KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Marius Zaviliauskas

ODRES METODO REALIZAVIMO NAUDOJANT STANDARTIZUOTAS MODELIAVIMO KALBAS GALIMYBIŲ TYRIMAS

Baigiamasis magistro projektas

Vadovas
prof. dr. R. Butleris

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

ODRES METODO REALIZAVIMO NAUDOJANT STANDARTIZUOTAS MODELIAVIMO KALBAS GALIMYBIŲ TYRIMAS

Baigiamasis magistro projektas

Informacinių sistemų inžinerijos studijų programa (kodas 621E15001)

Vadovas

prof. dr. R. Butleris

2015-05-25

Konsultantas

lekt. mag. T. Danikauskas

2015-05-25

Recenzentas

doc. dr. T. Blažauskas

2015-05-25

Projektą atliko

Marius Zaviliauskas

2015-05-25

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

(Fakultetas)

(Studento vardas, pavardė)

Informacinių sistemų inžinerijos studijų programa, 621E15001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „ODRES metodo realizavimo naudojant standartizuotas modeliavimo kalbas galimybių tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano, **Mariaus Zaviliausko**, baigiamasis projektas tema „.....“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Zaviliauskas, M. Research on the implementation of ODRES method using standardized modeling languages. *Final Degree Project of Master of Information Systems Egineering* / Supervisor Prof. Dr. Rimantas Butleris; Kaunas University of Technology, Faculty of Informatics.

Kaunas, 2015. 98 p.

SUMMARY

The purpose of the work is to explore and suggest a solution of *ODRES* method realization in a standardized modeling language capabilities.

ODRES is requirements capture and specification method, which is based on information flow analysis. *ODRES* does not have standardized models and their elements notation, so the use of method is complicated. The aim during work is to adapt *ODRES* into context of standardized language and to implement to that language modeling tool.

The analysis is carried out in *ODRES* method, it‘s elements and process analysis. The standardized modeling languages *IDEF*, *UML* and *BPMN* are analyzed too. After analysis and comparison of selected modeling languages, *UML* modeling language was selected for *ODRES* method adaptation. Modeling tools analysis was made and it was decided that *MagicDraw* tool is the best suitable tool for *ODRES* method implementation with *UML* language.

Requirements and project for *ODRES* method were made during work. In requirements specification part *ODRES* method use cases were formed indicating what activities should be carried out during information system specification by *ODRES* method process and design according to a made specification. Also processes were made, that show how information system requirements should be specified in *UML* language by *ODRES* method process. In project part *ODRES* method and *UML* language elements matches were found and formed.

ODRES method profile in *MagicDraw* tool was formed for extending *UML* metamodel with additional stereotypes necessary to adapt the method. Also *MagicDraw* project template was created. Starting a new project with *MagicDraw* tool according to concluded template a necessary folder structure is automatically created and *ODRES* profile is attached, thus facilitating requirements specification by *ODRES* method. Report generation template is formed through which specification and project reports can be generated.

An experimental chosen subject area information system specification and design of the specified *ODRES* method requirements were made. According to the defined *ODRES* method process and *UML* element matches, successfull information system specification and design was performed in the range of *ODRES* requirements.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

UML (angl. Unified Modelling Language) - modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.

BPMN (angl. Business Process Model Notation) – verslo procesų specifikavimo modeliavimo standartas, kuris suteikia grafinę notaciją verslo procesų specifikavimui.

IDEF (angl. Integration DEFinition) – modeliavimo kalbų šeima, kuri apima platų panaudojimą, nuo funkcinio modeliavimo iki duomenų, simuliacijų, objektiškai orientuotų sistemų analizės ir veikimo modeliavimo.

ODRES (angl. *Output Driven Requirements Specification Method*) – informacijos srautais paremtas informacinės sistemos reikalavimų specifikavimo metodas.

IS – informacinė sistema.

OMG (angl. Object Management Group) - ne pelno siekiantis technologijos standartų konsorciumas.

CASE (angl. Computer-aided software engineering) - programinės įrangos kūrimo įrankių sritis skirta programinės įrangos projektavimui ir realizavimui.

ĮVADAS

Kauno technologijos universitete (KTU) informacijos sistemų katedroje yra kuriamas informacijos srautų specifikavimo metodas *ODRES* (angl. *Output Driven Requirements Specification Method*). Metodas grindžiamas informacijos srautų analize ir specifikavimu, pasižymi nuosekliu reikalavimų surinkimo ir specifikavimo procesu. Šis magistrinis darbas priklauso KTU informatikos fakulteto informacijos sistemų inžinerijos studijų programai.

Darbo problematika ir aktualumas

Viena svarbiausių faktorių siekiant sėkmingai įgyvendinti informacinės sistemos kūrimo projektą yra gerai atlikta vartotojo reikalavimų inžinerija, kuri susideda iš vartotojo poreikių surinkimo ir analizės, bei jų dokumentavimo. *ODRES* metodas turi nestandartizuotą diagramų ir elementų notaciją, kas sukelia nepatogumų populiarinant ir taikant šį informacinių sistemų specifikavimo metodą. *ODRES* metodas skirtas informacinių sistemų specifikavimui remiantis sistemos informaciniais srautais.

Darbo tikslas ir uždaviniai

Darbo tikslas yra ištirti ir surasti *ODRES* metodo realizavimo standartizuotai modeliavimo kalbai galimybes. Darbo metu turi būti išnagrinėtos grafinės modeliavimo kalbos, jų galimybės. Iš šių modeliavimo kalbų turi būti parinkta viena tinkamiausia ir šios kalbos kontekste bus realizuojamas *ODRES* metodas. Metodo pritaikymas turi būti realizuotas pasirinktam modeliavimo kalbos programiniam įrankiui. Darbo metu turi būti atliktas eksperimentinis tyrimas, pasirinktai dalykiniai sričiai atlikta sistemos specifikacija pagal *ODRES* metodo procesą pasirinktos modeliavimo kalbos kontekste.

Darbo rezultatai ir jų svarba

Darbo metu surasti *ODRES* metodo modelių ir *UML* modeliavimo kalbos elementų atitikmenys, metodo procesai realizuoti *UML* modeliavimo kalbos kontekste. Darbo metu sumodeliuota pasirinktos dalykinės srities informacinė sistema pagal *ODRES* metodą. *ODRES* metodas pritaikytas *UML* modeliavimo kalbos kontekste pasirinktame *MagicDraw* modeliavimo įrankyje, sukuriant *ODRES* profilį, projekto šabloną, ataskaitos generavimo šabloną. Tai palengvins šio metodo panaudojimo ir populiarinimo galimybes informacinių sistemų specifikavimo srityje.

Darbo struktūra

Dokumentą sudaro aštuoni skyriai:

1. Probleminės srities analizė - skyriuje analizuojamas *ODRES* metodas, jo vartotojai, modeliavimo kalbos ir įrankiai;
2. *ODRES* metodo reikalavimų specifikacija ir projektas, formalus aprašas - aprašomi *ODRES* metodo reikalavimai, veikimo procesas;
3. *ODRES* metodo sprendimo arba eksperimentinės realizacijos projektas - skyriuje aprašoma kaip metodas turi būti realizuojamas *UML* kalbos rėžiuose;
4. *ODRES* sprendimo realizacija - pateikiama informacija kaip *ODRES* metodas buvo realizuotas *MagicDraw* įrankyje;
5. Eksperimentinis *ODRES* metodo tyrimas - pateikiamas eksperimentinis tyrimas, kaip atliekamas pasirinktos dalykinės srities Valstybinės miškų tarnybos „Miškų kadastro integruotos informacinės sistemos“ specifikavimas;
6. Rezultatų apibendrinimas ir išvados - pateikiami darbo rezultatai;

1. PROBLEMINĖS SRITIES ANALIZĖ

1.1. Analizės tikslas.

Analizės tikslas išsiaiškinti *ODRES* metodo taikymo principus, surasti esamus metodo taikymus ir juos išanalizuoti. Taip pat išanalizuoti standartizuotas modeliavimo kalbas, įrankius, skirtus modeliuoti analizuojamomis modeliavimo kalbomis. Palyginti modeliavimo kalbas ir išsiaiškinti tinkamiausią *ODRES* metodo realizavimui. Tikslui pasiekti bus atliekama tyrimo objekto analizė, vartotojų analizė, bus analizuojami literatūros šaltiniai, įvertinama panašių tyrimų situacija Lietuvoje ir pasaulyje. Atlikus *ODRES* metodo ir modeliavimo kalbų analizę bus apibrėžtas siekiamas sprendimas.

1.2. Tyrimo objektas, sritis ir problema

Tyrimo objektas yra informacijos srautų specifikavimo metodas *ODRES*. Tyrimo sritis - informacinių sistemų inžinerijos metodai, modeliavimo kalbos, įrankiai. Tyrimo problema: *ODRES* specifikavimo metode naudojamos diagramos naudoja savitą elementų notaciją. Todėl metodo populiarinimas ir pritaikymas yra komplikuotas tiek notacijos standartizavimo atžvilgiu, tiek pritaikymo grafinio modeliavimo įrankiams atžvilgiu. Darbe bus siekiama *ODRES* metodo metamodelį ir procesą perkelti į tinkamiausios modeliavimo kalbos metamodelį neprarandant semantikos. Taip pat realizuoti *ODRES* metodo pritaikymą vienam iš esamų modeliavimo kalbos įrankių palaikančių modeliavimo kalbos notaciją.

1.3. Tyrimo objekto analizė

Informacinė sistema apibrėžiama kaip informacijos apdorojimo sistema ir organizacijos išteklių visuma, skirta informacijai apdoroti, formuoti (kurti), skleisti (siųsti ir gauti). Kitaip tariant, tai struktūrizuotas procesų ir procedūrų rinkinys, kuriame yra kaupiami duomenys, organizuojami ir perduodami vartotojui. Bet kuri informacinė sistema turi savo gyvavimo ciklą, apimantį kūrimą, veikimą, tobulinimą bei naikinimą. Informacinės sistemos kūrimo fazė, tai daugialypis procesas, kurio metu vykdoma analizė ir specifikavimas, projektavimas, realizavimas [1].

Kuriant informacines sistemas joms keliami funkciniai ir nefunkciniai reikalavimai, kurie nusako kuriamos sistemos galimybes ir ribas. Reikalavimų aibė būna didelė, todėl jų surinkimas, įvertinimas ir įgyvendinimas turi būti specifikuojamas. Norint atlikti specifikavimą turi būti sukuriamas procesas, užtikrinantis, kad produkto kūrėjai žinotų ir galėtų kontroliuoti, kuriuos reikalavimus jie turi ir jau įvertino, o kuriuos ne. Reikalavimų specifikavimą galima apibrėžti kaip visų informacinės sistemos funkcijų, atributų ir sąsajų nustatymą ir aprašymą, kad kuriama sistema tenkintų užsakovo reikalavimus.

ODRES yra informacinių sistemų reikalavimų surinkimo ir specifikavimo metodas. Jis pasižymi nuosekliu reikalavimų surinkimo ir specifikavimo procesu [2], [3]. Naudojantis *ODRES* metodu sudaroma informacinių srautų specifikacija, o šios specifikacijos pagrindu siekiama suprojektuoti vartotojo reikalavimus tenkinančią sistemą. Informacinėje sistemoje galima išskirti tris pagrindinius srautų tipus:

- įeinantys informacijos srautai;
- išeinantys informacijos srautai;
- vidiniai informacijos srautai.

Operuojant šiais trimis srautų tipais yra sudaroma informacinių srautų specifikacija. Kadangi informacinę sistemą galima įvardinti, kaip informacijos srautų apdorojimo sistemą, tai informacinę sistemą galima specifikuoti sudarant informacinių srautų specifikaciją. Ši specifikacija pagrįsta informacinei sistemai keliamų funkcinių reikalavimų specifikavimu. Informaciniai srautai įvardijami taip:

- Duomenų šaltinis – veiklos objektas saugantis duomenis, reikalingus kompiuterizuojamoms veiklos funkcijoms atlikti. Tai gali būti organizacijoje cirkuliuojantys dokumentai, žodiniai pranešimai ir kitos informacijos laikmenos;
- Rezultatas – formuojamas IS funkcionavimo metu (ekraninė forma, ataskaita, duomenų aibė pateikta elektroniniu formatu), kuris iš dalies pakeičia iki kompiuterizuotos IS įdiegimo arba plėtros organizacijoje cirkuliavusius popierinius dokumentus (ataskaitas, suvestines ir panašiai), žodžiu ar kitomis komunikavimo priemonėmis perduodamus informacijos srautus;
- Duomenų srautas – duomenų aibė, kuri perduodama iš vieno duomenų šaltinio kitam duomenų šaltiniui arba rezultatui. Srauto struktūrą sudarantys elementai parodo, kokio duomenų šaltinio atributo reikšmės bus perduodamos ir koks rezultato arba kito duomenų šaltinio atributas jas gaus. Šios reikšmės suformuoja atitinkamų duomenų šaltinių arba rezultatų, kuriems perduodamas srautas, atributų reikšmes.

ODRES metodo sąvokos aprašomos 1.1 lentelėje.

1.1 lentelė. ODRES metodo sąvokos

Sąvoka	Aprašymas
Funkcijų hierarchija	Kompiuterizuojamos informacinės sistemos galimų vykdymo funkcijų hierarchija, kur aukštesnio lygio funkcijos realizuojamos žemesnio lygio funkcijomis
Funkcija	Veikla, kurią turi įvykdyti informacinė sistema, kad pasiektų tam tikrą užsibrėžtą tikslą
Duomenų šaltinis	Veiklos objektas saugantis duomenis, reikalingus kompiuterizuojamoms veiklos funkcijoms atlikti. Tai gali būti organizacijoje cirkuliuojantys dokumentai, žodiniai pranešimai ar kitos informacijos laikmenos
Rezultatas	Ekraninė forma, ataskaita, duomenų aibė pateikta elektroniniu formatu ar kitokiu būdu sistemos pateikiama informacija vartotojui.
Atributas	Kiekybinė arba kokybinė tam tikro organizacijos objekto, apie kurį turi būti saugoma informacija savybė. Gali būti rezultato arba duomenų šaltinio
Veiksmas	Organizacijos veiklą sudarantis veiksmas, kuris sukuria, pašalina, pakeičia tam tikrus organizacijoje cirkuliuojančius informacijos srautus
Aktorius	Asmuo, rolė, organizacijos padalinys, ar kita sistema, turinti įgaliojimus atlikti tam tikrus veiksmus arba gauti veiksmo metu suformuotus rezultatus
Ryšys tarp duomenų šaltinių	Įvardinta reikšminga asociacija tarp duomenų šaltinių
Duomenų srautas	Srautas, kuris perduoda duomenis iš vieno duomenų šaltinio kitam duomenų šaltiniui arba rezultatui`
Duomenų šaltinio apdorojimo etapas	Etapas, kurio metu užpildoma dalis arba visi duomenų šaltinio egzempliorių aprašantys atributai
Perėjimas tarp apdorojimo etapų	Parodo iš kokio apdorojimo etapo ir į kurį apdorojimo etapą pereina duomenų šaltinis agentui atliekant tam tikrą veiksmą
Duomenų šaltinio būsena	Ją aprašo tam tikras duomenų šaltinio atributų, įgijusių reikšmes, ir ryšių su kitais duomenų šaltiniais rinkinys. Būsena parodo kokiame etape yra duomenų šaltinis ir kokius etapus jis praėjo

ODRES metodo elementai turi savitą ir nestandartizuotą notaciją. Visi elementai, išskyrus ryšius, turi kodą ir pavadinimą. Lentelėje pažymėtuoje numeriu 1.2 pateikiamas kelių pagrindinių

ODRES metodo elementų sąrašas su jų grafinės notacijos pavyzdžiais. Duomenų šaltinio, rezultato ir duomenų šaltinio apdorojimo etapo elementai žymimi vienodai juos išskiriant tik tekstiniu simboliu žyminčiu elemento reikšmę: rezultatas - „O“ raide nuo angliško žodžio „output“, duomenų šaltinis - „I“ raide nuo angliško žodžio „input“, duomenų šaltinio apdorojimo etapas - „E“ raide reiškiančia etapą. Duomenų srautas žymimas jungiančia linija, kuri vaizduoja tarp kurių elementų perduodami duomenys, ir trikampiu žyminčiu rodyklę. Aktorius atvaizduojamas mažu stačiakampiu nurodant jam kodą ir pavadinimą.

1.2 lentelė. ODRES metodo elementų pavyzdžiai

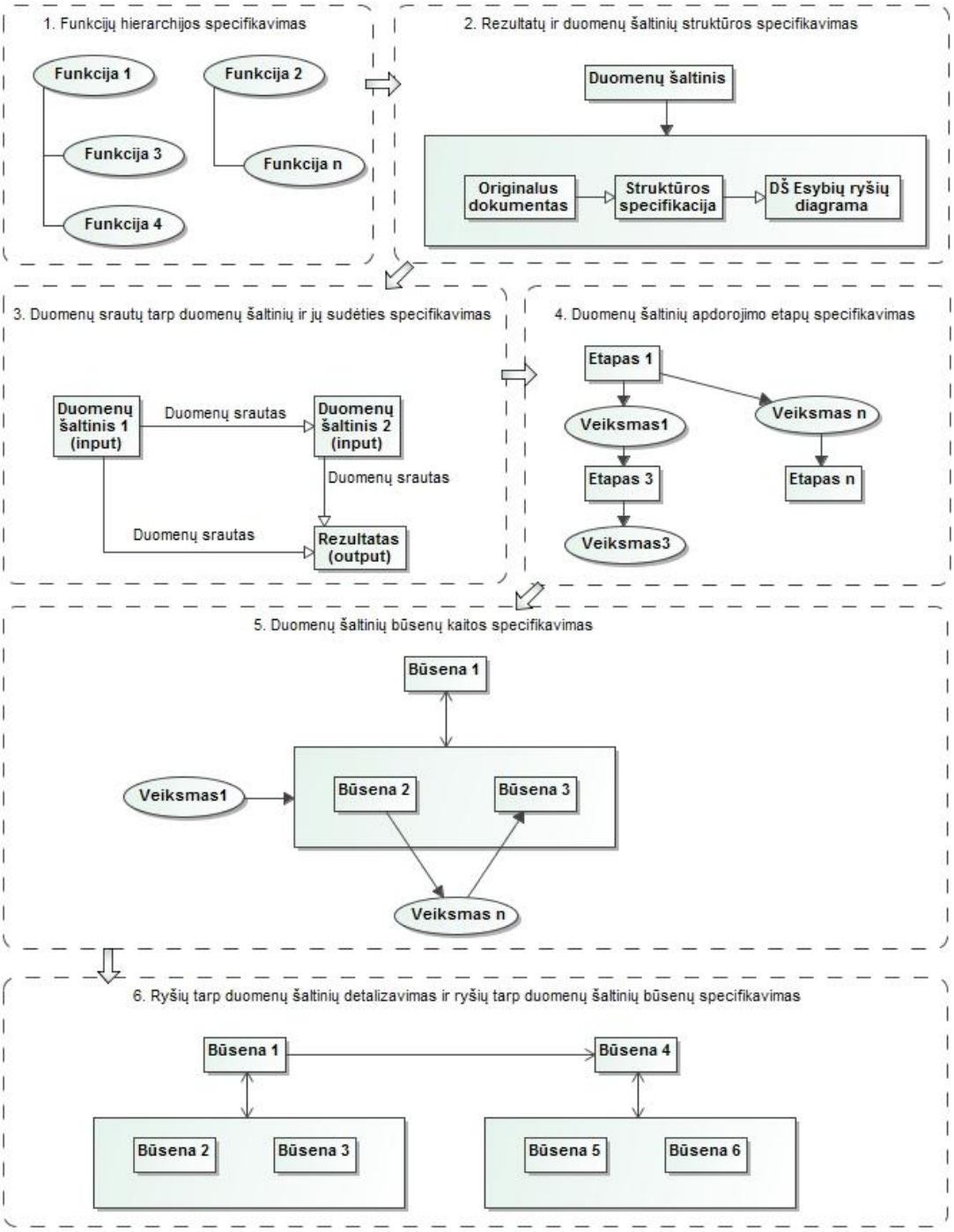
ODRES elementas	Grafinės notacijos pavyzdys
Funkcija	<kodas> <pavadinimas>
Rezultatas	O <kodas> <pavadinimas>
Aktorius	<kodas> <pavadinimas>
Duomenų šaltinis	I <kodas> <pavadinimas>
Duomenų srautas	<kodas> <pavadinimas>
Duomenų šaltinio apdorojimo etapas	E <kodas> <pavadinimas>
Būsena	<kodas> <pavadinimas>

1.1 paveiksle pateikiamas *ODRES* metodo darbinis procesas, kurį sudaro šeši pagrindiniai etapai.

Pirmajame etape sudaroma informacinės sistemos funkcijų hierarchija, surašomos ir sugrupuojamos visos funkcijos, kurias gali atlikti aktorius, padalinys ar kitoks nustatytas organizacinis vienetas.

Antrajame etape vykdomas duomenų šaltinių ir rezultatų struktūros specifikavimas, kiekvienam duomenų šaltiniui ir rezultatui sudaromas struktūros modelis ir esybių ryšių modelis.

Trečiajame etape specifikuojami duomenų srautai ir jų sudėtis. Duomenų srautus galima įvardinti kaip informacinius ryšius tarp duomenų šaltinių bei duomenų šaltinių ir rezultatų. Viena pagrindinių taisyklių šiems ryšiams specifikuoti yra tai, kad kiekvienam rezultatui formuoti reikalingas bent vienas duomenų šaltinis.



1.1 pav. ODRES metodo proceso konceptuali schema

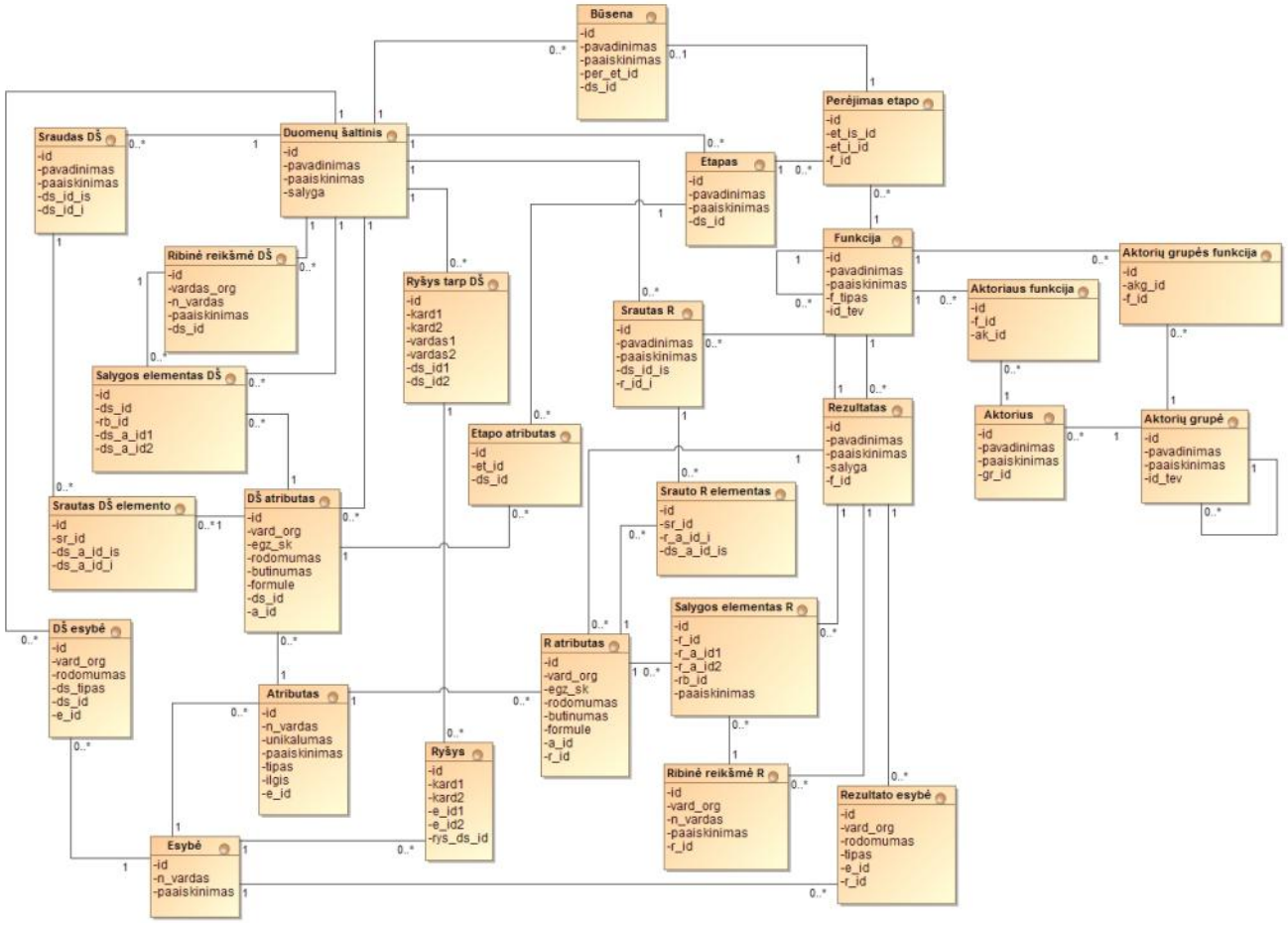
Ketvirto etapo metu specifikuojama, kaip apdorojami duomenų šaltiniai. Dažnai pasitaiko, kad duomenų šaltinis užpildomas informacija laipsniškai, tam tikrais atributų rinkiniais. Be to, šaltinio apdorojimas gali vykti lygiagrečiai su kito šaltiniu apdorojimu arba šaltinį apdoroja keletas aktorių.

