoja informaciją, gautą iš *MagicDraw* įrankiu sukurto XML projekto failo.

Keletas sukurtos programos langų pateikta VI paveiksle.



Paveikslas VI. Sukurtos sistemos langai

6. Išvados

Sistemos projektuotojai bet kuriuo atveju programinės įrangos kūrimo metu sudaro UML diagramas, aprašančias sistemos funkcionalumą, elgseną, struktūrą ir t.t. Šias diagramas galima panaudoti sistemos prototipo automatiniam generavimui. Tai naudinga ne tik projektuotojams, bet ir užsakovams, kuriems būsimą sistemą stengiamasi pademonstruoti kuo ankstesniuose kūrimo etapuose. Naudojant grafinės vartotojo sąsajos generavimo algoritmą, nebūtų naudojamas papildomas projektuotojų ir kitų specialistų darbas bei priemonės vartotojo sąsajos prototipui sudaryti.

Pastebėta, kad analizuotuose egzistuojančiuose sprendimuose vartotojo sąsajos generavimas yra sudėtingas ir užima daug laiko aprašant diagramas, be to reikalauja daug papildomos informacijos arba specialių aprašų.

Pagal šiame straipsnyje pasiūlytą algoritmą sukurta sistema naudoja minimalų reikalingų diagramų (panaudojimo atvejų, klasių bei veiklų) kiekį ir neperkrauna vartotojo reikalavimais įvesti daug papildomos informacijos. Grafinę vartotojo sąsają galima generuoti automatiškai, tačiau įrankio pagalba taip pat galima automatizuotai įkelti naujus bei redaguoti esančius grafinės sąsajos langus.

Ateityje planuojama algoritmą pritaikyti internetinės vartotojo grafinės sąsajos generavimui bei sistemos vartotojui pateikti sugeneruotos grafinės sąsajos programinį kodą tolimesniam darbui. Tobulinant ir plečiant sistemos galimybes, reikėtų išplėsti naudojamus grafinės sąsajos elementus, įtraukiant užduočių juostas, ir kt. sudėtingesnius grafinius elementus.

Literatūra

[1] D. Basin, M. Clavel, M. Egea, and M. Schlapfer „Automatic Generation of Smart Security-Aware“, ETH Zurich, Switzerland.

[2] „Unifiede Modeling Language Superstructure Specification“, Version 2.0. Dokumento numeris ptc/03-08-02. OMG, 2003.

[3] „OMG Unified Modeling Language**TM“** (OMG UML), Superstructure, Version 2.2.

[4] No Magic, Inc „MagicDraw User Manual“, 2010.

[5] M. Adler, A. Bers, C. Broxton, M. Caccamo, K. Chu, F. Rovitto „Developer/2000: Guidelines for Building Applications“ http://download.oracle.com/docs/pdf/A50994_2.pdf, 1997.

[6] W. Jones, J. Limburn „Build a user model with Rational Software Architect and the User Interface Generator“ http://www.ibm.com/developerworks/library/ar-modelingguide /index.html?S_TACT=105AGX01&S_CMP=LP, 2009.

[7] J. He, I-L. Yen „Adaptive User Interface Generation for Web Services“, IEEE International Conference on e-Business Engineering.

[8] N. Aloia, C. Concordia, M. T. Paratore „Automatic GUI Generation for Web Based Information Systems“.

[9] M. Elkoutbi, I Khriss., K. Keller R. „User Interface Prototyping using UML Specifications“.

[10] G. Polak, J. Jarosz „Automatic Graphical User Interface Form Generation Using Template Haskell“, Department of Computer Science University of Science and Technology in Krak´ow.