Penkto etapo metu duomenų šaltiniai detalizuojami iki būsenų lygmens. Būsenų specifikavimas suteikia galimybę apibrėžti visus įmanomus duomenų šaltinio apdorojimo variantus. Visų variantų įvertinimas yra prielaida, kad sukurta kompiuterizuota informacinė sistema pilnai palaikys visus organizacijoje vykstančius veiklos procesus.

Šešto etapo metu atliekamas, duomenų srautų identifikuotų ir aprašytų trečiojo etapo metu perkėlimas į duomenų šaltinių būsenų modelį. Šis etapas leidžia patikslinti jau esančius ryšius tarp duomenų šaltinių ir rezultatų.

Atlikus visų šešių etapų veiksmus sudaroma informacijos srautų specifikacija, kurią galima įvardinti, kaip reikalavimų informacinei sistemai specifikaciją. Sudarytos specifikacijos pagrindu galima atlikti informacinės sistemos projektavimą.

Analizuojant *ODRES* metodą buvo pateiktas procesas, kaip turi vykti informacinės sistemos specifikavimas pagal *ODRES* metodo darbinį procesą. Visų žingsnių metu yra kaupiama informacija, duomenys, kurių pagalba ir paruošiama informacinės sistemos specifikacija. *ODRES* metodui yra sudaryta loginė duomenų saugyklos schema, kuri pateikta 1.2 paveiksle. Šis modelis skirtas specifikacijos informacijai išsaugoti, kad sukauptą informaciją būtų galima panaudoti vėlesnėje sistemos projektavimo stadijoje. Pagrindinės dalykinės srities esybės yra duomenų šaltinis, rezultatas, duomenų srautas, funkcija, etapas ir būsena. Duomenų šaltiniai turi tarpusavio ryšius, ir ryšius su rezultatais, apdorojami etapais ir gali turėti kelias būsenas. Tiek duomenų šaltinis, tiek rezultatas turi savo atributus, ir abu galima apibendrinti kaip duomenų esybę. Duomenų srautas gali būti tarp duomenų šaltinių arba duomenų šaltinio ir rezultato. Rezultatus formuoja funkcijos, kurios priskiriamos aktoriams arba aktorių grupėms, aktorius taip pat gali priklausyti aktorių grupei. Duomenų šaltinio apdorojimo etapas turi savo atributus bei funkcijas, kurias atlikus pereinama nuo vieno etapo prie kito. Būsenos apibūdina duomenų šaltinių kaitas, taip pat gali turėti būsenas esančias prieš ir po.



1.2 pav. ODRES metodo metamodelio schema

1.4. Tyrimo objekto naudotojų analizė

Vartotojų aibė yra visi asmenys dalyvaujantys informacinės sistemos kūrimo procese. Vartotojų tipai ir savybės pateikiami 1.3 lentelėje. Kuriant informacines sistemas užsakovas visada yra suinteresuotas, kad sistema būtų sukurta kuo greičiau, kuo pigiau ir kuo geriau, kad atitiktų

užsakovo poreikius. Visi trys faktoriai priklauso ne tik nuo sistemos kūrėjų, bet ir nuo įvairių išorinių faktorių. Tačiau sukurta informacinė sistema puikiai atitiks užsakovo poreikius jeigu analitikas surinks visus reikalavimus ir juos tinkamai dokumentuos, projektuotojas suprojektuos sistemą pagal paruoštą specifikaciją, o programuotojas realizuos sistemą pagal paruoštą projektą. Vis dėl to visada vyrauja žmogiškasis faktorius ir niekas nėra apsaugotas nuo klaidų.

1.3 lentelė. Vartotojų tipai ir savybės

Vartotojo tipas	Savybės
Analitikas	Bendradarbiauja su užsakovu arba jo atstovais. Aiškinasi kuriamos informacinės sistemos kontekstą. Nustato reikalavimus kuriamai sistemai ir juos dokumentuoja. Ruošia kuriamos informacinės sistemos reikalavimų specifikaciją
Projektuotojas	Projektuoja informacinę sistemą pagal pateiktą specifikaciją. Ruošia informacinės sistemos projektą
Programuotojas	Realizuoja informacinę sistemą pagal pateiktą projektą. Atlieka informacinės sistemos programavimo ir diegimo darbus
Testuotojas	Testuoja sukurtą informacinę sistemą. Testavimo atvejai sudaromi remiantis informacinės sistemos reikalavimų specifikacija
Užsakovas	Asmuo arba įstaiga, užsakanti ir apmokanti už informacinės sistemos kūrimą

Kuriant informacines sistemas svarbiausia, kad sukurta sistema atitiktų visus užsakovo norus ir lūkesčius. Todėl labai svarbu, kad analitikas tiksliai nustatytų ir dokumentuotų visus užsakovo reikalavimus. Remiantis analitiko parašyta reikalavimų specifikacija projektuotojas projektuoja informacinę sistemą. Pagal paruoštą projektą ir reikalavimų specifikaciją programuotojas realizuoja sistemą, o testuotojas ją testuoja. Pateikus realizuotą sistemą užsakovui ir jam pastebėjus, kad trūksta tam tikro funkcionalumo, arba kažkas netinkama, keičiant reikalavimus turi būti keičiamas ir projektas, tuo pačiu ir papildomai vykdomi programavimo darbai bei testavimas. Viso to pasekoje sistemos kurimo kaštai išauga, taip pat pailgėja ir kūrimo laikas. Kadangi kuriant informacines sistemas ir remiantis *ODRES* metodu nustatomi sistemos visi informaciniai srautai, jų šaltiniai ir rezultatai, galimos būsenos, tai pritaikius *ODRES* metodą *UML* aplinkai būtų galima išvengti nepilnai sudaromos reikalavimų specifikacijos, kadangi galima nustatyti visus reikiamus duomenis ir jų apdorojimo variantus, ir sutaupyti tiek laiko, tiek finansines sanaudas informacinių sistemų kūrimo procese. Vartotojų tikslai ir problemos aprašytos 1.4 lentelėje.

1.4 lentelė. Vartotojų tikslai ir problemos

Vartotojo tipas	Tikslai	Problemos
Analitikas	Paruošti informacinės sistemos specifikaciją, kuo tiksliau ir pilniau aprašant reikalavimus pagal vartotojo poreikius	Paruošiama netiksli informacinės sistemos specifikacija, nenustatant visų vartotojo poreikių, specifikacija papildoma realizavus sistemą
Projektuotojas	Suprojektuoti sistemą atitinkančią reikalavimų specifikaciją	Suprojektuojama sistema neatitinkanti užsakovo lūkesčių, koreguojamas projektas pagal kintančius reikalavimus specifikacijoje
Programuotojas	Realizuoti sistemą pagal pateiktą projektą	Pakitus reikalavimams ir projektui, realizuota sistema turi būti koreguojama
Testuotojas	Patikrinti sistemos veikimą, ar ji atitinka visus specifikacijoje aprašytus informacinės	Vykdomi papildomi testavimo atvejai, kad būtų patikrintas sistemos veikimas pagal naujus

	sistemos reikalavimus	reikalavimus
Užsakovas	Gauti informacinę sistemą atitinkančią visus poreikius	Užsakyta informacinė sistema neatitinka užsakovo lūkesčių

1.5. Esamų problemos sprendimo metodų analizė

1.5.1. ODRES metodo sprendimų analizė

Kuriant informacines sistemas, norint struktūrizuoti, planuoti ir kontroliuoti kūrimo procesą, naudojamos įvairios informacinių sistemų kurimo metodikos. Kiekviena informacinių sistemų kūrimo metodika turi savitą kūrimo procesą ir modeliavimo kalbą. Modeliavimo kalba yra grafinių elementų ir taisyklių rinkinys, skirtas pavaizduoti informacijos, sistemos ar žinių struktūrą modeliuojamoje srityje. Kadangi darbo tema yra „*ODRES* metodo realizavimo naudojant standartizuotas modeliavimo kalbas galimybių tyrimas“, o darbo tikslas yra ištirti ir sudaryti *ODRES* metodo pritaikymo galimybes pasirinktai modeliavimo kalbai ir jos modeliavimo įrankiui, buvo ieškomi sprendimai ir darbai, kuriuose būtų aprašytas šio metodo pritaikymas arba pritaikymo galimybės. Visgi *ODRES* metodas nėra lengvai pritaikomas, nes turi savitą notacijų rinkinį. Šio metodo pagrindą sudaro informaciniai srautai ir jų specifikacija. Buvo surasti keli sprendimai pritaikant *ODRES* metodą *CASE* įrankiams, sudaryta informacinių srautų specifikacija buvo pritaikyta informacinės sistemos projektavimo uždaviniams spręsti: kompiuterizuotų darbo vietų projektavimui, statinio duomenų modelio projektavimui, vartotojo sąsajos projektavimui. Visi surasti metodo pritaikymai buvo pasirinkti todėl, nes juose remiamasi pagal *ODRES* metodą sudaryta specifikacija ir *ODRES* metodo metaduomenų bazę, esant reikalui papildant ją reikalingais elementais ar atributais. Taip pat buvo analizuojamos modeliavimo kalbos ir įrankiai pasirinktinai, kuriems būtų galima pritaikyti *ODRES* metodą.

Magistro darbe „Automatizuotas internetinių IS vartotojų sąsajų kūrimas“, kurį atliko Marius Pečiulaitis 2006 metais, buvo realizuotas grafinės vartotojo sąsajos generatoriaus prototipas duomenų bazėje saugomą vartotojo sąsajos specifikaciją transformuojantis į *XML* (angl. *Extensible Markup Language*) failus, o šios informacijos interpretavimui ir vartotojo sąsajos formavimui generuojantis taisyklių rinkinius *XSLT* (angl. *Extensible Stylesheet Language Transformations*) faile [4]. Prototipas sukurtas naudojantis *Visual Fox Pro* duomenų bazių valdymo sistema ir *Microsoft Visio* įrankį. *ODRES* metodo metabazės schema buvo papildyta keletu elementų ir atributų, kuriais aprašoma *HTML* (angl. *HyperText Markup Language*) kalbos žymės, jų parametrai, taip pat grafikos elementai ir jų dalys. Naudojantis sukurtu prototipu dalykinės srities kompiuterizavimo, reikalavimų specifikavimo ir projektavimo etapai atliekami pagal *ODRES* metodo procesą. Pradžioje nustatomas dalykinės srities kontekstas, tam sudaromas funkcijų hierarchijos modelis. Tada nustatomi duomenų šaltiniai ir rezultatai. Sudaromas duomenų srautų modelis, kuriame matoma kaip duomenų šaltinis siejasi su kitais duomenų šaltiniais ar rezultatais. Toks modelis panaudojamas sudarant navigacijos tarp langų medį arba informacinės sistemos meniu struktūrą. Iš nagrinėjamo modelio nustatomi duomenų šaltiniams reikalinga informacija, taip pat iš kurių duomenų šaltinių koks rezultatas gaunamas. Pagal duomenų šaltinius ir rezultatus sudaromas duomenų bazės modelis. Vėliau sudaromi duomenų šaltinių apdorojimo etapų modeliai, kurie parodo duomenų šaltinių apdorojimo procesus. Iš šių modelių jau galima numatyti, kurie etapai turi būti kompiuterizuojami. Tuomet darbo metu sukurtu vartotojo sąsajos projektavimo įrankiu, kuris realizuotas *Microsoft Visio 2002* įrankio pagrindu, sudaromi eskiziniai vartotojo sąsajos modeliai, pradiniai modeliai sugeneruojami automatiškai, vėliau juos galima koreguoti rankiniu būdu. Sudaryti modeliai išsaugomi kaip atskiri dokumentai, o taip pat visa informacija išsaugoma ir iš *ODRES* metodo duomenų bazėje. Galiausiai iš sukauptos informacijos gali būti generuojama vartotojo sąsaja, naudojantis darbo metu sukurtu įrankiu. Grafinės vartotojo sąsajos generatorius iš metaduomenų bazės pagal vartotojo užduotus parametrus atrenka informacijos aibę, kurios pagrindu sugeneruojami vartotojo sąsajos langai, vartotojas parenka pažymėdamas varnele formas ir jų dalis, kurioms generuos sąsają. Viską atlikus naršyklėje atidarius sugeneruotus *XML* duomenų rinkinius parodoma galutinai suformuota vartotojo sąsaja. Sukurtas vartotojo sąsajos generavimo prototipas gerai išnaudoja *ODRES* metodo procesą,

informacinės sistemos pradinė vartotojo sąsaja gali būti sukurta pakankamai greitai, visdėlto prototipas sukurtas naudojant pasenusius ir jau nebeplėtojamus įrankius, kurie nėra tinkami dabartinei rinkai, o ir pats prototipas nebevystomas.

Antrasis *ODRES* metodo pritaikymas aprašytas straipsnyje „Duomenų modelio automatizuoto sudarymo prototipo funkcionalumo tyrimas“ [5]. Tyrimas buvo skirtas informacinės sistemos statinio duomenų modelio sudarymui skirto prototipo ir jo funkcionalumo analizei. Prototipas realizuotas *Visual Basic for Applications* programavimo kalba, grafinės vartotojo sąsajos elementams sukurti pasinaudota *Microsoft Visio 2000* galimybėmis, sistemos prototipas funkcionuoja šio programinio paketo aplinkoje. Darbui su informacinių srautų specifikacijos metaduomenų baze ir jos turiniu naudojama *Microsoft Access 2000* duomenų bazė. Šis prototipas yra vienas *CASE* modulis, skirtas kompiuterizuojamos informacinės sistemos projektavimui. Minėto *CASE* pagrindas yra *ODRES* - funkcinių reikalavimų specifikavimo metodas, kuriuo remiantis sudaroma informacijos srautų specifikacija. Pagrindiniai informacijos srautų specifikacijos objektai yra duomenų šaltiniai ir rezultatai, kurie atitinka į sistemą įeinančius bei išeinančius informacinius srautus. Visa informacinių srautų specifikacijos informacija saugoma saugykloje sudarytoje pagal *ODRES* metodo metabazės schemą. Pradžioje nustatomi duomenų šaltiniai ir rezultatai, paruošiamas esybių ryšių modelis. Tada sudaromas duomenų srautų modelis, kuriame matoma kaip siejasi duomenų šaltiniai arba duomenų šaltiniai ir rezultatai. Prototipo darbinis procesas prasideda *MS Visio* aplinkoje suaktyvinus šabloną esybių ryšių modeliavimui. Tada galima pasirinkti duomenų šaltinius iš paruoštos metabazės. Sistema patikrina ar duomenų šaltiniai susieti duomenų srautu, ir juos galima suintegruoti, priešingu ateju reikia rinktis kitus susietus šaltinius. Sistema įsimena duomenų šaltinių sąrašą ir nuskaičius iš duomenų bazės atvaizduoja duomenų šaltinių esybių ryšių modelius. Esybės taip pat gali būti įtraukiamos ir rankiniu būdu. Tuomet esybės patikrinamos ar jas prasminga integruoti, ir esant teigiamam atsakymui jos suintegruojamos. Taip sudaromas informacinės sistemos statinis duomenų modelis tiriamo prototipo pagalba. Detalus IS duomenų modelio integravimo algoritmo aprašas pateiktas paskelbtuose straipsniuose [6]. Naudojantis prototipu efektyviai ir greitai galima suprojektuoti duomenų bazę, visgi informacinė sistema susideda iš daugiau elementų, kurie turi būti suprojektuoti, o šis prototipas nepritaikytas kitų informacinės sistemos elementų projektavimui. Taip pat prototipas realizuotas programinės įrangos kūrėjų nebevystomais įrankiais ir yra sunkiai pritaikomas bei suderinamas su šiandieniniais įrankiais.

Trečiasis *ODRES* metodo pritaikymo sprendimas aprašytas straipsnyje „Approach for IS workspace design based on output driven requirements specification“, straipsnio autoriai Tomas Danikauskas ir Rimantas Butleris [7]. Straipsnyje nagrinėjamos problemos, susijusios su informacinės sistemos darbo vietos projektavimu, šiuo atveju darbo vietos projektavimą galima įvardinti kaip informacinės sistemos grafinės vartotojo sąsajos, meniu sistemos projektavimą. Kompiuterizuojama darbo vieta projektuojama remiantis trimis proceso žingsniais:

- Pradžioje sudaromas funkcijų sarašas kiekvienam aktoriui susijusiam su kuriama sistema ar analizuojama situacija. Tada sudaroma funkcijų ir aktorių matrica, kad būtų galima nustatyti pasikartojančias aktorių funkcijas.
- Antrajame žingnyje sudaroma funkcijų hierarchija kiekvienam aktoriui, surūšiuojamos aukščiausio lygio ir visos pavaldžios funkcijos.
- Trečiajame žingsnyje projektuojama aktoriaus darbo vieta remiantis sudaryta funkcijų hierarchija. Projektuojant reikia laikytis kelių taisyklių: meniu punktas gali būti panaikintas, jeigu turi tik vieną žemesnio lygio meniu punktą, panaikinus aukštesniojo lygio meniu punktą, žemesnio lygio meniu punktas užima jo vietą. Taip pat meniu gali būti sudaromas aktorių grupėms juos grupuojant pagal pasirinktus kriterijus.

Konceptualus darbo vietos projektavimo įrankio prototipas buvo dalinai sukurtas naudojantis *Microsoft Visio 2002* galimybėmis. Sudarytą funkcijų hierarchiją buvo galima išsaugoti ne tik kaip įrankiu sukurtą paveikslą, bet ir išsaugoti į *MS Access* saugyklą. Visgi galutinės darbo vietos generavimas nebuvo realizuotas, taip pat baigtas prototipo kūrimas. Naudojantis šio sprendimo idėjomis galima sukurti labai didelių informacinių sistemų meniu sistemos prototipą, sumažinant klaidos galimybes iki minimumo, kadangi meniu struktūra sudaroma remiantis sistemos aktorių funkcijų hierarchija.

1.5.2. Modeliavimo kalbų analizė

Modeliavimo kalba gali būti grafinė arba tekstinė. Grafinė modeliavimo kalba naudoja diagramų modeliavimo techniką su grafiniais objektais ir tekstinėmis jų anotacijomis, kurie vaizduoja konceptus ir linijas jungiančias simbolius ir taip atvaizduoja ryšius ir apribojimus tarp įvairių elementų. Tekstinė modeliavimo kalba naudoja standartinius raktažodžius jungiančius su parametrais ir įvairiais natūralios kalbos terminais ir frazėmis, kad būtų suformuojami suprantami išsireiškimai tiek žmogui, tiek kompiuteriui. Kadangi turimas *ODRES* metodas naudoja savitą nestandartizuotą grafinę notaciją, buvo pasirinktos trys populiarios grafinio modeliavimo kalbos analizei: *IDEF*, *BPMN*, *UML*. Atlikus šių modeliavimo kalbų analizę bus pasirinkta viena, kurios pagalba bus realizuojamas *ODRES* metodas.

1.5.2.1. IDEF modeliavimo kalba

IDEF (angl. Integration DEFinition) - modeliavimo metodų grupė, skirta modeliuoti informacines sistemas ir programinės įrangos veikimą [8]. *IDEF* metodai naudojami įvairaus tipo modeliavimui, nuo funkcinio modeliavimo iki duomenų modeliavimo, taip pat simuliacijų, ar objektiškai orientuotų sistemų modeliavimui. Ši modeliavimo metodų grupė sukurta Jungtinių Amerikos Valstijų oro pajėgų, ir dar dabar jų naudojama, kaip ir JAV gynybos departamento. Geriausiai žinomi ir atpažįstami *IDEF* komponentai yra *IDEF0*, skirta funkciniam modeliavimui, ir *IDEF1X*, skirta duomenų modeliavimui. Visi *IDEF* modeliavimo metodai pateikiami 1.5 lentelėje.

1.5 lentelė. IDEF modeliavimo metodai

Metodas	**Paskirtis**
IDEF0	Funkcinis modeliavimas
IDEF1	Informacijos modeliavimas
IDEF1X	Duomenų modeliavimas
IDEF2	Simuliacijos modeliavimas
IDEF3	Procesų aprašymų nustatymas
IDEF4	Objektiškai orientuotas modelis
IDEF5	Ontologijų aprašų nustatymas
IDEF6	Loginio modelio sudarymas
IDEF7	Informacinių sistemų audito metodas
IDEF8	Vartotojo sąsajos modeliavimas
IDEF9	Scenarijais pagrįstos informacinės sistemos modelio sudarymas
IDEF10	Realizuojamos sistemos architektūros modeliavimas
IDEF11	Informacinių artefaktų modeliavimas
IDEF12	Organizacijos modeliavimas
IDEF13	Trijų schemų konvertavimo modelis
IDEF14	Tinklo modelis

Iki 1995 metų buvo pilnai išvystyti *IDEF0*, *IDEF1X*, *IDEF2*, *IDEF3*, *IDEF4* metodai. Po 1995 buvo realizuoti patikimam informacinių sistemų ir organizacijų modeliavimui *IDEF6*, *IDEF8*, *IDEF9* ir *IDEF14* metodai. Likę metodai nebuvo išvystyti daugiau negu jų pradinės sąvokos. Plačiau bus nagrinėjami funkcinio modeliavimo (*IDEF0*), duomenų modeliavimo (*IDEF1X*), procesų aprašymų nustatymo (*IDEF3*) metodai, kurių pagalba būtų galima realizuoti *ODRES* metodo procesą.

IDEF0 yra funkcinio modeliavimo metodas skirtas modeliuoti organizacijos ar sistemos veiklas, įvykius ar sprendimus [9]. *IDEF0* susideda iš grafinės modeliavimo kalbos (sintaksės ir semantikos) ir nurodymų kaip modeliuoti šiuo metodu. Metodas padeda palengvinti organizuojant sistemos analizę bei bendravimą tarp analitiko ir užsakovo dėl lengvai suprantamos grafinės notacijos. *IDEF0* naudojamas pavaizduoti duomenų srautus, sistemos valdymą ir sistemos gyvenimo

ciklo funkcinius procesus. Taip pat suteikia galimybę naudojant grafinę notaciją pavaizduoti įvairius verslo, gamybos ar kitų rūšių įmonių operacijas naudojant įvairius detalumo lygius. Šis metodas yra gerai ištestuotas per daugelį metų naudojant valstybinėse institucijose ir privataus verslo sektoriuose. *IDEF0* gali būti naudojamas modeliuoti įvairias automatizuotas ir ne-automatizuotas sistemas. Naujoms sistemoms jis gali būti naudojamas pradžioje sistemos reikalavimų ir funkcijų specifikavimui, ir tik po to modeliavimui, kaip veikia sistema, kaip jos funkcijos realizuoja reikalavimus. Jau sukurtom informacinėm sistemom, *IDEF0* gali būti naudojamas esamų funkcijų analizei atlikti. Naudojant *IDEF0* metodą kuriant sistemas gaunamas rezultatas, kuris susideda iš hierarchinės struktūros, kurią sudaro modeliuojamos diagramos, ir ryšių tarp šių modeliuojamų diagramų ir jų elementų. Du pagrindiniai modeliuojami elementai yra funkcijos ir duomenys bei objektai, kurie sąveikauja tarp šių funkcijų.

IDEF1X yra duomenų modeliavimo metodas skirtas modeliuoti semantinius duomenų modelius, reliacines duomenų bazes [9]. Metodas naudojamas gaminti grafinius informacijos modelius, kurie vaizduoja informacinės sistemos ar kitokios aplinkos informacijos struktūrą ir semantiką. Pagrindiniai *IDEF1X* standarto tikslai yra suteikti:

- priemones visiškai suprasti ir analizuoti organizacijų duomenų šaltinius;
- bendrąsias priemones atvaizduoti ir paaiškinti duomenų sudėtingumą;
- techniką pavaizduoti bendrą vaizdą duomenų, reikalingų organizacijos veikloms;
- priemones atvaizduoti duomenims, kurie gali būti validuojami vartotojų ir vėliau transformuojami į fizinę duomenų bazės struktūrą;
- techniką gauti duomenų modelius iš jau esamų organizacijos duomenų šaltinių.

IDEF1X principinis tikslas yra palaikyti integravimą. Požiūris į integravimą sutelkiamas ties konceptualia schema. Konceptuali schema pateikia vieną integruotą organizacijos duomenų modelį, kuris nėra priklausomas nuo to kaip duomenys yra saugomi ar prieinami organizacijoje. Konceptualios schemos pagrindinis tikslas yra pateikti nuoseklų vientisą apibrėžimą reikšmių ir tarpusavio ryšių tarp duomenų, kuriais gali būti naudojamasi duomenų integralumui išlaikyti. Konceptuali schema privalo turėti tris svarbias charakteristikas:

- Vientisumą visoje verslo infrastruktūroje ir teisingumą tarp visų organizacijos sistemų;
- Praplečiamumą, kad nauji duomenys galėtų lengvai būti įtraukiami, be ankstesnių duomenų ištrynimo ar praradimo;
- Transformuojamumą, tiek vartotojo peržiūroms, tiek skirtingoms duomenų saugykloms ir struktūroms.

Informacinių sistemų projektavime naudojant *IDEF1X* metodą naudojamas trijų schemų projektavimo būdas, kuriame konceptualus modelis yra raktas į vientisą duomenų struktūrą. Šios trys schemos yra: išorinės schemos skirtos vartotojams peržiūrėti, konceptuali schema integruoja išorines schemas, vidinė shema, kuri apibrėžia fizines duomenų struktūras. Pagrindinė konceptuali schema apibrėžia konceptų ontologiją taip kaip naudotojai galvoja apie dalykinę sritį. Fizinė schema apibrėžia vidinius duomenų formatus duomenų bazėje, o išorinės schemos nustato kaip duomenys atvaizduojami programų.

IDEF3 yra procesų aprašymų nustatymo metodas skirtas modeliuoti verslo procesus papildant *IDEF0* metodą [9]. *IDEF3* metodas yra scenarijais paremtas procesų srauto nustatymo metodas, kuriuo aprašoma kaip tam tikra sistema veikia. *IDEF3* metodas suteikia galimybę pavaizduoti abu: procesų srautą aprašant ryšius tarp įvykių specifinio scenarijaus kontekste, objektų būsenų perėjimus aprašant galimas būsenas ir sąlygas. Šis metodas buvo sukurtas vaizduoti įvykių sekas, kas yra standartinis mechanizmas norint pavaizduoti procesą ar tam tikrą situaciją. Pagrindinis *IDEF3* tikslas yra suteikti struktūrizuotą metodą, kuriuo dalykinės srities ekspertas galėtų pavaizduoti žinias apie tam tikros sistemos ar organizacijos operacijas. Poreikiai dėl kurių išvystytas *IDEF3*:

- Pagreitinti verslo sistemų modeliavimo procesus;
- Suteikti duomenų gyvavimo ciklo informacijos aprašymo mechanizmą;
- Palaikyti projektų valdymo technikas naudojant automatizuotus įrankius;

- Suteikti konceptus, sintaksę ir procedūras aprašant sistemos reikalavimus;
- Suteikti galimybę apjungti skirtingus *IDEF* modeliavimo šeimos metodus.

IDEF modeliavimo metodų grupė buvo išvystyta kaip standartinių metodų rinkinys verslo procesų dokumentavimui ir analizavimui. *IDEF* yra puikiai dokumentuotas standartas, kuris gali būti naudojamas informacinių sistemų modeliavimui. Dokumentacija yra nemokama ir laisvai prieinama. *IDEF* pagalba galima modeliuoti įvairias sistemas bet kokiame kontekste, turi standartizuotą notaciją. Asmuo, nesusidūręs su *IDEF* notacija, gali būti išmokytas skaityti ir suprasti diagramas greičiau nei per valandą laiko. Visdėlto *IDEF* metodų grupė vystyta buvo seniau negu *CASE* įrankiai buvo pradėti kurti todėl darosi sudėtinga modeliuoti su šiuolaikiniais įrankiais. *IDEF* labiau tinkamas verslo analizei negu kuriamų sistemų modeliavimui, be to kartais sunku integruoti skirtingas *IDEF* technikas į vieną visumą. Dauguma programinės įrangos kūrimo įrankių nebepalaiko *IDEF* notacijos, dėl to darosi sunku šią modeliavimo kalbų grupę ir jos notaciją taikyti sistemų kūrime.

1.5.2.2. BPMN modeliavimo kalba

BPMN (angl. Business Process Model and Notation) yra grafinio modeliavimo standartas skirtas specifikuoti verslo procesus naudojant verslo procesų modeliavimo diagramas [9]. *BPMI* (angl. Business Process Management Initiative) sukūrė *BPMN*, nuo 2005 metų šią notaciją tobulino *OMG* (angl. Object Management Group), kai šios dvi organizacijos susijungė. Naujausia *BPMN* versija yra 2.0 išleista 2011 metų kovą. Su versijos pasikeitimu buvo pakoreguotas ir pavadinimas iš „*Business Process Model Notation*“ į „*Business Process Model and Notation*“, tuo pačiu su nauja versija *BPMN* kalba apibrėžė ne tik notaciją ir jos informaciją, bet ir procesų veikimo semantiką.

Pagrindinis *BPMN* sukūrimo tikslas yra suteikti notaciją, kuri yra gerai suprantama visiems verslo dalyviams, nuo verslo procesų analitiko, kuris kuria pirminius verslo procesų projektus, iki verslo sistemų programuotojo, kuris kuria technologijas skirtas palaikyti apibrėžtiems verslo procesams. O taip pat ir visiems verslo dalyviams, kurie įgyvendins visus verslo procesus. Taigi *BPMN* buvo sukurtas kaip tiltas, jungiantis plyšį tarp verslo procesų kūrimo iki jų realizavimo.

BPMN 2.0 standartą sudaro 3 pagrindinės dalys:

- procesai: vaizduoja veiklos procesus, įvykius ir pranešimus;
- kooperacijos: rodo, kaip procesą įgyvendina bendradarbiaujantys proceso dalyviai ir detalius jų pokalbius;
- choreografija: rodo pranešimus ir informacinius srautus, kuriuos dalyviai siunčia vieni kitiems.

BPMN 2.0 turi 3 diagramas:

1. procesų diagramą: rodo veiklos procesus, pranešimus ir įvykius. Procesų diagramos būna kelių rušių:
   - Privatūs vidiniai ne vykdomi verslo procesai;
   - Privatūs vidiniai vykdomi verslo procesai;
   - Vieši procesai;
2. bendradarbiavimo diagramą, kuri rodo, kaip procesą įgyvendina bendradarbiaujantys proceso dalyviai, ir detalius jų pokalbius;
3. choreografijos diagramą: rodo duomenų ir pranešimų srautus.

BPMN yra sukurtas palaikyti verslo procesų modeliavimo konceptus. Tai reiškia kad kitokių sričių ar tipų modeliavimas, kurį atlieka organizacijos, susijęs su verslo tikslais nepatenka į *BPMN* modeliavimo rėžius. Taigi toliau pateikti aspektai nepriklauso *BPMN* modeliavimo sričiai:

- Organizacijos struktūros ir resursų modeliavimas;
- Funkcijų modeliavimas;
- Duomenų ir informacijos modeliavimas;
- Organizacijos strategijų modeliavimas;
- Veiklos taisyklių modeliavimas.

BPMN apima tris pagrindines procesų modeliavimo sritis: privačius procesus, viešus procesus ir choreografijas. Naudojant šiuos tris *BPMN* diagramų potipius galima sukurti įvairių rūšių diagramas. Toliau pateikiami verslo procesai, kuriuos galima modeliuoti remiantis *BPMN* notacija:

- Aukšto lygio verslo procesus;
- Detalius verslo procesus;
- Esamus (angl. „as is“) arba senus verslo procesus;
- Būsimus (angl. „to be“) arba naujus verslo procesus;
- Dviejų ar daugiau verslo dalyvių tikėtiną tarpusavio bendravimą - choreografiją;
- Detalius vidinius verslo procesus pridedant bendradarbiavimą ir komunikavimą su išorinėm esybėm;
- Dviejų ar daugiau detalių procesų tarpusavio sąveikas;
- Detalius verslo procesų ryšius su choreografija;
- Du ar daugiau išorinius procesus;
- Išorinių procesų ryšys su choreografija;
- Du ar daugiau detalius verslo procesus sąveikaujant per choreografiją.

1.5.2.3. UML modeliavimo kalba

UML yra modeliavimo kalba skirta informacinių sistemų modeliavimui, kuri suteikia standartizuotą būdą suprojektuoti sistemą [10]. *UML* buvo sukurta ir tobulinama Grady Booch, Ivar Jacobson ir James Rumbaugh 1994-1996 metais. 1997 metais *OMG* priėmė *UML* kaip standartą, ir nuo tada *UML* modeliavimo kalba buvo tobulinama šios organizacijos. *UML* modeliavimo kalba buvo sukurta remiantis Booch metodu, *OMT* (angl. Object-modeling technique) technika ir *OOSE* (angl. Object-oriented software engineering) kalba, visa tai apjungus į vientisą modeliavimo kalbą. 2000 metais *UML* kalba buvo patvirtinta kaip *ISO* (angl. International Organization for Standardization) standartas. Nuo tada ji periodiškai atnaujinama, kad atitiktų naujausius reikalavimus. *UML* kalba yra *OMG* dažniausiai naudojama specifikacija, kurios pagalba modeliuojama informacinių sistemų struktūra, veikimas, elgsena, architektūra, taip pat verlso procesai ir duomenų struktūros.

UML kalba nėra programinės įrangos ar informacinių sistemų kūrimo metodas, bet buvo sukurta, kad būtų suderinama su objektiškai orientuotos programinės įrangos kūrimo metodais. *UML* paskirtis yra suteikti informacinių sistemų architektams, inžinieriams ir programuotojams įrankius analizei, projektavimui ir realizavimui sistemų, kurios grindžiamos programine įranga, o taip pat ir verslo ar panašių procesų modeliavimui. Vienas iš pagrindinių tikslų yra pagerinti objektiškai orientuotų programinės įrangos modeliavimo įrankių galimybes modeliuojant įvairių pramonės sričių organizacijas ir jų informacines sistemas. Visgi įgalinant prasmingą modelių informaciją tarp įrankių ir skirtingų modelių, reikalingas semantikos ir notacijos suderinimas. Toliau pateikiami reikalavimai *UML* kalbai:

- Formalus *MOF* (angl. Meta Object Family) paremtas metamodelio apibrėžimas, kuris apibrėžia abstrakčią *UML* sintaksę. Abstrakti sintaksė nustato *UML* modeliavimo konceptų rinkinį, jų atributus ir ryšius, o taip pat ir taisykles kaip jungti tuos konceptus kuriant dalinius ar pilnus *UML* modelius.
- Kiekvieno modeliavimo koncepto detalus semantinis paaiškinimas. Semantika apibrėžia, kaip *UML* konceptai bus realizuojami kompiuteryje, nepriklausomai nuo naudojamų technologijų.
- Specifikacija žmogui suprantamų notacijos elementų, skirtų vaizduoti skirtingus *UML* modeliavimo konceptus ir taisykles, kaip kombinuoti juos skirtingų diagramų rūšių įvairovėje, modeliuojant sistemos įvairius aspektus.

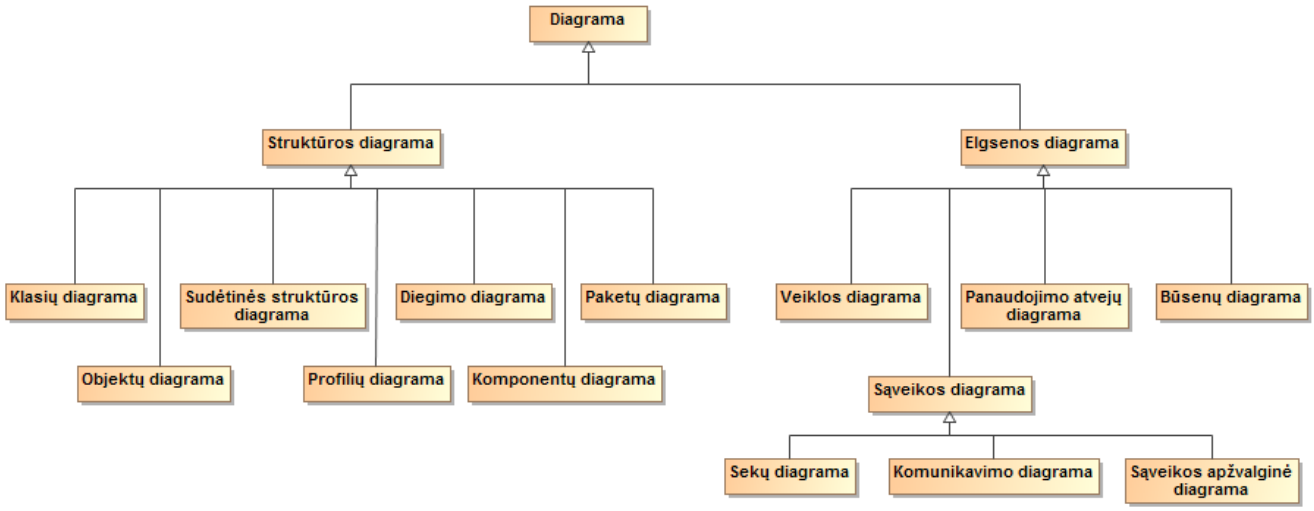
UML yra kalba su plačia taikymo sritim, kuri apima didelį ir įvairiapusį taikymo sričių rinkinį. Ne visos jos modeliavimo galimybės yra reikalingos įvairiose taikymo srityse. Tai reiškia, kad modeliavimo kalbos struktūra yra modulinė, su galimybe pasirinkti tas kalbos dalis, kurios yra reikalingos modeliuojant. Visdėlto toks lankstumas padidina tikimybę, kad keli skirtingi *UML* modeliavimo įrankiai palaikys skirtingas kalbos dalis, kas gali įtakoti nesuderinamumo problemas. Vadinasi *UML* kalba reikalauja pusiausvyros tarp modulariškumo ir skirtingų įrankių suderinamumo.

Labai svarbu suprasti skirtumą tarp *UML* modelio ir diagramų rinkinio vaizduojančio sistemą. Diagrama yra sistemos modelio dalies grafinis atspindys. Diagramų rinkinys nebūtinai turi

padengti visą modelį ir diagramos pašalinimas nepakeičia modelio prasmės. *UML* diagramos vaizduoja du skirtingus sistemos modelio aspektus:

- Statinis vaizdas: akcentuojama statinė sistemos struktūra naudojant objektus, atributus, operacijas ir ryšius.
- Dinaminis vaizdas: akcentuojama dinaminė sistemos elgsena vaizduojant bendradarbiavimą tarp objektų ir objektų vidinių būsenų pasikeitimus.

UML kalboje yra keliolika įvairių tipų diagramų, kurios yra suskirstytos į dvi kategorijas: struktūros ir elgsenos. Dalis diagramos vaizduoja struktūrinę informaciją, o likusios vaizduoja tam tikrus elgsenos tipus, apimant ir tarpusavio objektų sąveikos aspektus. *UML* diagramos ir jų struktūra pavaizduota 1.3 paveiksle. Diagramų sąrašas ir jų aprašymai pateikiami 1.6 lentelėje.



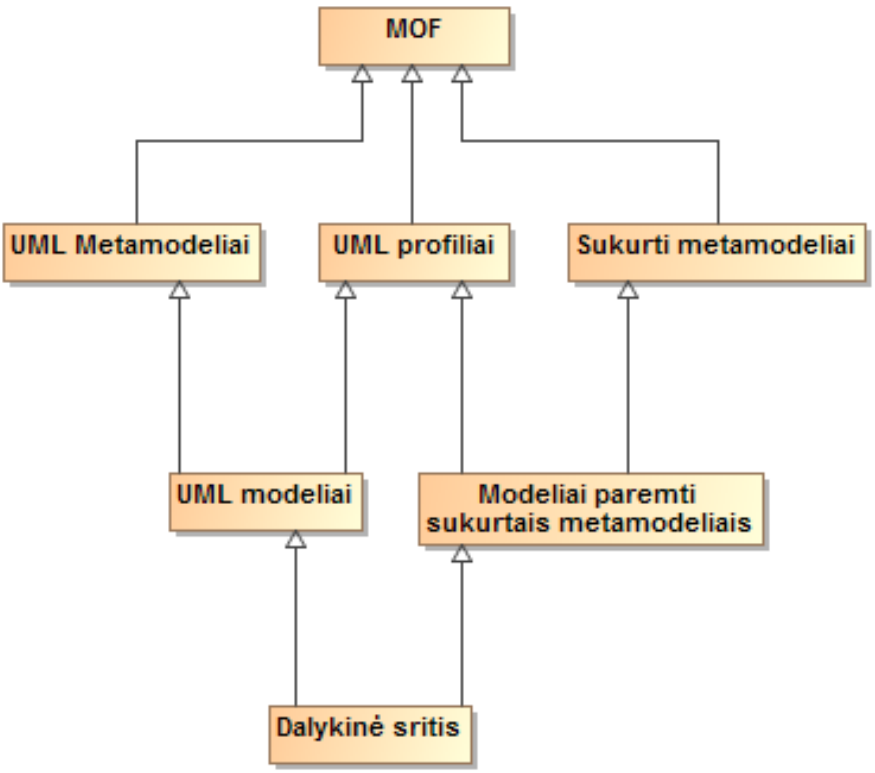
1.3 pav. UML diagramų tipų struktūros diagrama

1.6 lentelė. UML diagramos

Diagrama	Aprašymas
Klasių	Statinės struktūros diagrama, kurios pagalba aprašoma ir vaizduojama sistemos struktūra vaizduojant sistemos klases, jų atributus, operacijas ir ryšius tarp objektų.
Komponentų	Vaizduoja kaip programinės įrangos komponentai siejasi tarpusavyje ir formuoja didesnius komponentus.
Sudėtinės struktūros	Statinės struktūros diagrama su kuria vaizduojama vidinė klasių struktūra ir bendradarbiavimą tarp šių klasių.
Diegimo	Vaizduoja sistemos diegimo fizinius artifaktus ir mazgus.
Objektų	Vaizduoja pilną arba dalinį modeliuojamos sistemos struktūros vaizdą tam tikru laiko momentu.
Paketų	Vaizduoja ryšius tarp paketų, kurie sudaro sistemos modelį.
Profilių	Profilių diagrama vaizduoja metamodelio lygio stereotipus, jų sąsajas, apribojimus.
Veiklos	Vaizduoja darbo eigos veiklas ar įvykius pažingsniui, su pasirinkimo, iteracijų ir konkurencingumo galimybėmis. Jomis galima modeliuoti kompiuterinius arba organizacijos procesus.
Komunikavimo	Vaizduoja objektų ar jų dalių bendravimą.
Sąveikos apžvalginė	Vaizduoja sistemos kontrolinį srautą su mazgais, kuriuose gali būti vaizduojama, įdėta kita sąveikos diagrama.
Sekų	Sąveikos diagrama, kuri vaizduoja dviejų ar daugiau dalyvių proceso vidinę struktūrą, jo veikimą ir veikimo tvarką.

Būsenų	Vaizduoja sistemos, programinės įrangos ar verslo proceso būsenas ir šių pasikeitimus.
Panaudojimo atvejų	Vaizduoja sistemos naudotojo sąveiką su sistema, kur matomi sistemos naudotojų ryšiai su panaudojimo atvejais, kuriuose jie dalyvauja.

UML yra kalba skirta labai plačiai taikymo sričiai, turinti modelių rinkinį, kurį galima taikyti įvairiose dalykinėse srityse. *UML* kalbos metamodelis yra paremtas *MOF* (*angl. Meta-Object Facility*) *OMG* (*angl. Object Management Group*) grupės standartu, kuris skirtas modeliais paremtai sistemų inžinerijai. *MOF* yra metaduomenų valdymo sistema (framework) ir metaduomenų paslaugų rinkinys suteikiantis modelių vystymą ir sąveikas metaduomenimis paremtose sistemose. Sistemos naudojančios *MOF* standartą susideda iš modeliavimo ir kūrimo įrankių, duomenų bazių sistemų ir metaduomenų saugyklų. *UML* metamodelis sudarytas iš keturių sluoksnių, pavyzdinis paveikslas pateikiamas 1.4 paveiksle. Aukščiausiame sluoksnyje yra *MOF* meta-metamodelis, kuris apibrėžia abstrakčia sintaksę kaip turi būti sudaryta *UML*. Abstrakti sintaksė nustato *UML* modeliavimo konceptų rinkinį esantį antrame lygyje, jų atributus ir ryšius, tuo pačiu ir taisykles kaip jungiant konceptus sudaryti dalinius ar pilnus *UML* modelius. Pagal antro lygio konceptus ir jų taisykles sudaromi *UML* modeliai, kurie matomi trečiajame modelių lygyje. Modeliai atspindi ketvirtą lygį, realų pasaulį, modeliuojamą dalykinę sritį.



1.4 pav. UML metamodelio sluoksnių struktūra

1.5.3. Modeliavimo įrankių analizė

Visoms trims modeliavimo kalboms, *UML*, *BPMN*, *IDEF*, buvo ieškomi modeliavimo įrankiai, kurių pagalba pasitelkus atitinkamą modeliavimo kalbą būtų galima realizuoti *ODRES* metodą. *IDEF* modeliavimo kalbos naudojimui skirtų šiuolaikinių modeliavimo įrankių rasti nepavyko, nei atvirojo kodo, nei mokamų licenzijuojamų įrankių. Šios kalbos modeliavimo įrankiai yra senesnių laikų, nelabai populiarūs ir nėra plačiai naudojami, dėl ko priimtas sprendimas nagrinėti *BPMN* ir *UML* modeliavimo kalbų įrankius. Kadangi šios abi modeliavimo kalbos yra kuriamosir

plėtojamos *OMG* organizacijos, tai didelė dalis modeliavimo įrankių palaikančių *UML* notaciją palaiko *BPMN* notaciją ir atvirkščiai. Buvo surasta nemažai šis kalbas palaikančių modeliavimo įrankių, kurie yra aktyviai naudojami, nuolatos palaikomi ir atnaujinami tų įrankių kūrėjų.

ODRES metodo realizavimo naudojant standartizuotas modeliavimo kalbas analizei buvo pasirinkti du įrankiai, *MagicDraw* ir *Visual Paradigm*, palaikantys *UML* ir *BPMN* modeliavimo kalbas [11] [12]. *MagicDraw* yra veiklos procesų, architektūros, programinės ir techninės įrangos modeliavimo įrankis skirtas veiklos ir programinės įrangos analitikams, programuotojams, programinės įrangos inžinieriams ir dokumentacijos rašytojams. Įrankis kuriamas *No Magic* kompanijos jau septynioliką metų, kompanijos būstinė yra Lietuvoje, Kauno mieste. *Visual Paradigm* yra programinės įrangos specifikavimo ir projektavimo įrankis skirtas greitiems programinės įrangos projektams. Įrankis vystomas *Visual Paradigm* kompanijos, kurios būstinė yra Honkonge.

Abu įrankiai buvo analizuojami iš *ODRES* metodo realizavimo modeliavimo kalbai galimybių perspektyvos. Buvo pasirinkti analizavimo kriterijai, kurie gali smarkiai įtakoti pritaikymo galimybes. Vienas iš pagrindinių aspektų, kad abu įrankiai palaiko 2.4 versijos *UML* kalbos notaciją, jais galima braižyti visas *UML* kalbos palaikomas diagramas. Kitas svarbus aspekatas, kad šie įrankiai turi ir *BPMN* modeliavimo kalbos 2.0 versijos palaikymą. Abu įrankiai suteikia galimybes sukurti specializuotus profilius pagal vartotojo norus, kas gali būti labai svarbu norint susikurti specializuotus stereotipus *UML* elementams. Taip pat leidžia generuoti ataskaitas, kurios yra paruoštos įrankių kūrėjų, arba yra galimybė generuoti pagal naudotojo paruoštą šabloną. Vis dėl to *MagicDraw* suteikia galimybę vykdyti modelių transformacijas, kas gali būti aktualu norint modelius panaudoti pakartotinai juos modifikuojant. Dar vienas svarbus dalykas, tai *MagicDraw* įrankyje yra galimybės pasirinkus vienokio tipo diagramą, joje naudoti kitokio tipo diagramos elementus, ko neleidžia daryti *Visual Paradigm*. Abu modeliavimo įrankiai turi praplėtimo galimybes, jiems galima sukurti papildinius pagal savo poreikius ir įdiegti, kad veiktų kartu su įrankiu. Kaip dauguma *UML* ir *BPMN* modeliavimo įrankių, taip ir šie du turi licenzijas, yra mokami. Pasirinkti įrankių vertinimo kriterijai ir įrankių palyginimas pateikiamas 1.7 lentelėje.

1.7 lentelė. Įrankių palyginimas pagal pasirinktus kriterijus

Kriterijus	MagicDraw	Visual Paradigm
Palaiko *UML* 2.4 standartą	Taip	Taip
Palaiko *BPMN* 2.0 standartą	Taip	Taip
Ataskaitų generavimas	Taip	Taip
Modelių transformacijos	Taip	Ne
Galimybės kurti profilius	Taip	Taip
Programinio kodo generavimo galimybės	Taip	Taip
Galimybė naudoti skirtingų diagramų elementus	Taip	Ne
Turi įrankio praplėtimo galimybes	Taip	Taip
Mokamas	Taip	Taip
Komandinio darbo galimybės	Taip	Taip
OO (angl. Object-oriented) projektavimas	Taip	Taip

1.5.4. Lyginamosios analizės apibendrinimas

Buvo išanalizuoti keli *ODRES* metodo pritaikymo variantai. Visi prototipai veikia efektyviai savo srities uždaviniams spręsti. Didžiausia esamų sprendimų problema, kad jie realizuoti programinės įrangos kūrėjų nebevystomais įrankiais, o visi sukurti prototipai irgi nėra vystomi prototipų kūrėjų. Taip pat *ODRES* metodo notacija turi savitą notacijų rinkinį, kuris nėra standartizuotas.

Buvo pasirinktos ir išanalizuotos trys gerai žinomos modeliavimo kalbos: *IDEF*, *BPMN* ir *UML*. Šių modeliavimo kalbų palyginimas pateikiamas 1.8 lentelėje. *IDEF* modeliavimo metodų grupė turi penkiolika skirtingų modeliavimo metodų, tačiau ne visi yra išvystyti, o tik keli pagrindiniai. *IDEF* panaudojimo sritis yra pakankamai plati, galima modeliuoti organizacijų ar duomenų struktūras, įvairius procesus. Visgi ši modeliavimo metodų grupė ir atskiri modeliavimo metodai nėra labai populiarūs, daugiausiai naudojami JAV karo pramonėje. Taip pat *IDEF* palaikymo neturi daug šiuolaikinių modeliavimo įrankių ir praplečiamumas nėra galimas. *BPMN* yra daugiausiai skirta įvairių procesų modeliavimui, neturint statinių organizacijos ar sistemos struktūrų modeliavimo galimybių. Tačiau *BPMN* yra pakankamai populiari modeliavimo kalba plačiai naudojama visame pasaulyje. Kadangi ši modeliavimo kalba populiari, tai turi ir gerą įrankių palaikymą. Dar vienas svarbus šios modeliavimo kalbos aspektas, kad ją galima praplėsti papildomais elementais, kurie nepažeidžia *BPMN* specifikacijos taisyklių. Visgi lyginti *BPMN* ir *UML* modeliavimo kalbas, galima sugretinant *BPMN* procesų diagramas su *UML* veiklos ar sekų diagramomis, kurių pagalba modeliuojami procesai neatsižvelgiant į statines struktūras, kurių *BPMN* kalboj modeliuoti negalima [14]. *UML* modeliavimo kalbos panaudojimo sritis yra labai plati, galima modeliuoti įvairias struktūras, įvairius procesus, objektų tarpusavio sąveikas. *UML*, kaip ir *BPMN* modeliavimo kalba, turi daug šiuolaikinių palaikomų įrankių, bei yra populiari ir plačiai naudojama visame pasaulyje. Taip pat *UML* kalba turi praplečiamumo galimybes, turi savo metamodelį, kurį galima papildyti naujais elementais nenusižengiant *UML* kalbos reikalavimams.

Po modeliavimo kalbų analizės nuspręsta, kad *ODRES* metodas ir jo procesas bus realizuojamas *UML* kalboje. *IDEF* atmesta dėl savo nepopuliarumo, prasto modeliavimo įrankių palaikymo ir dėl to, kad neturi praplėtimo galimybių, kas gali būti reikalinga neradus visų *ODRES* metodo elementų atitikmenų *IDEF* elementams. *BPMN* modeliavimo kalba atmesta dėl siauros savo taikymos srities. Joje trūksta struktūrų modeliavimo galimybių. *UML* modeliavimo kalba atitinka visus reikalavimus siekiant surasti *ODRES* metodo elementų atitikmenis ir pritaikyti *ODRES* metodo procesą modeliavimui šia kalba.

1.8 lentelė. Modeliavimo kalbų palyginimas

Kriterijus	IDEF	BPMN	UML
Populiarumas	Daugiausiai naudojama JAV karinių oro pajėgų ir gynybos departamento	Plačiai naudojama visame pasaulyje	Plačiai naudojama visame pasaulyje
Įrankių palaikymas	Nedaug, senesni	Daug įvairių	Daug įvairių
Panaudojimo sritis	Plati, tinka organizacijų, duomenų, procesų modeliavimui	Siaura, įvairių procesų modeliavimui	Plati, organizacijų, duomenų, procesų, sąveikos, struktūrų modeliavimui
Praplečiamumas	Nėra	Yra	Yra

Kadangi *IDEF* neturi šiuolaikinių modeliavimo įrankių pasirinkimo, buvo analizuojami įrankiai palaikantys *UML* ir *BPMN* kalbas. Pasirinkti ir nuodugniau išnagrinėti *MagicDraw* ir *Visual Paradigm* modeliavimo įrankiai. Atlikus įrankių analizę nuspręsta, kad siekiamas magistrinio darbo sprendimas bus realizuotas *MagicDraw* įrankio aplinkoje. Nors abu įrankiai turi panašias savybes ir projektavimo galimybes, visgi modelių transformacijos galimybės ir skirtingo tipo diagramų elementų naudojimas vienoje diagramoje yra svarbūs aspektai sprendimo realizavime. Taip pat *MagicDraw* įrankis yra naudojamas mokymo tikslais Kauno technologijos universitete, dėl to gerai žinomos jo įvairios galimybės ir savybės.

1.6. Darbo tikslas, uždaviniai ir siekiami privalumai

Darbo tikslas: ištirti ir sudaryti *ODRES* metodo realizavimo standartizuotose modeliavimo kalbose galimybes. Tikslui pasiekti keliami tokie uždaviniai:

1. Išanalizuoti informacinių sistemų modeliavimo metodus, jų galimybes, jų modeliavimo įrankius.
2. Išanalizuoti *ODRES* informacinių srautų specifikavimo metodą.
3. Išanalizuoti *ODRES* metodo pritaikymo sprendimus.
4. Surasti pasirinktos modeliavimo kalbos atitikmenis *ODRES* metodui, atliekant analizę metamodelių lygmenyje.
5. Pritaikyti *ODRES* metodo procesą pasirinktam modeliavimo kalbos įrankiui.
6. Sudaryti ir aprašyti *ODRES* metodo pritaikymo pasirinktoje modeliavimo kalbos aplinkoje sprendimą.
7. Parengti informacinių srautų specifikacijos dokumento generavimo šabloną.
8. Atlikti eksperimentinį pasiūlyto sprendimo tyrimą pasirinktai dalykiniai sričiai.

1.7. Siekiamo sprendimo apibrėžimas

Atlikus *ODRES* metodo pritaikymo sprendimų, modeliavimo kalbų ir jų įrankių analizę, buvo nuspręsta *ODRES* metodą realizuoti *UML* kalboje pasirenkant *MagicDraw* modeliavimo įrankį. *ODRES* metodo procesas bus apibrėžiamas *UML* kalboje, kiekvienam proceso žingsniui parenkant reikiamas diagramas ir elementus modeliavimui. Apibrėžus *ODRES* metodo procesą, jis bus pritaikomas pasirinktame modeliavimo įrankyje. Jeigu reikės, bus sukurtas *ODRES* metodo profilis *MagicDraw* įrankyje, kuris praplės esamą *UML* metamodelį papildomais elementais, leisiančiais naudojantis metodo proceso schema specifikuoti informacinės sistemos reikalavimus. Bus parengta *ODRES* metodo reikalavimų specifikacija, projektas, ir pagal tai paruoštas *MagiDraw* įrankis *ODRES* metodo procesui. Taip pat bus paruoštas dokumento generavimo šablonas *MagicDraw* įrankiui, kurio pagalba bus galima generuoti ataskaitą suspecifikuotai sistemai, pagal *ODRES* metodo procesą. Atlikus visus metodo specifikavimo, projektavimo ir realizavimo darbus, bus atliktas eksperimentinis tyrimas, pasirenkant dalykinę sritį ir jai suprojektuojant sistemą pagal ODRES metodo procesą. Atliktas eksperimentas leis nustatyti realizuoto sprendimo teigiamas ir neigiamas sąvybes, padarytas klaidas.

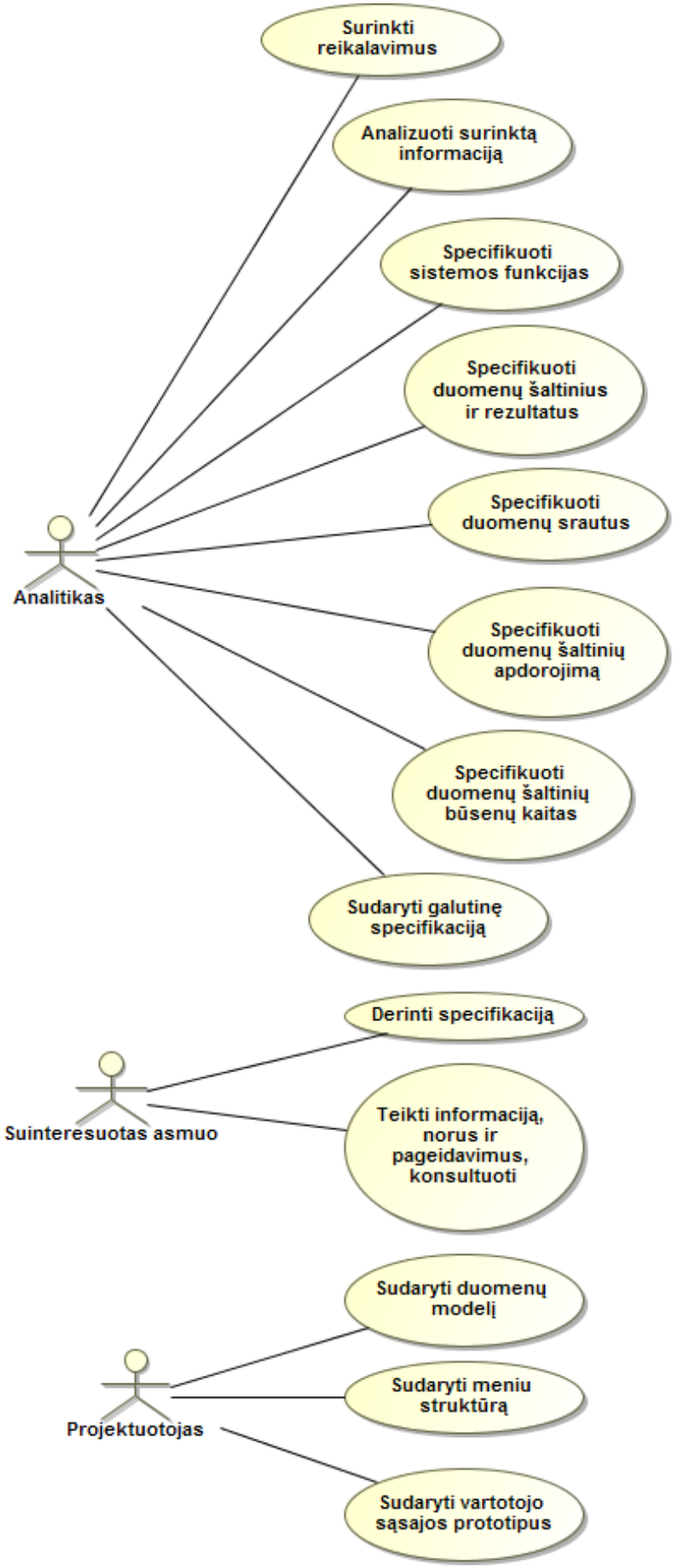
1.8. Analizės išvados

1. Analizės metu buvo susipažinta su tema, surasti ir išanalizuoti literatūros šaltiniai *ODRES* metodo tematikoje.
2. Atlikta *ODRES* metodo analizė, apibrėžtas jo kontekstas: metodo darbinį procesą sudaro šeši žingsniai, kurių metu pilnai apibrėžiami sistemos informaciniai srautai ir sudaroma informacinės sistemos specifikacija.
3. Buvo analizuojami esami sprendimai, kuriuose atsispindi *ODRES* metodas ir jo procesas.
4. Buvo atlikta modeliavimo kalbų ir jos įrankių analizė. *ODRES* metodo pritaikymui pasirinkta *UML* modeliavimo kalba dėl savo populiarumo, lankstumo ir plačių taikymo galimyvių.
5. Taip pat išanalizuoti modeliavimo įrankiai. Kadangi *IDEF* modeliavimo kalba neturėjo tinkamų įrankių, buvo pasirinkti *MagicDraw* ir *Visual Paradigm* modeliavimo įrankiai, kurie palaiko *UML* ir *BPMN* modeliavimo kalbas. *ODRES* metodo realizavimui pasirinktas *MagicDraw* paketas dėl šiek tiek didesnių modeliavimo galimybių, taip pat dėl dažno naudojimo KTU mokymo tikslais.
6. Apibrėžtas siekiamas sprendimas, kuris bus realizuojamas tolimesniuose magistro darbo etapuose. Bus sudaryta *ODRES* metodo reikalavimų specifikacija ir projektas, bus surasti *UML* kalbos ir *ODRES* metodo metamodelių atitikmenys, *ODRES* metodas pritaikytas *MagicDraw* įrankiui, bei paruoštas specifikacijos dokumento generavimo šablonas.

2. ODRES METODO REIKALAVIMŲ SPECIFIKACIJA IR PROJEKTAS, FORMALUS APRAŠAS

2.1. Reikalavimų specifikacija

ODRES metodo panaudojimo atvejai pateikti 2.1 paveiksle. Visi panaudojimo atvejai vaizduoja funkcijas ir veiklas, kurias reikia atlikti norint sudaryti IS specifikaciją remiantis metodo proceso schema. Analitikas surenka reikalavimus bendraudamas su suinteresuotu asmeniu, kuris teikia savo norus ir pageidavimus kuriamai sistemai, jos funkcionalumui, nusako savo sistemos viziją. Analitikas išanalizuoja reikalavimus ir specifikuoja sistemos veikimą bei funkcionalumą remiantis *ODRES* metodo proceso schema. Po kiekvieno *ODRES* metodo proceso žingsnio analitikas derina sistemos specifikaciją su kuriamos sistemos užsakovu, kad šis patvirtintų ar pareikštų papildomus reikalavimus. Atlikęs visus metodo proceso žingnius analitikas turi sudaryti galutinę specifikaciją, kuri pateikiama suinteresuotam asmeniui, kad šis ją patvirtintų. Patvirtinta specifikacija teikiama projektuotojui, kuris suprojektuoja kuriamą IS. Projektuojamai sistemai sudaromas duomenų modelis remiantis duomenų šaltinių ir rezultatų specifikacijomis, kurias sudaro analitikas. Taip pat projektuotojas sudaro kuriamos informacinės sistemos meniu struktūrą, o turint šia struktūrą sudaro vartotojo sąsajos prototipus, vaizduojančius sistemos interfeisą.



2.1 pav. ODRES metodo panaudojimo atvejai

2.2. Dalykinės srities modelis

Atlikus analizę buvo nuspręsta *ODRES* metodą pritaikyti *UML* aplinkoje, todėl tai įgyvendinti reikia surandant *UML* metamodelio ir *ODRES* metodo elementų atitikmenis. Dalykinės srities modelis atitinka *UML* metamodelį, kuris pavaizduotas ir aprašytas pirmame šio darbo skyriuje, kuriam bus pritaikomi *ODRES* metodo elementai.

2.3. Naudotojų sąsajos modelis

Norint rengti informacinės sistemos specifikaciją pagal *ODRES* metodą pritaikytą *UML* aplinkoje reikalingas programinis įrankis kuris:

- Suteikia galimybę modeliuoti *UML* diagramas:
    1. Panaudojimo atvejų;
    2. Klasių;
    3. Veiklos;
    4. Būsenų;
    5. Paketų.
- Leidžia modeliuoti skirtingų diagramų elementus vienoje diagramoje.
- Suteikia galimybe atsekti ryšius tarp elementų ir diagramų.
- Leidžia modeliuoti vartotojo sąsajos prototipo diagramas.

Taip pat programinis įrankis turi suteikti galimybę modeliuoti sekų ir komponentų diagramas, kurios gali būti reikalingos papildomam sistemos specifikacijos atvaizdavimui. Komponentų diagrama taip pat bus reikalinga IS projektavimui.

Programinio įrankio vartotojo sąsaja turi turėti elementų medį, kuriame matomos visos diagramos ir jų elementai, ir iš kurio galima nutempti elementus į modeliuojamą diagramą „drag and drop“ principu. Taip pat diagramų ir jų elementų kūrimo įrankių juostas, bei programinį langą diagramų modeliavimui.

2.4. Formalus sprendimo aprašas

ODRES metodas neturi standartizuotos notacijos, o naudojama savita notacija reikalavimų specifikacijai sudaryti. Pritaikius *ODRES* metodą *UML* aplinkoje, galima lengviau atsekti ryšius tarp specifikuojamų elementų ir modelių, pritaikyti metodą prie šiuolaikinių specifikavimo įrankių ir didinti šio specifikavimo metodo populiarumą.

ODRES metodą sudaro šeši proceso žingniai:

1. Funkcijų hierarchijos specifikavimas;
2. Rezultatų ir duomenų šaltinių struktūros specifikavimas;
3. Duomenų srautų tarp duomenų šaltinių ir jų sudėties specifikavimas;
4. Duomenų šaltinių apdorojimo etapų specifikavimas;
5. Duomenų šaltinių būsenų kaitos specifikavimas;
6. Ryšių tarp duomenų šaltinių detalizavimas ir ryšių tarp duomenų šaltinių būsenų specifikavimas.

Sudarytas *ODRES* metodo procesas, kaip galima būtų specifikuoti informacinę sistemą *ODRES* metodo proceso pagalba. Specifikavimui buvo surasta *ODRES* metodo proceso kiekvienam etapui reikalingos *UML* diagramos arba jų rinkiniai. Visas metodo realizavimas vaizduojamas *MagicDraw* programiniam įrankiui, tačiau tiktų ir kitiems programiniams įrankiams turinčiams panašias sąvybes ir galimybes kaip *MagicDraw*. Atlikus informacinės sistemos specifikavimą, galima pradėti sistemos projektavimą. Projektuojama sistemos duomenų modelis, meniu struktūros hierarchija ir vartotojo sąsajos prototipai. Sudarius sistemos specifikaciją pagal *ODRES* metodo procesą, specifikacijos dokumentas generuojamas pagal iš anksto paruoštą šabloną. Dokumentas gali būti generuojamas ir sistemos projektinei daliai arba specifikacijos ir projektinės dalis apjungus į

vieną bendrą dokumentą. Norint, kad būtų sugeneruotas kuo pilnesnis specifikacijos dokumentas, diagramoms ir tam tikriems diagramų elementams reikia pildyti „documentation“ skiltį, parašant norimą informaciją apie aprašomą elementą arba aprašymą apie specifikuojamą diagramą.

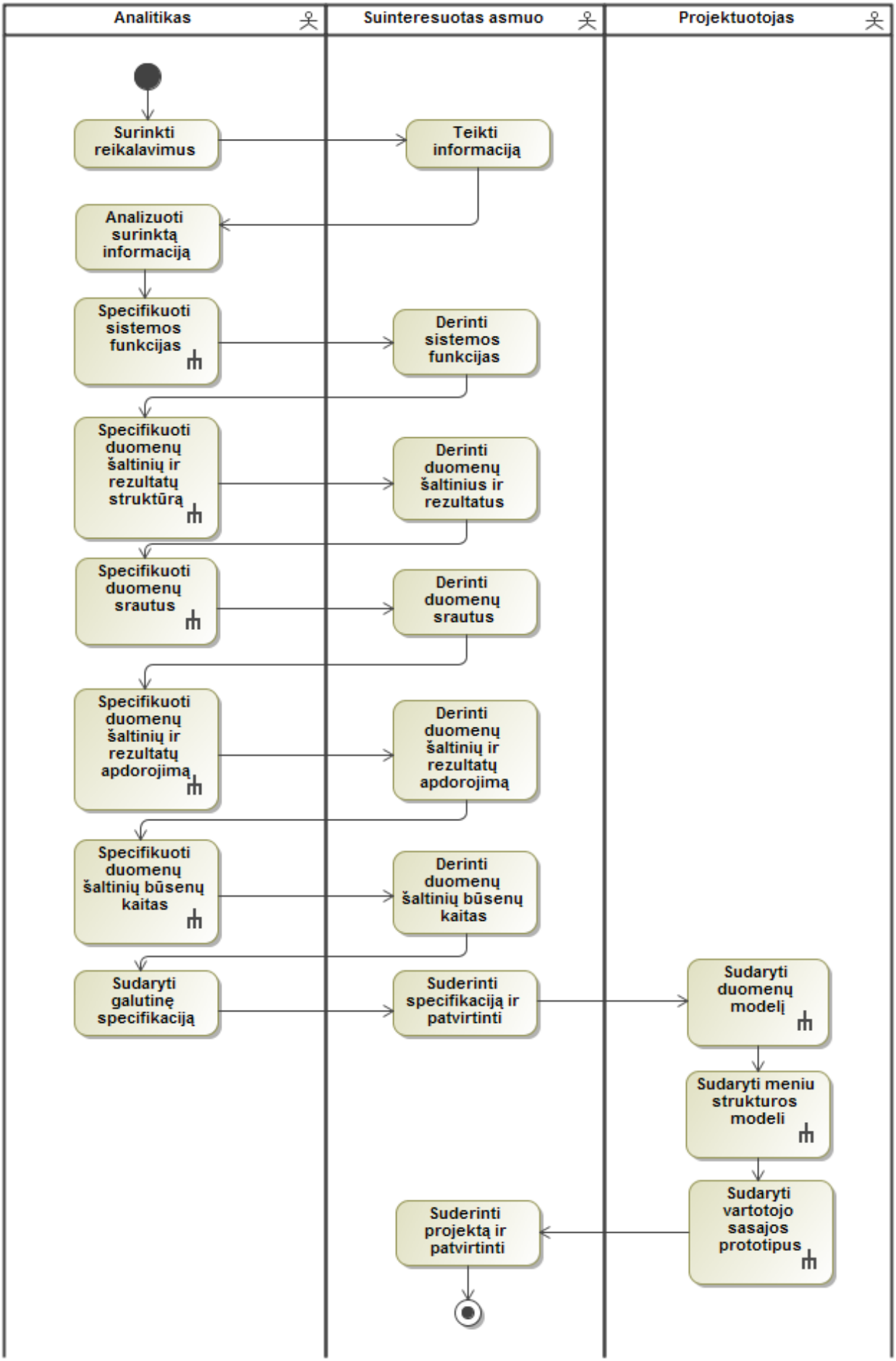
Modeliavimo įrankyje projektą rengti rekomenduojama pagal pateiktą struktūrą, kuri suskirstoma į dvi dalis: sisteos reikalavimų specifikacija ir sistemos projektavimas. Pirmoji turi penkis skirtingus aplankus atspindinčius pirmus penkis *ODRES* metodo proceso žingsnius:

1. Funkcijų hierarchija: aplanke talpinama visa informacija apie funkcijų hierarchiją. Taip pat papildomai gali būti pateikiama informacija apie aktorius vykdančius atitinkamas funkcijas.
2. Rezultatai ir duomenų šaltiniai: aplanke specifikuojami visi duomenų šaltiniai ir rezultatai. Tiek rezultatams, tiek duomenų šaltiniams sukuriami papildomi aplankai, kuriuose kiekviename atskirai kiekvienam rezultatui ar duomenų šaltiniu gali būti kuriamas atskiras aplankas ir jame talpinama visa informacija apie atitinkama duomenų šaltinį ar rezultatą.
3. Duomenų srautai: aplanke specifikuojami duomenų srautai. Kiekvienam duomenų srautui kuriamas atskiras aplankas kuriame talpinama visa infoamacija, diagramos apie tą duomenų srautą.
4. Duomenų šaltinių apdorojimo etapai: aplanke talpinamos veiklos diagramos aprašančios funkcijų hierarchijos funkcijas, kurios realizuoja duomenų šaltinių ir rezultatų apdorojimą.
5. Būsenų kaita: aplanke talpinamos būsenų kaitos specifikacijos.

Antroje modeliavimo įrankio projekto struktūros dalis yra sistemos projektavimas, kuris susideda iš trijų standartinių aplankų:

1. Duomenų modelis: sudaromas sistemos duomenų modelis, kurio pagalba sukuriama duomenų bazė. Duomenų modelis sudaromas remiantis pirmoje dalyje sudarytų duomenų šaltinių ir rezultatų struktūrų diagramomis.
2. Sistemos meniu struktūra: meniu struktūra sudaroma remiantis sudaryta funkcijų hierarchija. Kadangi funkcijos esančios funkcijų hierarchijoje vaizduoja tam tikras sistemos veiklas, jų iškvietimas ir pasiekimas turi būti galimas pasirenkant meniu punktą, kuris realizuos tą veiklą.
3. Vartotojo sąsajos prototipai: talpinami visi vartotojo sąsajos prototipai. Prototipai sudaromi remiantis duomenų šaltinių apdorojimo etapais ir funkcijų hierarchijos struktūra.

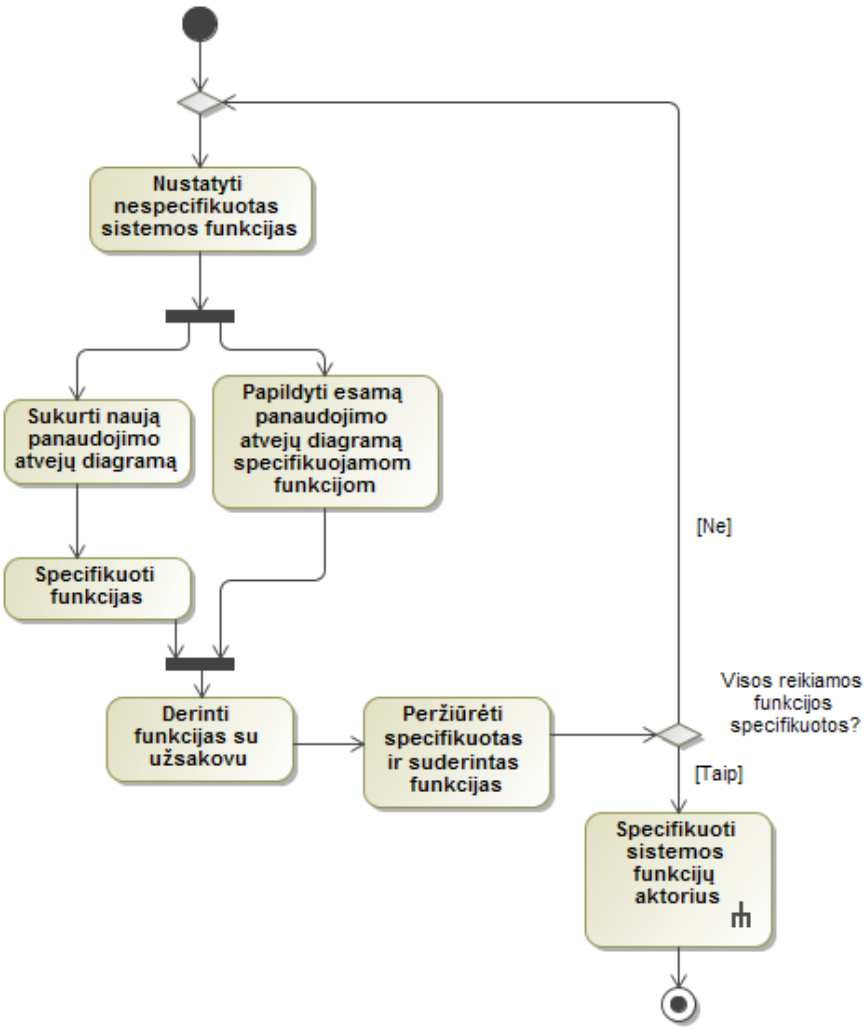
ODRES metodo pagrindinis veiklos procesas pateikiamas 2.2 paveiksle. Analitikas surenka reikalavimus iš suinteresuoto asmens, kuris teikia savo pageidavimus kuriamai IS. Surinkęs informaciją analitikas ją analizuoja ir remiantis ja sudarinėja sistemos specifikaciją pagal *ODRES* metodą. Po kiekvieno *ODRES* metodo proceso žingsnio analitikas derina suspecifikuotą sistemos informaciją su suinteresuotu asmeniu teikiančiu pageidavimus apie sistemą. Sudarius galutinę specifikaciją ji derinama ir tvirtinama suinteresuoto asmens. Nors sistemos specifikavimas pagal *ODRES* metodą vyksta nuosekliai, po kiekvieno specifikavimo etapo reikia peržiūrėti ankstesnių etapų sudarytus modelius, ir patikrinti ar kažkas nebuvo praleista. Taip pat paveiksle matoma, kad bendravimas tarp analitiko ir suinteresuoto asmens turi vykti nuolatos, peržiūrint ir tikslinant esamus ir atsiradusius naujus IS reikalavimus. Patvirtinus sistemos reikalavimus, specifikacija perduodama projektuotojui, kuris remdamasis sudaryta specifikacija projektuoja kuriamą IS. Šiame darbe apimami trys projektavimo aspektai, kurie remiasi *ODRES* metodo proceso pagalba paruošta specifikacija. Pradžioje projektuojamas sistemos duomenų modelis, pagal kurį kuriama informacinės sistemos duomenų bazė. Toliau projektuojama meniu struktūra reikalinga kuriamai informacinei sistemai, kuri kuriama funkcijų hierarchijos pagrindu. Turint meniu struktūrą, o taip pat remiantis duomenų šaltinių apdorojimo etapais, sudaromi vartotojo sąsajos prototipai, kuriuose preliminariai matomas kuriamos sistemos interfeisas.



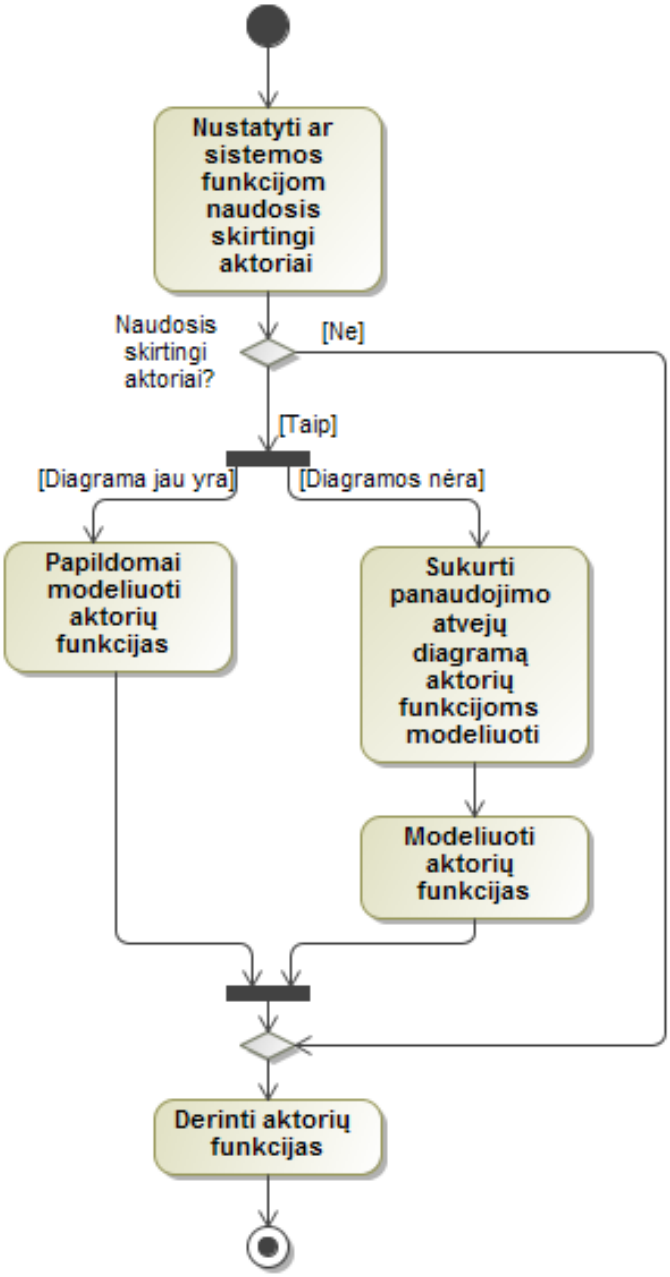
2.2 pav. ODRES metodo proceso veiklos diagrama

Pirmasis specifikavimo etapas pagal *ODRES* metodą „Specifikuoti sistemos funkcijas“ pateikiamas 2.3 paveiksle. Analitikas, žinodamas kokių sistemos funkcijų pageidauja suinteresuotas asmuo, specifikuoja sistemos funkcijas panaudojimo atvejų diagramoje. Funkcijos gali būti specifikuojamos vienoje panaudojimo atvejų diagramoje, arba jeigu funkcijų yra daug, žemesnio lygio panaudojimo atvejai gali būti specifikuojami atskirose panaudojimo atvejų diagramose. Kiekvienas pavaizduotas panaudojimo atvejis atitinka tam tikrą sistemos funkciją. Aukštesnio lygio panaudojimo atvejai tik apibendrina žemesnio lygio panaudojimo atvejus. Prie aukštesnio lygio panaudojimo atvejų žemesnio lygio panaudojimo atvejai jungiami „include“ ryšiu. Žemiausiame lygyje turėtų būti ne atominės sistemos funkcijos, o taip pat apibendrinančios, kurios realizuojamos elementariomis operacijomis. Žemiausio lygio funkcijų detalumas specifikuojamas ketvirtame *ODRES* metodo proceso etape.

Funkcijų specifikavimas prasideda analitikui nustatant ir išsiaiškinant kokios bendrai turi būti sistemos funkcijos. Tada analitikas sukuria panaudojimo atvejų diagramą sistemos funkcijoms specifikuoti. Suspecifikavęs funkcijas analitikas turi jas suderinti su užsakovu. Suderinus reikia dar kartą peržvelgti ar tikrai visos funkcijos įtrauktos į specifikaciją. Jeigu yra trūkstamų funkcijų, analitikas jas papildomai specifikuoja. Jeigu nėra analitikas turi specifikuoti sistemos aktorių funkcijas. Sistemos aktorių funkcijų specifikavimo diagrama pateikiama 2.4 paveiksle. Pradžioje analitikas nustato ar sistema naudosis daugiau negu vieno tipo aktorius. Jeigu sistema naudosis keli skirtingi aktoriai, kuriama panaudojimų atvejų diagrama jiems specifikuoti. Diagramoje vaizduojami aktoriai ir prie jų jungiami funkcijų hierarchijos žemiausio lygio panaudojimo atvejai, priklausantys atitinkamam aktoriui. Jeigu aktoriui priklauso visos funkcijos, kurias apibendrina aukštesnio lygio funkcija, tada jungiama aukštesnio lygio funkcija. Specifikuotos aktorių funkcijos taip pat turi būti derinamos su užsakovu.



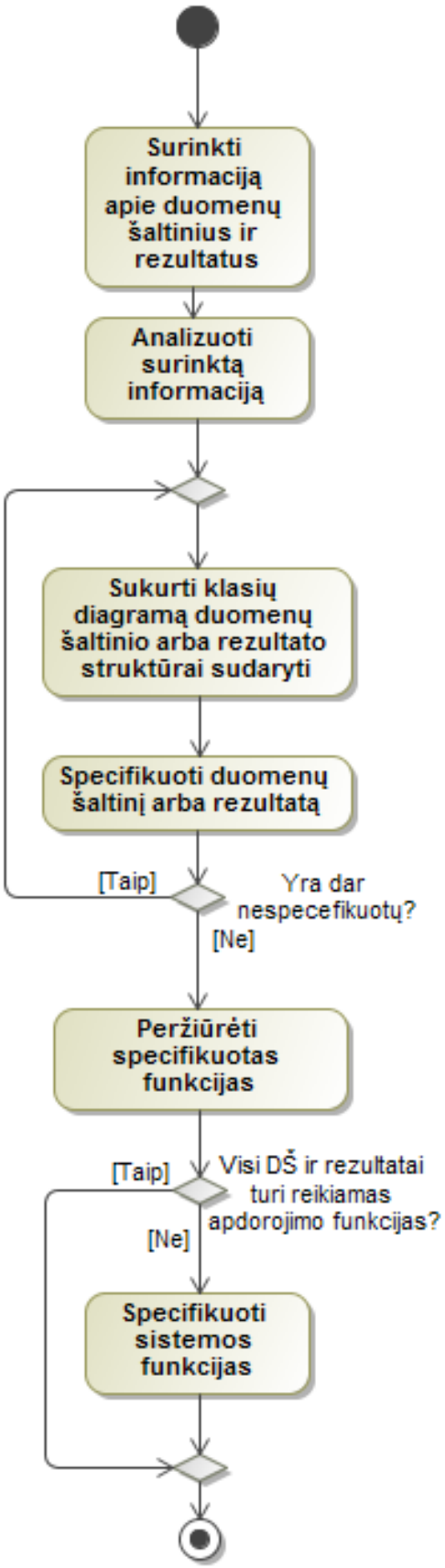
2.3 pav. Sistemos funkcijų specifikavimo veiklos diagrama



2.4 pav. Sistemos aktorių funkcijų specifikavimo veiklos diagrama

Antrasis specifikavimo etapas pagal *ODRES* metodą „Duomenų šaltinių ir rezultatų specifikavimas“ pateikiamas 2.5 paveiksle. Analitikas turi surinkti duomenų šaltinių ir rezultatų pavyzdžius, informaciją apie juos iš suinteresuoto asmens ir išanalizavęs specifikuoja duomenų šaltinių ir rezultatų struktūras klasių diagramomis. Rezultatais arba duomenų šaltiniais gali būti įvairūs realaus pasaulio objektai: spausdinti dokumentai, žodiniai pranešimai, laiškai, ataskaitos, liktinės informacinės sistemos, jų duomenų bazės ir t.t. Kiekvienas rezultatas ir duomenų šaltinis specifikuojamas atskiroje klasių diagramoje, kurioje vaizduojamos visos duomenų šaltinio arba rezultato esybės, bei jų atributai.

Pradžioje analitikas surenka visus duomenų šaltinius ir rezultatus. Surinkęs juos analizuoja ir tada kiekvienam duomenų šaltiniu arba rezultatui atskirai sukuria klasių diagramas, kol nebelieka nespecifikuotų duomenų šaltinių arba rezultatų. Atlikęs antrojo etapo specifikavimo veiksmus, analitikas turi peržiūrėti specifikuotas sistemos funkcijas, nes duomenų šaltiniai ir rezultatai turi turėti juos apdorojančias funkcijas. Jeigu trūksta funkcijų, funkcijų hierarchija papildoma reikiamomis funkcijomis.



2.5 pav. Duomenų šaltinių ir rezultatų struktūros specifikavimo veiklos diagrama

Trečiajame etape vykdomas duomenų srautų specifikavimas. Duomenų srautams specifikuoti surasti du variantai. 2.6 paveiksle vaizduojamas pirmojo varianto veiklos procesas, 2.7 paveiksle vaizduojamas antrojo varianto veiklos procesas. Antrasis variantas skiriasi tik diagramų

tipais ir kiekiu, kurias reikia modeliuoti. Duomenų srautai specifikuojami kiekvienam rezultatui atskirai.

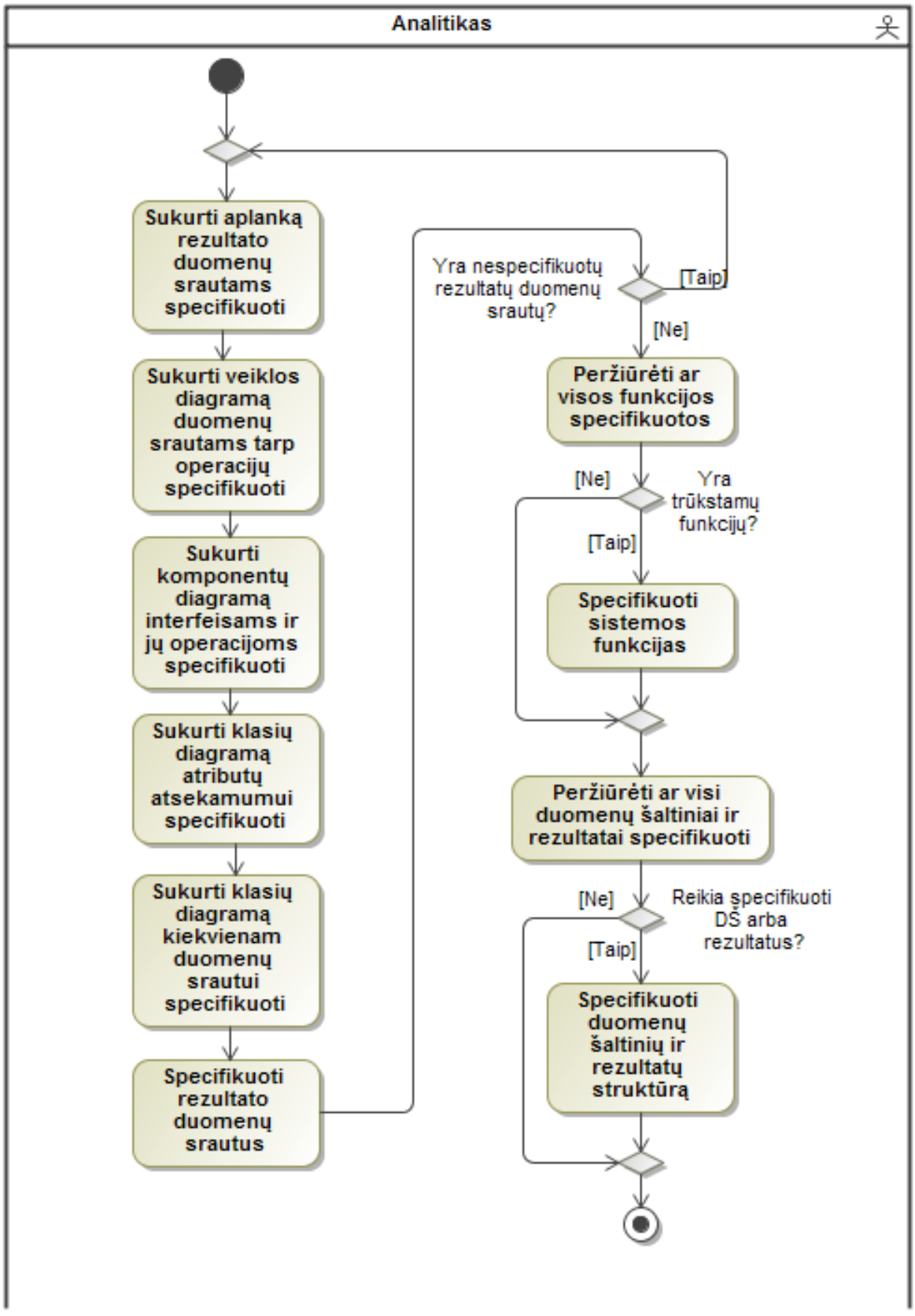
Pirmojo varianto metu kuriamos kelios pagrindinės diagramos ir taip pat papildomų diagramų. Diagramos:

- Veiklos - duomenų srautams ir operacijoms tarp kurių tie duomenų srautai keliauja atvaizduoti. Veiklos atitinka interfeiso operacijas, o tarp veiklų vaizduojami duomenų objektai, kurie atvaizduojami klasių diagramomis ir realizuoja duomenų srautus.
- Komponentų - skirta atvaizduoti komponentui, kuris realizuojamas interfeisais, kurių pagalba atliekamos operacijos, šios operacijos yra veiklos diagramos veiklos. Taip pat esant didesniam interfeisų specifikavimo detalumo lygiui, kiekvienam interfeisui galima kurti klasių diagramą, kurioje vaizduojamas interfeisas ir jo naudojamos duomenų esybės.
- Kuriamos klasių diagramos skirtos kiekvieno duomenų srauto specifikacijai atvaizduoti. Diagrama vaizduoja visą duomenų srautą, o klasės vaizduoja duomenų srauto esybes. Taip pat specifikuojami ir duomenų srauto esybių atributai.
- Taip pat kuriama klasių diagrama skirta rezultato esybių ir duomenų šaltinių esybių atributų atsekamumui vaizduoti. Šia diagrama vaizduojama kurie duomenų šaltinio esybės atributai formuoja atitinkamus rezultato esybės atributus. Tarp atributų atsekamumui vaizduoti nautojamas „*trace*“ ryšys.

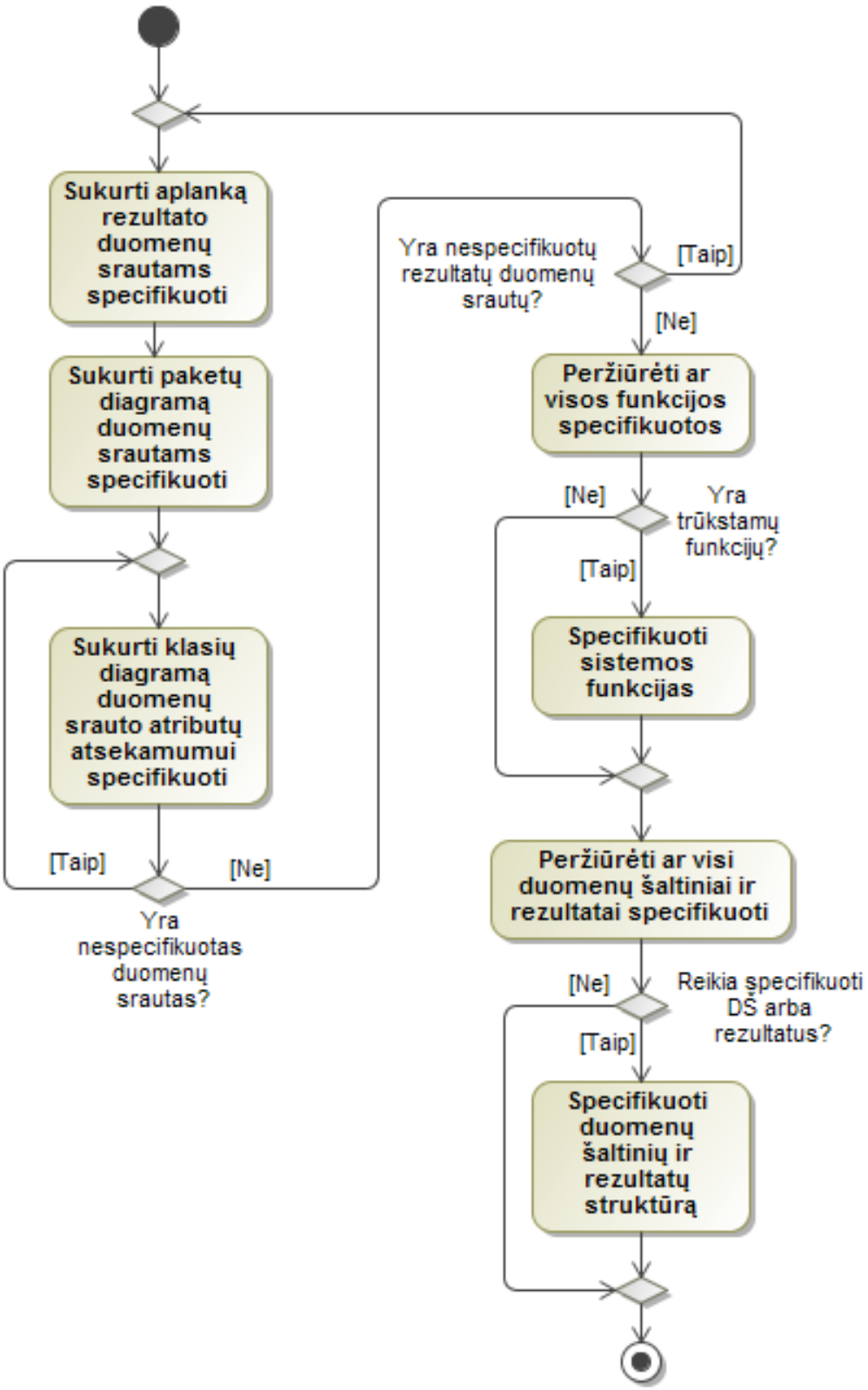
Antrojo varianto metu kuriamos kelios diagramos, paketų ir klasių:

- Pirmoje vaizduojami paketai, kurie atspindi duomenų šaltinį arba rezultatą. Paketai jungiami ryšiu rodančiu kryptį, taip aiškiai matoma duomenų srauto kryptis. Ryšiui uždedama nuoroda į klasių diagramą, kurioje vaizduojami tų rezultato ir duomenų šaltinio atributų ryšiai.
- Antroji diagrama yra klasių diagrama, kuri vaizduojama pirmoje klasių diagramoje ant jungiančio paketų ryšio. Ši klasių diagrama skirta rezultato esybių ir duomenų šaltinių esybių atributų atsekamumui vaizduoti. Šia diagrama vaizduojama kurie duomenų šaltinio esybės atributai formuoja atitinkamus rezultato esybės atributus. Tarp atributų atsekamumui vaizduoti nautojamas „*trace*“ ryšys.

Analitikas specifikuodamas duomenų srautus sukuria visas reikiamas diagramas ir jas pildo lygiagrečiai, kol sumodeliuojamas rezultato duomenų srautai. Kai baigiami specifikuoti visų rezultatų duomenų srautai, analitikas vėl turi peržvelgti ankstesnių specifikavimo etapų diagramas ir įsitikinti, kad nereikia jų papildyti. Specifikuojant sistemos duomenų srautus reikia naudotis tik vienu duomenų srautų specifikavimo variantu. Rekomenduojama naudotis antruoju variantu, kadangi šiuo būdu sukuriama mažiau diagramų, diagramos kuriamos greičiau, taupomas analitiko darbo laikas, o tuo pačiu ir sistemos kūrimo kaštai. Visgi jeigu reikalingas labai detalus duomenų srautų specifikavimas, kai sistemos kūrime gali daug kainuoti klaidos tikimybė, galima naudoti pirmąjį duomenų srautų specifikavimo variantą, kurio pagalba galima smulkiai detalizuoti duomenų srautus nurodant kiekvieno duomenų srauto operacijas, struktūras, naudojamas esybes, tarpusavio ryšius.



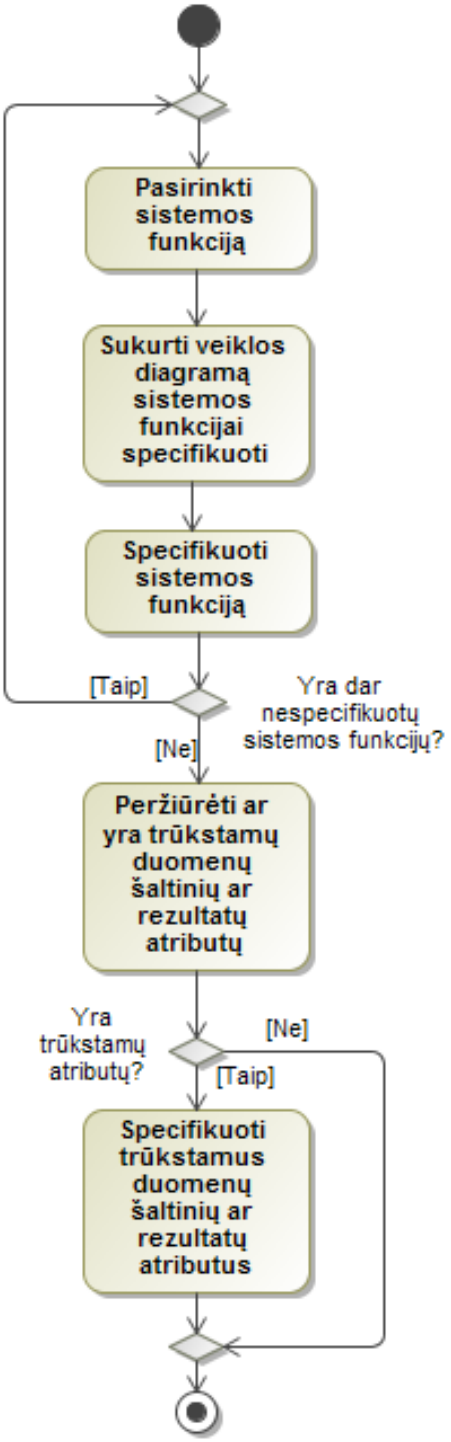
2.6 pav. Duomenų srautų specifikavimo pirmojo varianto veiklos diagrama



2.7 pav. Duomenų srautų specifikavimo antrojo varianto veiklos diagrama

Ketvirtasis *ODRES* metodo proceso etapas yra „Duomenų šaltinių ir rezultatų apdorojimo etapai“. Duomenų šaltiniai ir rezultatai apdorojami funkcijomis suspecifikuotomis pirmojo etapo metu, todėl norint specifikuoti duomenų šaltinių apdorojimo etapus reikia specifikuoti kiekvieną funkcijų hierarchijos žemiausio lygio funkciją, jai sudarant veiklos diagramą. Specifikuodami funkcijas apimame visų duomenų šaltinių įvairias apdorojimo ir tuo pačiu rezultatų gavimo ar išvedimo veiklas.

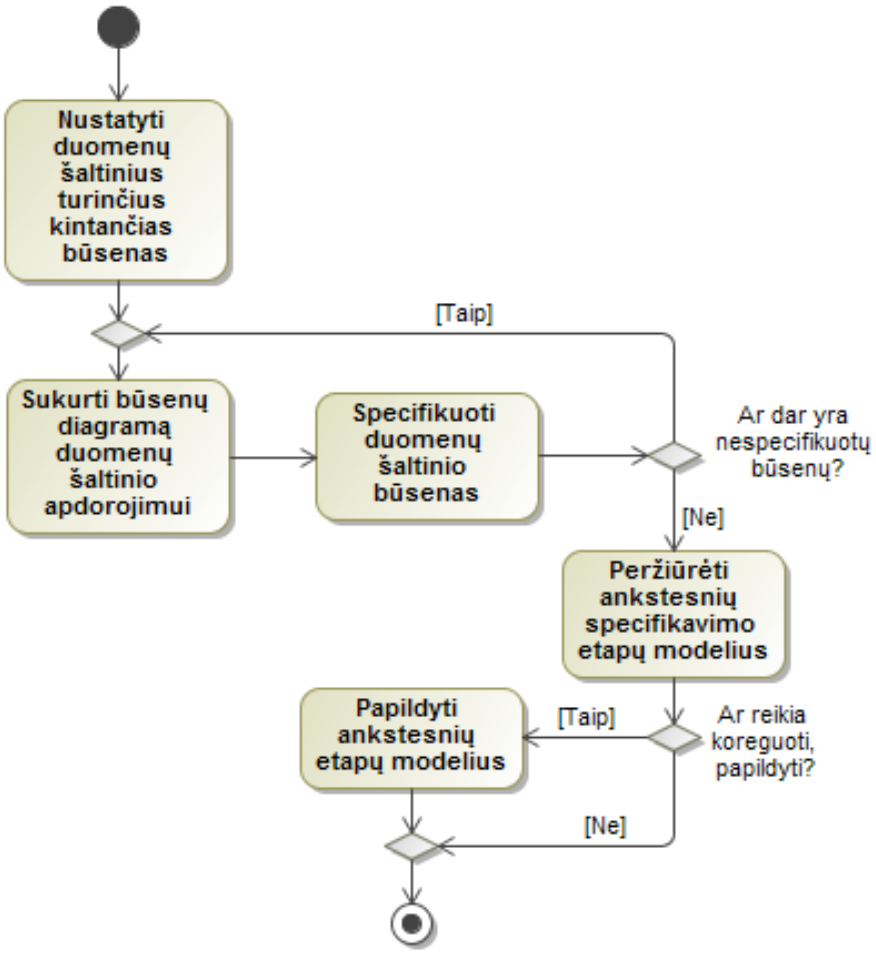
Duomenų šaltinių ir rezultatų apdorojimo etapų specifikavimo veiklos diagrama pateikiama 2.8 paveiksle. Analitikas pasirenka sistemos funkciją, sukuria jai veiklos diagramą ir specifikuoja funkciją modeliuodamas veiklos diagramą. Atlikus visų funkcijų specifikavimą, analitikas turi peržvelgti duomenų modelį ir patikrinti ar nereikia įtraukti papildomų atributų ar duomenų esybių, nustatytų specifikuojant funkcijas.



2.8 pav. Duomenų šaltinių apdorojimo etapų specifikavimo veiklos diagrama

Penktasis *ODRES* metodo proceso etapas yra busenų kaitos specifikavimas. Šio etapo metu svarbu specifikuoti sistemos objektų, duomenų šaltinių, rezultatų būsenas, kurios įtakoja duomenų pasikeitimus kiekvienos būsenos metu. Būdenų kaitos specifikavimo veiklos diagrama pateikiama 2.9 paveiksle. Pradžioje analitikas turi nustatyti duomenų šaltinius, rezultatus arba sistemos objektus, kurių būsenų pasikeitimai atlieką svarbų vaidmenį sistemos veikime. Kiekvienam tokiam objektu

analitikas sukuria būsenų diagramą, kurioje specifikuoja būsenų kaitas. Baigus būsenų specifikavimą reikia peržiūrėti ankstesnių etapų diagramas ir esant reikalui jas papildyti, patikslinti.



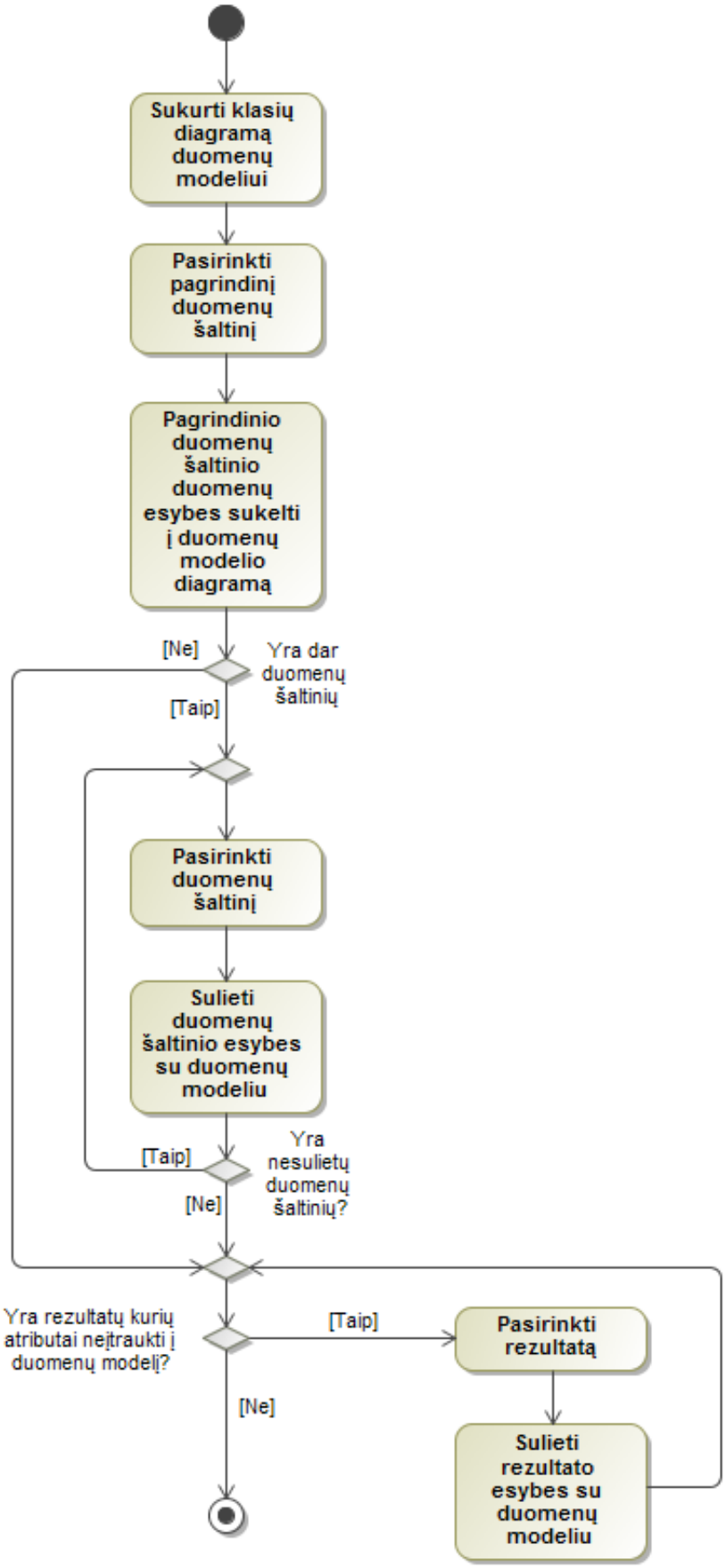
2.9 pav. Būsenų kaitos specifikavimo veiklos diagrama

Šeštajam etapui, ryšių tarp duomenų šaltinių detalizavimas ir ryšių tarp duomenų šaltinių būsenų specifikavimas atitikmenų nerasta, taigi rengiant specifikaciją pagal *ODRES* metodo procesą *UML* aplinkoje šio etapo atsisakyta.

Atlikus informacinės sistemos reikalavimų specifikavimo darbus ir suderinus specifikaciją su sistemos užsakovais, pereinama prie sistemos projektavimo etapo. Projektuojant sistemą galima modeliuoti įvairius sistemus aspektus reikalingus kuriamai sistemai, tačiau šio darbo metu nuspręsta apsiriboti duomenų modelio, meniu struktūros ir vartotojo sąsajos prototipų sudarymu, kadangi specifikacija sudaryta pagal *ODRES* metodą gali ženkliai palengvinti šių sričių projektavimą.

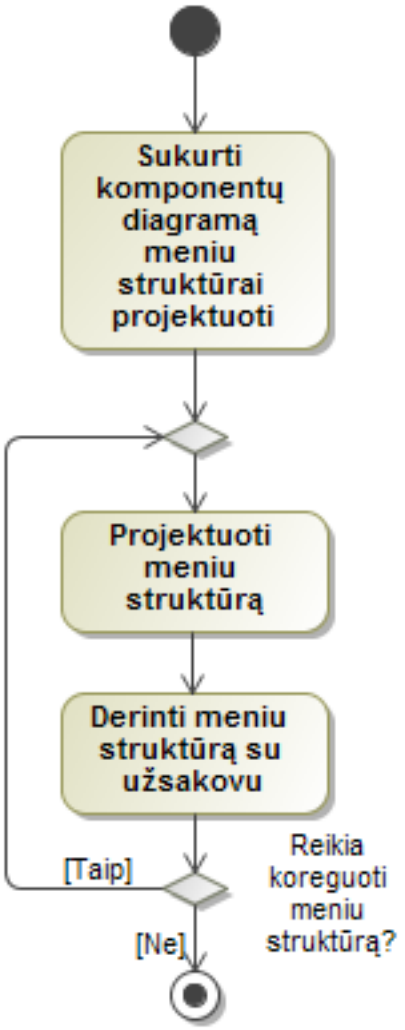
Duomenų modelis projektuojamas sudarant klasių diagramą. Duomenų modelio projektavimo veiklos diagrama pateikiama 2.10 paveiksle. Projektuotojas sukuria klasių diagramą duomenų modeliui. Tada pasirenka pagrindinį duomenų šaltinį arba rezultatą, kurio pagrindu bus modeliuojamas duomenų modelis. Duomenų šaltinio arba rezultato esybės perkeliamos į duomenų modelį su visais ryšiais ir atributais. Tada paeiliui imami visi duomenų šaltiniai ir ju esybės suliejamos į pagrindinį duomenų modelį. Jeigu duomenų šaltinio esybė sutampa su jau esančia duomenų modelio esybe, tai į duomenų modelį įtraukiami tik nesutampantys duomenų šaltinio esybės atributai. Tačiau jeigu duomenų modelyje tokios esybės nėra kaip duomenų šaltinyje, visa duomenų šaltinio esybė įtraukiama į duomenų modelį. Peržiūrėjus visus duomenų šaltinius reikia peržiūrėti ar visų rezultatų esybių atributai turėjo juos formuojančius duomenų šaltinių esybių atributus. Jeigu neturėjo reikia rezultato esybės atributus ar netgi visą esybę taip pat įtraukti į duomenų modelį. Projektuojant sistemą ir naudojantis modeliavimo įrankiu, tokiu kaip *MagicDraw*, kuris turi transformavimo į duomenų bazės struktūrą galimybes, sudaryta duomenų modelio klasių

diagrama gali būti transformuojama į duomenų bazės schemą. Tada pagal gautą duomenų bazės schemą galima generuoti programinį kodą, kurio pagalba sukuriama fizinė duomenų bazė su visom schemoje esančiom lentelėm ir ryšiais tarp jų.



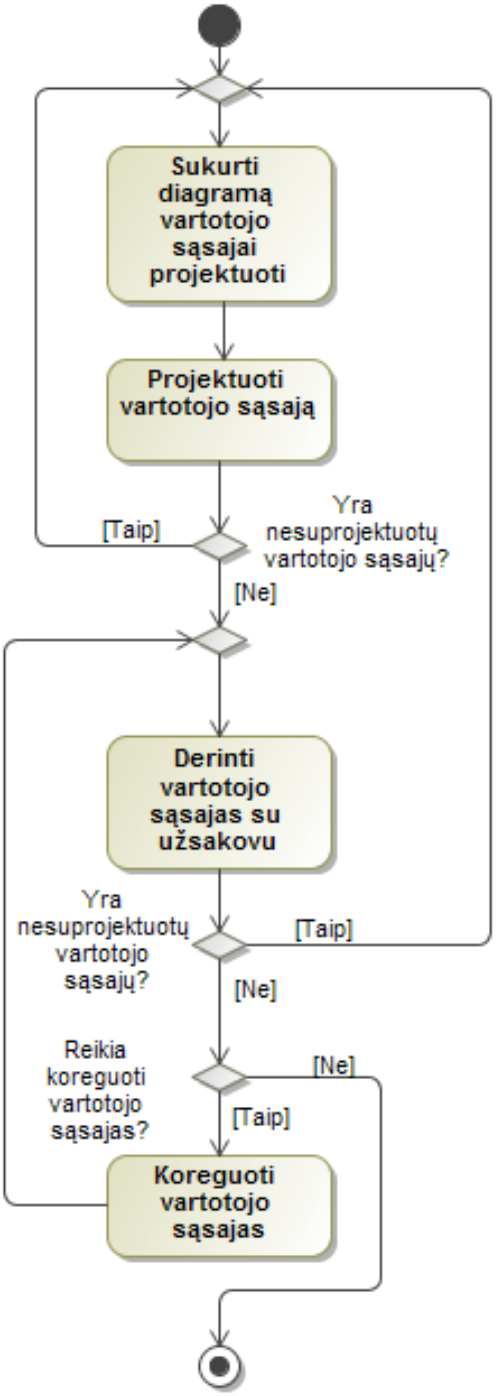
2.10 pav. Duomenų modelio projektavimo veiklos diagrama

Sudarius duomenų modelį projektuotojas gali sudaryti kuriamos informacinės sistemos meniu struktūrą. Meniu struktūra sudaroma sukuriant komponentų diagramą. Ši struktūra sudaroma remiantis funkcijų hierarchija. Kiekvienas meniu punktas vaizduojamas interfeiso komponentu, o jam priklausantys meniu punktai jungiami priklausomumo ryšiu. Taip gaunama struktūra, kuri informacinės sistemos programinėje įrangoje atvaizduos meniu struktūrą. Antro lygio meniu struktūros elementai bus pagrindiniai meniu punktai sistemoje, o prie jų prijungti meniu struktūros elementai bus atitinkamai išskleidžiami, prijungiami. Informacinės sistemos meniu struktūros sudarymo veiklos diagrama pateikima 2.11 paveiksle.



2.11 pav. Sistemos meniu struktūros sudarymo veiklos diagrama

Kuriamos informacinės sistemos vartotojo sąsajos prototipų sudarymo veiklos diagrama pateikiama 2.12 paveiksle. Vartotojo sąsajos prototipai sudaromi remiantis *ODRES* metodo pagalba sudarytomis duomenų šaltinių apdorojimo etapų specifikacijomis, o taip pat sudaryta meniu struktūra ir jos elementais, kur žemiausio lygio meniu struktūros elementams reikalinga vartotojo sąsaja norint atlikti reikiamas funkcijas. Projektuotojas kiekvienai reikiamai vartotojo sąsajai sukuria diagramą, ir joje modeliuoja prototipą. Sumodeliavus derina viską su suinteresuotu asmeniu, kuris išsako savo pastebėjimus ar pageidavimus vartotojo sąsajai. Esant reikalui prototipai koreguojami, tobulinami, o jeigu viskas yra tinkama, patvirtinama.



2.12 pav. Sistemos vartotojo sąsajos prototipų sudarymo veiklos diagrama

2.5. Reikalavimų apibendrinimas

Norint sudaryti pilną informacinės sistemos specifikaciją pagal *ODRES* metodą reikia sudaryti funkcijų hierarchijos, duomenų šaltinių ir rezultatų struktūros, jų apdorojimo etapų, duomenų srautų, būsenų specifikacijas, galiausiai viską apjungiant į vieną bendrą dokumentą, kurį patvirtina suinteresuotas asmuo. Specifikacijai rengti reikalingas programinis įrankis palaikantis *UML* notaciją, kuriuo galima modeliuoti klasių, būsenų, veiklos, komponentų, paketų ir panaudojimo atvejų diagramas. Sudarius specifikaciją galima rengti sistemos projektą remiantis sudaryta specifikacija pagal *ODRES* metodą. Pradiniai projektuotojo uždaviniai sudaryti duomenų modelį, sistemos meniu struktūrą ir vartotojo sąsajos prototipus. Tam reikalingos klasių ir komponentų diagramos, o vartotojo sąsajos prototipai gali būti rengiami įvairiuose įrankiuose leidžiančiuose modeliuoti vartotojo sąsajos eskizus. Tolimesnių projektavimo darbų šio darbo metu neįtraukiame.

3. ODRES METODO EKSPERIMENTINĖS REALIZACIJOS PROJEKTAS

ODRES metodo procesą sudaro šeši žingsniai, tačiau nuspręsta realizuoti penkis, atsisakant paskutiniojo. Kiekvienam *ODRES* proceso žingniui realizuoti reikalingos atitinkamos diagramos ir jų elementai, kas šiuo metu neturi standartizuotos notacijos. Taigi kiekvienam metodo proceso etapui surasti *UML* diagramų ir elementų atitikmenys.

Pirmasis *ODRES* metodo proceso žingnis yra funkcijų hierarchijos specifikavimas. Funkcijų hierarchija specifikuojama panaudojimo atvejų diagrama. Yra du funkcijų hierarchijos specifikavimo etapai, pradžioje specifikuojama informacinės sistemos funkcijų hierarchija, o ją turint specifikuojamos funkcijų priklausomybės atitinkamiems aktoriams. Funkcijų specifikavimo veiklos procesai apibrėžti ir pavaizduoti antrame skyriuje 2.3 ir 2.4 paveiksluose. Pirmasis funkcijų hierarchijos specifikavimo etapas:

- Specifikuojama funkcijų hierarchija, tam sukuriama panaudojimo atvejų diagrama.
- Panaudojimo atvejis atspindi sistemos funkciją.
- Norint atskirti panaudojimo atvejį nuo funkcijos papildomai sukuriamas stereotipas „Funkcija“, kuris priskiriamas panaudojimo atvejui. Stereotipas sukuriamas, kad praplėstų *UML* metamodelio panaudojimo atvejo metaklasę.
- Sistemos funkcijos vaizduojamos lygiais nuo aukščiausio iki žemiausio. Aukštesnio lygio funkcija apibendrina jai priklausančias žemesnio lygio funkcijas arba atvirkščiai, žemesnio lygio funkcijos detalizuoja aukštesnio lygio funkcijų galimybes.
- Funkcijos arba panaudojimo atvejai jungiami „include“ ryšiais. Prie aukštesnio lygio funkcijos, ją detalizuojančios žemesnio lygio funkcijos prijungiamos „include“ ryšiu rodant kryptį nuo aukštesnio lygio funkcijos į žemesnio. Šis ryšys *UML* metamodelyje reiškia, kad panaudojimo atvejis realizuojamas šiuo ryšiu prie jo prijungtais panaudojimo atvejais.

Antrasis funkcijų hierarchijos specifikavimo etapas skirtas funkcijų priklausomybės aktoriams specifikavimas:

- Funkcijų priklausomybės aktoriams specifikuojamos panaudojimo atvejų diagrama, tam sukuriama antra panaudojimo atvejų diagrama.
- Panaudojimo atvejų diagramoje sukuriami visi aktoriai.
- Kiekvienam aktoriui asociacijos ryšiais jungiamos funkcijos iš prieš tai sudarytos funkcijų hierarchijos. Prie aktorių jungiamos žemiausio lygio funkcijų hierarchijos funkcijos, tačiau jeigu aktorius vykdo visas žemesnio lygio funkcijas, kurias apibendrina aukštesnio lygio funkcija, tada prie aktoriaus jungiama aukštesnio lygio funkcija.
- Jeigu aktoriai sudaro aktorių grupę ir jie visi vykdo tam tikras bendras funkcijas, sukuriamas aktorių grupės aktorius, prie jo jungiamos bendros funkcijos, o aktoriai jungiami prie aktorių grupės elemento generalizacijos ryšiu, taip nurodant, kad aktorius paveldi (vykdo) visas aktorių grupės funkcijas. Visos ne bendrai vykdomos funkcijos jungiamos tiesiogiai prie jas vykdančio aktoriaus.
- Jeigu vieno aktoriaus visas funkcijas vykdo antras aktorius, tai fukcijos jungiamos prie pirmojo, o antrasis aktorius jungiamas prie pirmojo generalizacijos ryšiu parodant paveldimumą. Visos papildomos antrojo aktoriaus vykdomos funkcijos jungiamos tiesiogiai prie čio aktoriaus asociacijos ryšiu.

Visi *UML* elementai naudojami *ODRES* metodo proceso pirmajame etape specifikuojant funkcijų hierarchiją pateikiami 3.1 lentelėje.

3.1 lentelė. UML elementai funkcijų hierarchijai specifikuoti

ODRES metodo elementas	UML elementas
Funkcijų hierarchija	Panaudojimo atvejų diagrama
Funkcijų priklausomybė aktoriams	Panaudojimo atvejų diagrama
Funkcija	Panaudojimo atvejis
Ryšys tarp funkcijų	„Include“ ryšys
Aktorius	Aktorius
Ryšys tarp aktoriaus ir funkcijos	Asociacijos ryšys
Paveldimumo ryšys tarp aktorių	Generalizacijos ryšys

Antrasis *ODRES* metodo proceso žingsnis yra duomenų šaltinių ir rezultatų struktūros specifikavimas. Kiekvienas duomenų šaltinis ar rezultatas specifikuojamas atskira klasių diagrama. Duomenų šaltinių ir rezultatų veiklos procesas aprašytas antrame skyriuje ir pavaizduotas 2.5 paveiksle. Duomenų šaltinio arba rezultato specifikavimas:

- Sukuriamas atskiras aplankas duomenų šaltinių arba rezultatų aplanke atitinkamai skirtas specifikuojamam duomenų šaltiniui arba rezultatui.
- Aplanke sukuriama klasių diagrama struktūros specifikavimui.
- Kiekviena duomenų šaltinio arba rezultato esybė vaizduojama klasės elementu.
- Kiekviena duomenų šaltinio arba rezultato esybė gali turėti atributus, detalizuojančius jų informaciją, tai detalizuojama klasės atributais.
- Norint detalizuoti klases nurodant, kad tai duomenų šaltinio arba rezultato esybė sukuriami papildomi stereotipai „Duomenų šaltinio esybė“ ir „Rezultato esybė“. Stereotipui priskiriama *UML* metamodelio klasės metaklasė.
- Duomenų šaltinio arba rezultato esybės tarpusavyje jungiamos asociacijos ryšiais nurodant kardinalumą vienas kito atžvilgiu.

Visi *UML* elementai naudojami *ODRES* metodo proceso antrame etape specifikuojant duomenų šaltinių ir rezultatų struktūrą pateikiami 3.2 lentelėje.

3.2 lentelė. UML elementai duomenų šaltiniams ir rezultatams specifikuoti

ODRES metodo elementas	UML elementas
Duomenų šaltinis	Klasių diagrama
Rezultatas	Klasių diagrama
Duomenų šaltinio esybė	Klasė
Duomenų šaltinio esybės atributas	Klasės atributas
Rezultato esybė	Klasė
Rezultato esybės atributas	Klasės atributas
Ryšys tarp esybių	Asociacijos su tarpusavio kardinalumais ryšys

Trečiasis *ODRES* metodo proceso žingsnis yra duomenų srautų specifikavimas. Duomenų srautams specifikuoti surasti du variantai. Jų veiklos procesai aprašyti antrame skyriuje ir pavaizduoti atitinkamai 2.6 ir 2.7 paveiksluose. Kuris variantas yra tinkamesnis bus nuspręsta atlikus eksperimentinę pasirinktos dalykinės srities informacinės sistemos specifikavimą pagal *ODRES* metodą. Pirmasis variantas duomenų srautų specifikavimui:

- Pradžioje pasirenkamas rezultatas, kuriam duomenų srautai bus vaizduojami.
- Pasirinkus rezultatą pirma sukuriama komponentų diagrama kurioje vaizduojamas programinės įrangos komponentas realizuojantis pasirinkto rezultato gavimą. Toliau vaizduojami interfeisai, kurie realizuoja pasirinkto rezultato komponentą.
- Kiekviename interfeise detalizuojamos operacijos, kurių pagalba gaunamas pasirinktas rezultatas. Operacijoms detalizuojami įėjimo ir išejimo parametrai, vaizduojantys duomenis, kurių reikia sėkmingam operacijos įvykdymui ir duomenų srauto perdavimui.

- Antroji kuriama veiklos diagrama. Joje vaizduojamas procesas kokia tvarka vykdomos interfeisų operacijos norint gauti pasirinktą rezultatą, o tarp operacijų veiklų vaizduojami duomenų objektai, kuriuose nurodomos nuorodos į duomenų srautus tarp operacijų. Veiklos sukuriamos joms nurodant atitinkamą operacijos atributą.
- Kiekvienam duomenų objektui, kuris atvaizduoja duomenų srautą, sukuriama klasių diagrama. Kiekviena duomenų srauto esybė vaizduojama klase, jai nurodant atitinkamai klasės atributus, kurie vaizduoja duomenų srauto esybės atributus.
- Duomenų srauto esybės tarpusavyje jungiamos asociacijos ryšiais nurodant kardinalumus tarpusavio priklausomybės.
- Norint detalizuoti klases vaizduojančias duomenų srauto esybes, papildomai sukuriamas stereotipas „Duomenų srauto esybė“. Stereotipui priskiriama *UML* metamodelio klasės metaklasė.

Visi *UML* elementai naudojami *ODRES* metodo proceso trečiame etape specifikuojant duomenų srautus pirmuoju variantu pateikiami 3.3 lentelėje. Lentelėje nepaminėti papildomi elementai ir diagramos siekiant detalizuoti duomenų srautus, juos apdorojančias operacijas sistemoje ir papildomai specifikuojamą informaciją apie kuriamą sistemą.

3.3 lentelė. UML elementai duomenų srautams specifikuoti pagal pirmą variantą

ODRES metodo elementas	UML elementas
Duomenų srautas	Klasių diagrama
Duomenų srauto esybė	Klasė
Duomenų srauto esybės atributas	Klasės atributas
Rezultatas	Nevaizduojama
Duomenų šaltinis	Nevaizduojama

Antrasis variantas duomenų srautų specifikavimui:

- Pradžioje pasirenkamas rezultatas, kuriam duomenų srautai bus specifikuojami.
- Sukuriama paketų diagrama, kurioje bus vaizduojami pasirinkto rezultato duomenų srautai.
- Rezultatas ir kiekvienas jį formuojantis duomenų šaltinis vaizduojamas paketu, kuriame sukuriama nuoroda į to rezultato ar duomenų šaltinio klasių diagramą.
- Paketai jungiami „dependency“ ryšiu nurodant duomenų srauto kryptį. Ryšiui priskiriamas „Duomenų srautas“ stereotipas kuris sukuriamas papildomai. Stereotipui priskiriama *UML* metamodelio „dependency“ ryšio metaklasė.
- Pats duomenų srautas specifikuojamas sukuriama klasių diagrama, kurioje sukeliamos reikiamos duomenų šaltinio ir rezultato arba duomenų šaltinio ir duomenų šaltinio esybės tarp kurių detalizuojamas duomenų srautas. Ryšiui tarp paketų taip pat sukuriama nuoroda į duomenų srautą specifikuojančią diagramą.
- Klasių diagramoje specifikuojami duomenų srautai atributų lygmenyje. „*Trace*“ ryšiu jungiami duomenų šaltinio esybės atributas iš kurio imami duomenys ir duomenų šaltinio arba rezultato esybės atributas į kurį patenka duomenys. Visi formuojamo duomenų šaltinio ar rezultato esybių atributai turi turėti į juos patenkančius „trace“ ryšius nurodančius iš kur gaunami duomenys.

Visi *UML* elementai naudojami *ODRES* metodo proceso trečiame etape specifikuojant duomenų srautus antruoju variantu pateikiami 3.4 lentelėje.

3.4 lentelė. UML elementai duomenų srautams specifikuoti pagal antrą variantą

ODRES metodo elementas	UML elementas
Duomenų srautas	„Dependency“ ryšiu detalizuojant klasių diagrama atributų lygmenyje.
Duomenų srautas atributų lygmenyje	„Trace“ ryšys

Duomenų srauto esybė	Nevaizduojama
Duomenų srauto esybės atributas	Nevaizduojama
Rezultatas	Paketas
Duomenų šaltinis	Paketas

Ketvirtas *ODRES* metodo proceso žingsnis yra duomenų šaltinių apdorojimo etapų specifikavimas. Duomenų šaltinių apdorojimo etapų veiklos procesas aprašytas antrame skyriuje ir pavaizduotas 2.8 paveiksle. Duomenų šaltinių apdorojimas specifikuojamas veiklos diagramomis. Pirmajame etape sudarytai funkcijų hierarchijai, kiekvienai žemiausio lygio funkcijai kuriama veiklos diagrama. Duomenų šaltinių apdorojimo etapų specifikavimas:

- Kiekvienai žemiausio lygio funkcijų hierarchijos funkcijai sukuriama veiklos diagrama.
- Diagramoje detalizuojamas funkcijos veiklos procesas.
- Kiekviena veikla realizuoja duomenų šaltinio arba rezultato apdorojimo etapą.

Visi *UML* elementai naudojami *ODRES* metodo proceso ketvirtame etape specifikuojant duomenų šaltinių apdorojimo etapus pateikiami 3.5 lentelėje.

3.5 lentelė. UML elementai duomenų šaltinių apdorojimo etapams specifikuoti

ODRES metodo elementas	UML elementas
Duomenų šaltinio apdorojimas	Veiklos diagrama
Apdorojimo etapas	Veikla
Perėjimas tarp etapų	„Control flow“ ryšys

Penktasis *ODRES* metodo proceso žingsnis yra būsenų kaitos specifikavimas. Būsenų kaita specifikuojama sukuriant būsenų diagramą. Specifikuojamos tik tos būsenų kaitos, kurios įtakoja kažkokius duomenų pasikeitimus specifikuojamoje sistemoje. Būsenų kaitos specifikavimo veiklos procesas aprašytas antrame skyriuje ir pavaizduotas 2.9 paveiksle. Būsenų kaitos specifikavimas:

- Pradžioje nustatoma ar yra specifikuojamos sistemos objektas ar procesas, kuriam reikia specifikuoti būsenų kaitas. Specifikavimui sukuriama būsenų diagrama.
- Diagramoje vaizduojamos būsenos ir jų eiliškumas. Kiekvieną būseną atitinka *UML* būsenos elementas

Visi *UML* elementai naudojami *ODRES* metodo proceso penktame etape specifikuojant būsenų kaitas pateikiami 3.6 lentelėje.

3.6 lentelė. UML elementai duomenų šaltinių apdorojimo etapams specifikuoti

ODRES metodo elementas	UML elementas
Būsenų kaita	Būsenų diagrama
Būsena	Būsena
Perėjimas tarp būsenų	„Transition“ ryšys

Pirmų trijų *ODRES* metodo proceso žingsnių metu, specifikuojant sistemą *UML* aplinkoje reikalingi papildomi stereotipai, kurių pagalba lengviau detalizuoti diagramas ir jos tampa lengviau skaitomos ir suprantamos. *ODRES* metodo realizavimui *UML* aplinkoje reikalingi papildomi stereotipai pateikiami 3.7 lentelėje.

3.7 lentelė. Nauji UML stereotipai

Stereotipas	Elementas	Aprašymas
Funkcija	Panaudojimo atvejis	Kiekvienas panaudojimo atvejis reiškia tam tikro lygio funkciją sistemoje, taigi panaudojimo atvejams pridedamas funkcijos stereotipas.
Duomenų šaltinio esybė	Klasė	Specifikuojant duomenų šaltinius kiekvienam kuriama klasių diagrama. Kiekviena diagramos klasė realizuoja tam tikrą duomenų šaltinio esybę.

Rezultato esybė	Klasė	Specifikuojant rezultatus kiekvienam kuriama klasių diagrama. Kiekviena diagramos klasė realizuoja tam tikrą rezultato esybę.
Duomenų srauto esybė	Klasė	Specifikuojant duomenų srautus pirmuoju variantu, kuriamos klasių diagramos kurios vaizduoja duomenų srautus, tai kiekviena klasė realizuoja duomenų srauto esybę.
Duomenų srautas	Priklausomybės ryšys	Specifikuojant duomenų srautus antruoju variantu, kuriamos paketų diagramos, kuriose vaizduojami duomenų srautai, kiekvienas priklausomybės ryšys realizuoja tam tikrą duomenų srautą.
Duomenų šaltinis	Paketas	Specifikuojant duomenų srautus paketų diagramoje nurodomas duomenų šaltinio stereotipas, jeigu paketas vaizduoja duomenų šaltinį.
Rezultatas	Paketas	Specifikuojant duomenų srautus paketų diagramoje nurodomas paketo rezultato stereotipas, jeigu paketas vaizduoja rezultatą.

Atlikus sistemos specifikavimo etapus pagal *ODRES* metodą galima pradėti sistemos projektavimo darbus. Šiame darbe nuspręsta apsiriboti trimis projektavimo aspektais: duomenų modelio sudarymu, meniu struktūros sudarymu ir vartotojo sąsajos prototipų sudarymu. Duomenų modelis gali būti sudaromas bet kokiu metu kai pilnai specifikuoti visi duomenų šaltiniai ir rezultatai, labiausiai rekomenduojama kai baigta visos sistemos specifikacija. Tokiu būdu į duomenų modelį bus įtrauktos visos reikalingos duomenų esybės ir jų atributai. Duomenų modelis turi būti sudaromas vienoje klasių diagramoje kopijuojant į ją reikiamas duomenų šaltinių ir rezultatų esybes ir esant reikalui jos gali būti apjungiamos arba papildomos naujais kitų esybių atributais, jo sudarymo veiklos procesas aprašytas antrame skyriuje 2.10 paveiksle. Duomenų modelio projektavimas:

- Pradžioje sukuriama klasių diagrama duomenų modelio sudarymui.
- Pasirenkamas pagrindinis rezultatas arba duomenų šaltinis, kuris turi daugiausiai esybių įtraukiamų į duomenų modelį. Visos reikiamos esybės kopijuojamos į duomenų modelį.
- Pasirenkami vienas po kito paeiliui rezultatai ir duomenų šaltiniai. Jų esybės įtraukiamos į duomenų modelį, jeigu neturi jau įtrauktų esybių atitikmenų. Jeigu turi atitikmenų tikrinami atributai ir trūkstami įtraukiami į duomenų modelio esybės atributus.
- Sudaryta duomenų modelio diagrama gali būti transformuojama į *SQL* (angl. Structured Query Language) klasių diagramą, iš kurios gali būti generuojamas programinis kodas duomenų bazės sukūrimui.

Meniu strukūra sudaroma remiantis sistemos funkcijų hierarchija. Meniu struktūros hierarchija atspindi funkcijų hierarchijos struktūrą. Meniu struktūros sudarymas pavaizduotas 2.11 paveiksle ir aprašytas antrame skyriuje. Meniu struktūros projektavimas:

- Sukuriama komponentų diagrama meniu struktūros projektavimui.
- Meniu struktūra projektuojama remiantis sudaryta funkcijų hierarchija, kur kiekviena funkcija atspindi meniu punktą. Meniu elementas realizuojamas interfeiso komponentu.
- Atliekant sistemos programavimą meniu struktūros antro lygio elementai pavirsta pagrindiniais meniu punktais, kurie gali būti išskleidžiami ir kiekvienami atitinkamai vaizduojami detalesni jam priklausantys meniu punktai iš trečio lygio meniu struktūros ir t.t.

Vartotojo sąsajos prototipai sudaromi remiantis duomenų šaltinio apdorojimo etapų specifikacijomis ir sudaryta meniu struktūra. Vartotojo sąsajos prototipų sudarymo veiklos procesas pateikiamas ir aprašytas antrame skyriuje 2.12 paveiksle. Vartotojo sąsajos prototipų projektavimas:

- Sudarant vartotojo sąsają kiekvienam prototipui sukuriama interfeiso projektavimo diagrama.
- Kiekvienas duomenų įvedimo laukas, mygtukas ar kitoks vartotojo sąsajos elementas turi realizuoti tam tikrą veiklą duomenų šaltinio apdorojimo etapų specifikacijose, jai reikalingus duomenis iš duomenų modelio, ar turėti panašų ryšį.
- Kiekvienas žemiausio lygio meniu struktūros elementas turi turėti vieną ar daugiau jį realizuojantį vartotojo sąsajos prototipą.

Projektavimo etapo elementai, reikalingi projektuojant sistemą remiantis sistemos specifikacija sudaryta pagal *ODRES* metodą, ir jų aprašymai pateikiami 3.8 lentelėje.

3.8 lentelė. Sistemos projektavimo dalies elementai

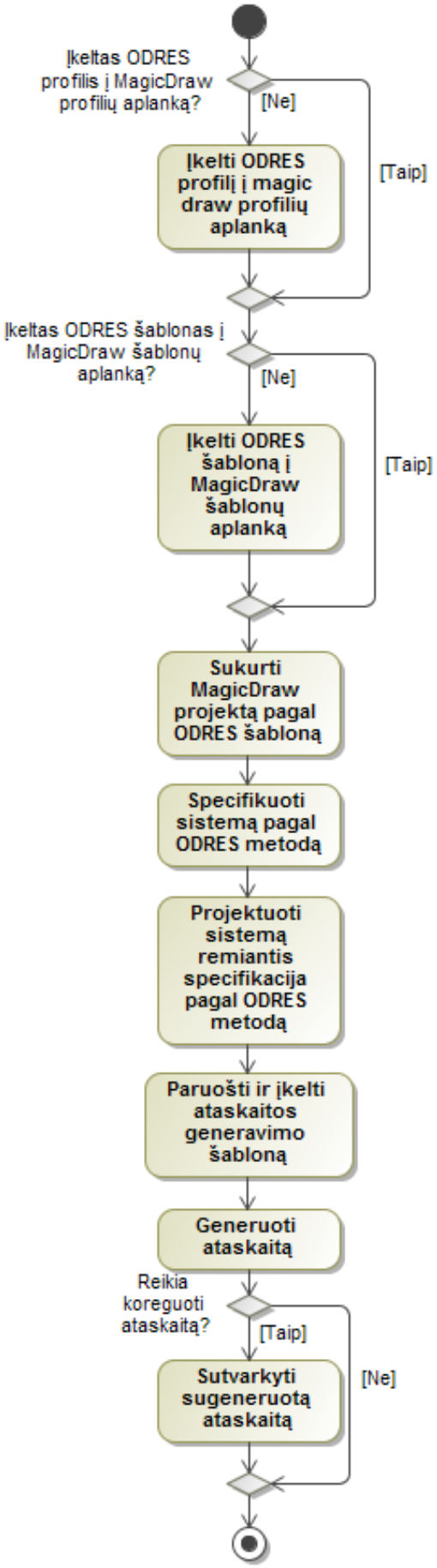
Elementas	UML elementas	Aprašymas
Duomenų modelis	Klasių diagrama	Duomenų modelis sudaromas klasių diagramoje, kuri transformuojama į SQL klasių diagramą. Tada generuojamas programinis kodas duomenų bazės sudarymui.
Duomenų modelio esybė	Klasė	Kiekviena duomenų modelio esybė vaizduojama klase, jai nurodant atributus ir atributų tipus.
Meniu struktūra	Komponentų diagrama	Meniu struktūra sudaroma komponentų diagrama.
Meniu struktūros elementas	Interfeiso komponentas	Kiekvienas meniu elementas vaizduojamas interfeiso komponentu .
Ryšys tarp meniu struktūros elementų	„Dependency“ priklausomumo ryšys	Ryšys detalizuoja kurie žemesnio lygio meniu elementai priklauso aukštesnio lygio meniu elementui
Vartotojo sąsajos prototipas	Vartotojo sąsajos modeliavimo diagrama	Vartotojo sąsajos prototipas arba eskizas gali būti modeliuojamas projektuotojo pasirinktu modeliavimo įrankiu, leidžiančiu modeliuoti vartotojo sąsają.
Vartotojo sąsajos prototipo elementas	Įvairūs vartotojo sąsajos modeliavimo elementai	Vartotojo sąsaja gali būti sudaryta iš įvairių standartinių vartotojo sąsajos elementų.

4. ODRES SPRENDIMO REALIZACIJA

4.1. Sprendimo realizacijos ir veikimo aprašas

ODRES metodas ir jo procesas realizuojamas *MagicDraw* įrankyje, kurio pagalba sistemos analitikas gali paruošti sistemos specifikaciją pagal *ODRES* metodą. *ODRES* metodo realizacijai buvo sukurtas *MagicDraw* profilis, kuriame yra stereotipai reikalingi sudarant sistemos specifikaciją pagal *ODRES* metodą. Taip pat paruoštas *MagicDraw* šablonas, kurio pagalba kuriant projektą įrankyje automatiškai sukuriama failų struktūra ir pridedamas *ODRES* profilis. Buvo sudarytas pradinis ataskaitos generavimo šablonas, kurio pagalba sugeneruojama ataskaita iš MagicDraw įrankio paimant reikalingas diagramas ir jas sudedant pagal vartotojo nurodytą tvarką.

Informacinės sistemos specifikavimo pagal *ODRES* metodą *MagicDraw* įrankyje veiklos procesas pateikiamas 4.1 paveiksle. Pradžioje reikia įkelti *MagicDraw* įrankio *ODRES* profilį į profilių aplanką, kad būtų galima profilį pridėti rankiniu būdu arba jis bus pridedamas automatiškai kuriant projektą pagal *ODRES* šabloną. Profilių aplankas pavadinimu „Profiles“ yra *MagicDraw* įrankio instaliaciniame aplanke. Į aplanką reikia įkelti *ODRES* profilio projektą, taip jis automatiškai atsiras *MagicDraw* profilių sąraše. Įkėlus odres profilį, toliau reikia įkelti *ODRES* projekto šabloną į *MagicDraw* įrankio šablonų aplanką, kuris yra *MagicDraw* instaliaciniame aplanke pavadinimu „Templates“. Įkėlus šablonas automatiškai atsiras *MagicDraw* šablonų sąraše. Turint *ODRES* profilį ir šabloną įkeltus į reikiamas vietas, galima kurti *MagicDraw* projektą informacinės sistemos specifikavimui. Projektas kuriamas pasirenkant kūrimą iš šablono, nurodant *ODRES* šabloną. Sukūrus projektą automatiškai sukuriami pradiniai reikiami aplankai sistemos specifikavimui pagal *ODRES* metodą, projektavimui remiantis sudaryta specifikaija ir automatiškai prie projekto pridedamas *ODRES* profilis. Toliau specifikuojama informacinė sistema pagal *ODRES* metodą, specifikavimo algoritmas aprašytas antrame ir trečiame skyriuose. Po specifikavimo atliekami informacinės sistemos projektavimo darbai taipogi aprašyti šio darbo antrame ir trečiame skyriuose. Atlikus sistemos specifikaciją ir projektavimą paruošiamas ataskaitos generavimo šablonas, kuriame atitinkamose vietose įrašoma, kokias diagramas analitikas nori matyti pasirinktinai. Paruoštas generavimo šablonas įkeliamas į *MagicDraw* įrankį. Toliau generuojama ataskaita pagal šabloną į analitiko nurodytą vietą kompiuteryje. Sugeneruotą ataskaitą dar reikia peržiūrėti ir pakoreguoti, jeigu randami kažkokie netikslumai.

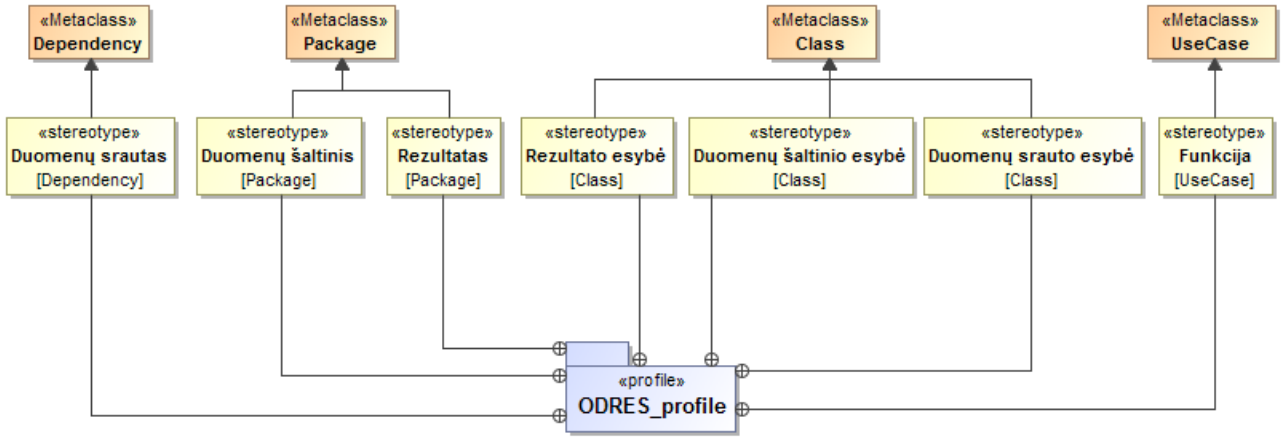


4.1 pav. Modeliavimo MagicDraw įrankyje veiklos procesas

Darbo metu sudarytas *ODRES* metodo profilis *MagicDraw* modeliavimo įrankiui. Profilyje sukurti penki stereotipai:

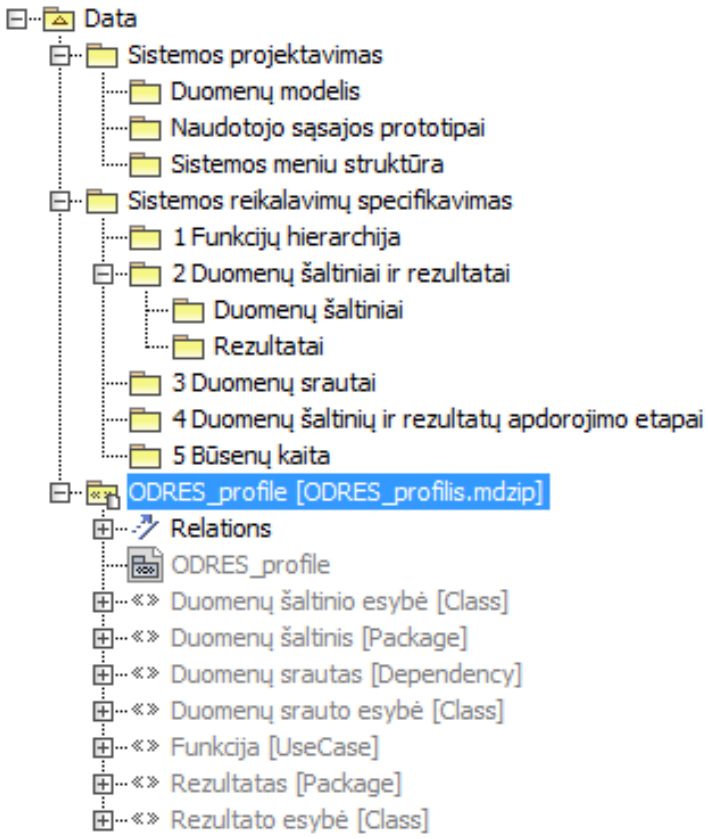
- Funkcija (panaudojimo atvejams);
- Duomenų srautas (priklausomybės ryšiams);
- Rezultato esybė (klasėms);
- Duomenų šaltinio esybė (klasėms);
- Duomenų srauto esybė (klasėms);
- Duomenų šaltinis (paketams);
- Rezultatas (paketams).

Visi stereotipai ir jų priklausomybės *UML* metamodelio klasėms pateikiama 4.2 paveiksle.



4.2 pav. ODRES profilio diagrama

Darbo metu buvo sukurtas *ODRES* šablonas skirtas *MagicDraw* įrankiui. Specifikuojant informacinę sistemą pagal *ODRES* metodą naudojantis *MagicDraw* įrankiu projektas sukuriamas pagal sudarytą *ODRES* šabloną. Kuriant naują projektą pasirenkamas projektas pagal šabloną ir nurodomas *ODRES* šablonas. Sukūrus projektą automatiškai sukuriama projekto pradinė aplankų struktūra skirta modeliavimui. Sukuriami du pagrindiniai aplankai skirti reikalavimų specifikavimui ir projektavimui, kurių keikvienas turi savo struktūrą. Sistemos reikalavimų specifikavimo vidiniai aplankai atspindi pirmuosius penkis *ODRES* metodo proceso žingsnius, nes šeštojo buvo nuspręsta atsisakyti. Sistemos projektavimo aplanke sukuriami papildomi aplankai duomenų modelio, meniu struktūros ir vartotojo sąsajos prototipų projektavimui. Taip pat automatiškai prie projekto pridedamas ir *ODRES* profilis, tačiau jeigu naudotojas neįkėlė profilio į *MagicDraw* profilių aplanką, pateikiamas pranešimas, kad profilio nėra ir jį teks pridėti rankiniu būdu. *MagicDraw* įrankyje sukuriama aplankų struktūra pateikiama 4.3 paveiksle.



4.3 pav. ODRES šablono aplankų struktūra

Buvo sudarytas ataskaitos generavimo šablonas skirtas *MagicDraw* modeliavimo įrankiui, kurio pagalba naudotojas gali greičiau paruošti suspecifikuotos informacinės sistemos ataskaitą. Ataskaitos generavimo šablonas pateikiamas šio darbo prieduose. Generavimo šablonas sudarytas trim skirtingiems atvejamams, kai naudotojas nori generuoti tik reikalavimų specifikacijos ataskaitą, tik projektavimo dalies ataskaitą arba abiejų dalių bendrą ataskaitą. Sudarius kiekvienos informacinės sistemos specifikaciją, jos pagrindu atlikus projektavimą ir norint pasinaudoti ataskaitos generavimo šablonu, jį reikės papildyti ir pakoreguoti rankiniu būdu pagal naudotojo poreikius. *MagicDraw* įrankio visi elementai turi dokumentacijos atributą, kuriame gali būti patalpinta įvairi informacija apie elementą ar kažkokia pagalbinė informacija, kurią tik sugalvos naudotojas. Generuojant ataskaitą tų elementų dokumentacija bus atspausdinta pagal paruoštą formą. Tai žymiai pagreitina ir palengvina naudotojo darbą, nes jis modeliuodamas gali kartu ir dokumentuoti diagramas ir jos elementus. Taip darbas vyksta greičiau lyginant su darbu kai naudotojas modeliuoja įrankyje, o ruošiant ataskaitą visas diagramas nuosekliai dėlioja ir jas bei jų elementus dokumentuoja. Elementai, kuriuos galima dokumentuoti ir jų dokumentacija bus atspausdinta pagal paruoštą formą:

- Visos *UML* diagramos;
- Klasė;
- Panaudojimo atvejis.

Buvo paruošti trys ataskaitos generavimo šablonai. Reikalavimų specifikacijos ataskaitos generavimo šablonas *MagicDraw* modeliavimo įrankiui paruoštas pagal struktūrą atitinkančią *ODRES* metodo proceso žingnius atskiriant duomenų šaltinius nuo rezultatų. Tačiau analitikas gali lengvai pakoreguoti šabloną pagal savo poreikius. Šablono skyrių struktūra:

1. Funkcijų hierarchija;
2. Rezultatai;
3. Duomenų šaltiniai;
4. Duomenų srautai;
5. Duomenų šaltinių apdorojimo etapai;

6. Būsenų kaita;

Sistemos projektavimo ataskaitos generavimo šablonas *MagicDraw* modeliavimo įrankiui turi tris skyrius, tačiau projektuotojas juos gali koreguoti pagal savo poreikius:

1. Duomenų modelis;
2. Sistemos meniu struktūra;
3. Naudotojo sąsajos prototipai.

Trečiasis ataskaitos generavimo šablonas sudarytas apjungiant abejas dalis, sistemos reikalavimų specifikaciją ir sistemos projektavimą:

1. Sistemos reikalavimų specifikacija;
   1.1. Funkcijų hierarchija;
   1.2. Rezultatai;
   1.3. Duomenų šaltiniai;
   1.4. Duomenų srautai;
   1.5. Duomenų šaltinių apdorojimo etapai
   1.6. Būsenų kaita;
2. Sistems projektavimas;
   2.1. Duomenų modelis;
   2.2. Sistemos meniu struktūra;
   2.3. Naudotojo sąsajos prototipai.

Generavimo šablone paruoštas funkcionalumas spausdinti trijų tipų diagramas. Diagramas kurias analitikas nori atspausdinti kiekviename skyriuje pasirenka pats papildydamas generavimo šabloną įrašydamas nuorodas į norimas diagramas:

- Klasių diagrama - atspausdinamas klasių diagramos paveikslas, jo dokumentacija ir toliau paeiliui atspausdinama lentelė su kiekviena klase ir jos dokumentacija.
- Panaudojimo atvejų diagrama - atspausdinamas panaudojimo atvejų diagramos paveikslas, žemiau pateikiama diagramos dokumentacija, o toliau paeiliui atspausdinamos kiekvieno panaudojimo atvejo lentelės su panaudojimo atvejo dokumentacija.
- Diagrama nepriklausomai nuo tipo - atspausdinamas diagramos paveikslas ir po jo atspausdinamas diagramos dokumentacija. Taip galima atspausdinti ir klasių bei panaudojimo atvejų diagramas, jeigu nereikalingas papildomas tų diagramų spausdinimo funkcionalumas.

5. EKSPERIMENTINIS ODRES METODO TYRIMAS

5.1. Eksperimento planas

Eksperimentinis *ODRES* metodo proceso tyrimas vykdomas remiantis pasirinktos informacinės sistemos specifikavimu pagal *ODRES* metodą ir jo procesą bei projektavimu, aprašytais šio darbo antrame ir trečiame skyriuose, naudojant *MagicDraw* modeliavimo įrankį. Eksperimentine realizacijai pasirinktas valstybinei miškų tarnybai vykdytas projektas „Miškų kadastro integruotos informacinės sistemos plėtra“ (MKIIS). Projektą vykdė Kauno technologijos universiteto darbuotojai. MKIIS informacija aktuali daugeliui Lietuvos Respublikos valstybinių institucijų, fiziniams ir juridiniams asmenims sprendžiant miško ūkinės veiklos planavimo, vykdymo uždavinius. Todėl MKIIS kaupiama informacija turi būti tiksli ir prieinama tikslinėms naudotojų grupėms. MKIIS skirta Lietuvos Respublikos teritorijoje esančių miškų sklypinės inventorizacijos metu gautu duomenų saugojimui, apdorojimui ir teikimui kaip numato Miškų kadastro nuostatos. MKIIS plėtros projektas buvo pakankamai platus, todėl nuspręsta pagal *ODRES* metodo procesą specifikuoti ir projektuoti tik miškų kadastro duomenų teikimo posistemio (MKDTP) plėtros projekto dalį. MKDTP skirtas pažymų užsakymui ir duomenų, apie kadastrinius sklypus ir juose esančius miškus, peržiūrai. Sudaryta MKIIS posistemio specifikacija pagal *ODRES* metodo procesą, sudaryti modeliai pagal tokius punktus: funkcijų hierarchija, duomenų šaltiniai ir rezultatai, duomenų srautai, duomenų šaltinių apdorojimo etapai, būsenų kaita. Pagal sudarytą specifikaciją buvo atlikti sistemos projektavimo darbai projektuojant modelius pagal tokius punktus: sistemos duomenų modelis, sistemos mmeniu struktūra, sistemos naudotojų sąsajos prototipai.

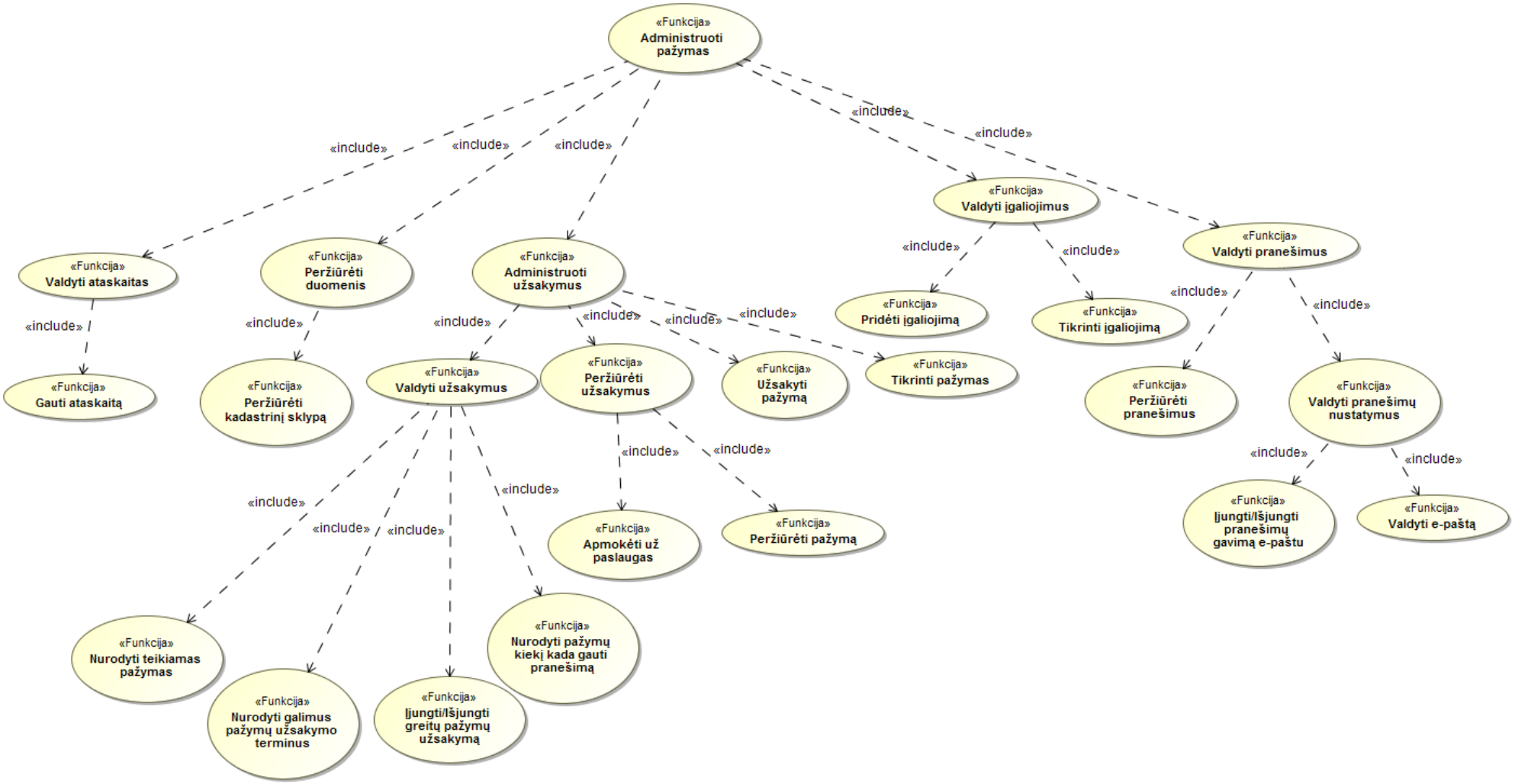
Eksperimento metu buvo tiriama kaip pavyksta atlikti sistemos specifikacija naudojantis *ODRES* metodu ir atlikti sistemos projektavimą remiantis sudaryta specifikacija pagal tokius kriterijus:

1. Funkcijų hierarchijos sudarymas;
2. Funkcijų priklausomumo aktoriams sudarymas;
3. Rezultatų specifikavimas;
4. Duomenų šaltinių specifikavimas;
5. Duomenų srautų specifikavimas (1 variantas);
6. Duomenų srautų specifikavimas (2 variantas);
7. Duomenų šaltinių apdorojimo etapų specifikavimas;
8. Būsenų kaitos specifikavimas;
9. Duomenų modelio sudarymas;
10. Sistemos meniu struktūros sudarymas;
11. Naudotojo sąsajos prototipų sudarymas.

5.2. Eksperimento rezultatai

5.2.1. Funkcijų hierarchija

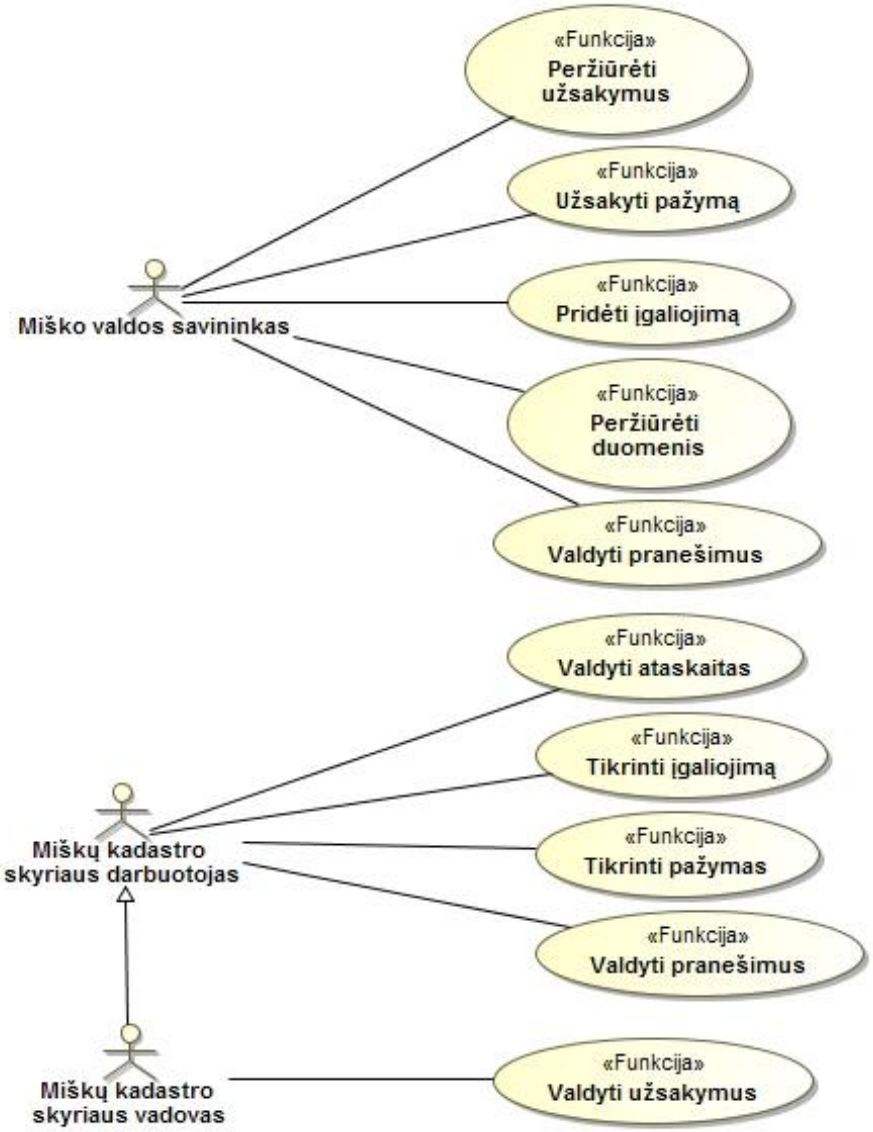
Pirmasis *ODRES* metodo proceso etapas yra funkcijų hierarchijos specifikavimas. Sistemos specifikavimas pradedamas nuo funkcijų hierarchijos sudarymo. Buvo surinkti sistemos reikalavimai iš atsakingų asmenų valstybinėje miškų tarnyboje. Sistemos funkcijos specifikuojamos panaudojimų atvejų diagrama. MKDTP pagrindinė funkcija yra administruoti pažymas, kuri apima visą posistemio veikimą. Prie šio panaudojimo atvejo jungiami smulkesni, detalizuojantys panaudojimo atvejai include ryšiu. Funkcijų hierarchija gali turėti keletą lygių kol pasiekiamas reikiamas funkcijų detalumas. Visgi žemiausio lygio funkcijos negali būti atominės, vienos operacijos funkcijos. Sudaryta miškų kadastro duomenų teikimo posistemio funkcijų hierarchija pateikiama 5.1 paveiksle.



5.1 pav. MKDTP funkcijų hierarchija

5.2.2. Funkcijų priklausomumas aktoriams

Jeigu specifikuojama sistema turi keletą skirtingų aktorių, kurie atlieka atskiras funkcijas, tai reikia specifikuoti aktorių funkcijas. MKDTP turi 3 aktorių tipus: miško valdos savininkas, miškų kadastro skyriaus darbuotojas ir miškų kadastro skyriaus vadovas. Šių trijų aktorių funkcijos, ir aktorių tarpusavio funkcijų paveldimumas vaizduojamas 5.2 paveiksle. Specifikuojant aktorių funkcijas, jeigu visos žemesnio lygio funkcijos priklauso tik vienam aktoriui, tai modeliuojant jam priskiriama aukštesnio lygio funkcija, taip supaprastinant ir palengvinant diagramos skaitomumą.



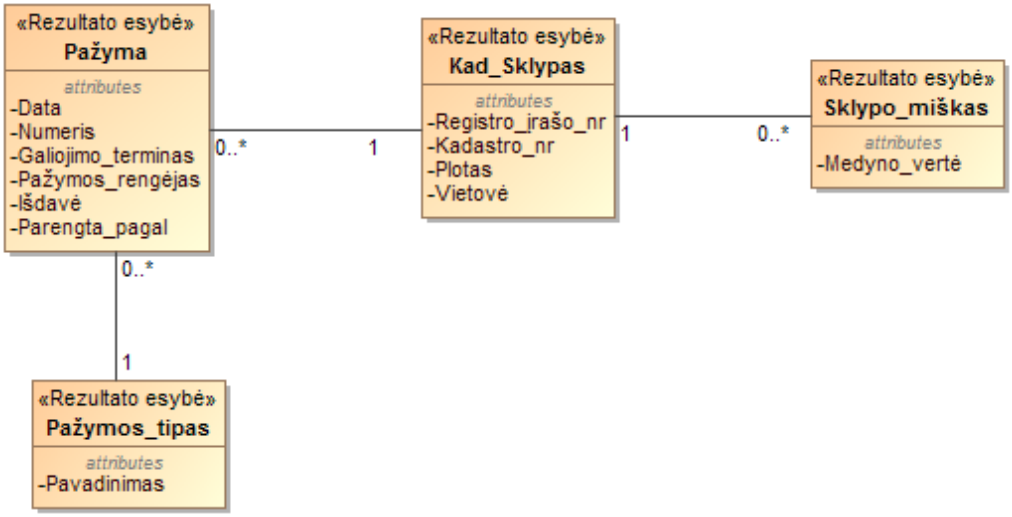
5.2 pav. Aktorių funkcijos

5.2.3. Duomenų šaltiniai ir rezultatai

Antrasis *ODRES* metodo proceso etapas yra duomenų šaltinių ir rezultatų specifikavimas. Specifikuojant duomenų šaltinius ir rezultatus, kiekvienam atskirai sukuriama klasių diagrama, kurioje aprašoma duomenų šaltinio arba rezultato struktūra, juos sudarančios esybės, šių tarpusavio ryšiai. Duomenų šaltiniu arba rezultatu gali būti įvairūs organizacijoje esantys dokumentai, esančių liktinių sistemų ar duomenų bazių informacija, žodiniai pranešimai, įvairios dyomenų esybės egzistuojančios organizacijoje. Visa tai išgryninama, atskiriama kas yra rezultatai, kas yra duomenų šaltiniai, ir kiekvienam atskirai sudaroma specifikacija.

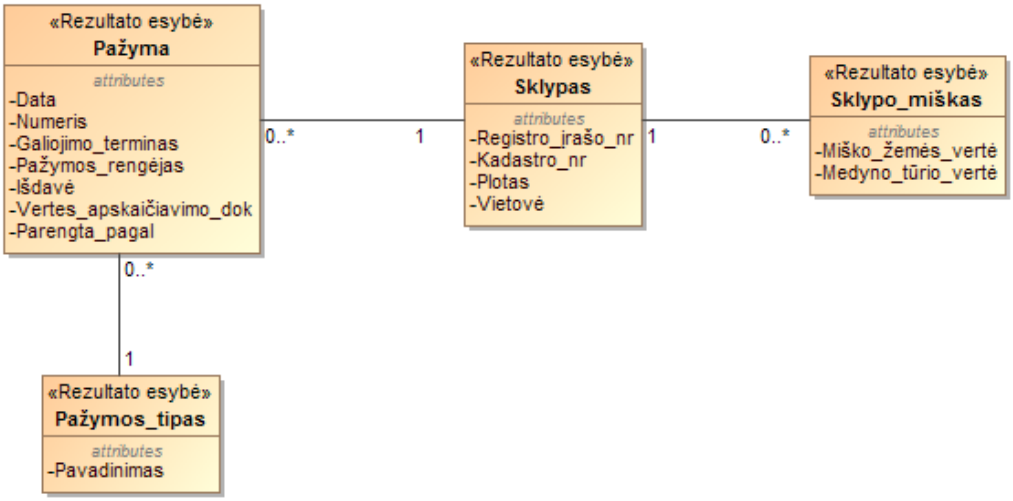
5.2.3.1. Rezultatai

MKDTP sistemoje rezultatai yra teikiamos pažymos miško valdos savininkams, bei darbuotojams svarbios yra sudaromos užsakytų pažymų ataskaitos. Visi rezultatai, pažymų pavyzdžiai ir pažymų ataskaitos pavyzdys buvo pateikti porpierinėje formoje, pagal tai ir sudarytos rezultatų specifikacijos. Pirmasis rezultatas yra pažyma - medynų vidutinė rinkos vertė, šiai pažymai sudaryta specifikacija pateikiama 5.3 paveiksle.



5.3 pav. Pažymos medynų vidutinė rinkos vertė specifikacija

Antrasis rezultatas yra dar viena pažyma - miško žemės ir medynų tūrio kaina, šios pažymos specifikacija pateikiama 5.4 paveiksle



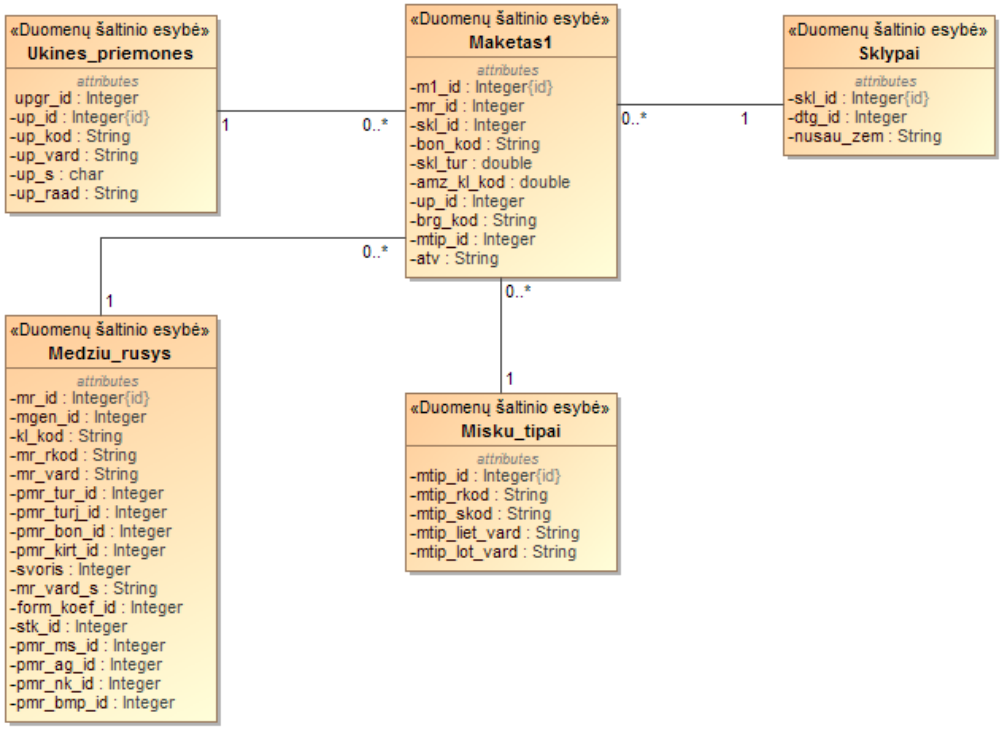
5.4 pav. Pažymos miško žemės ir medynų tūrio kaina specifikacija

Kitų rezultatų specifikacijos pateikiamos prieduose, 8.1 skyriuje.

5.2.3.2. Duomenų šaltiniai

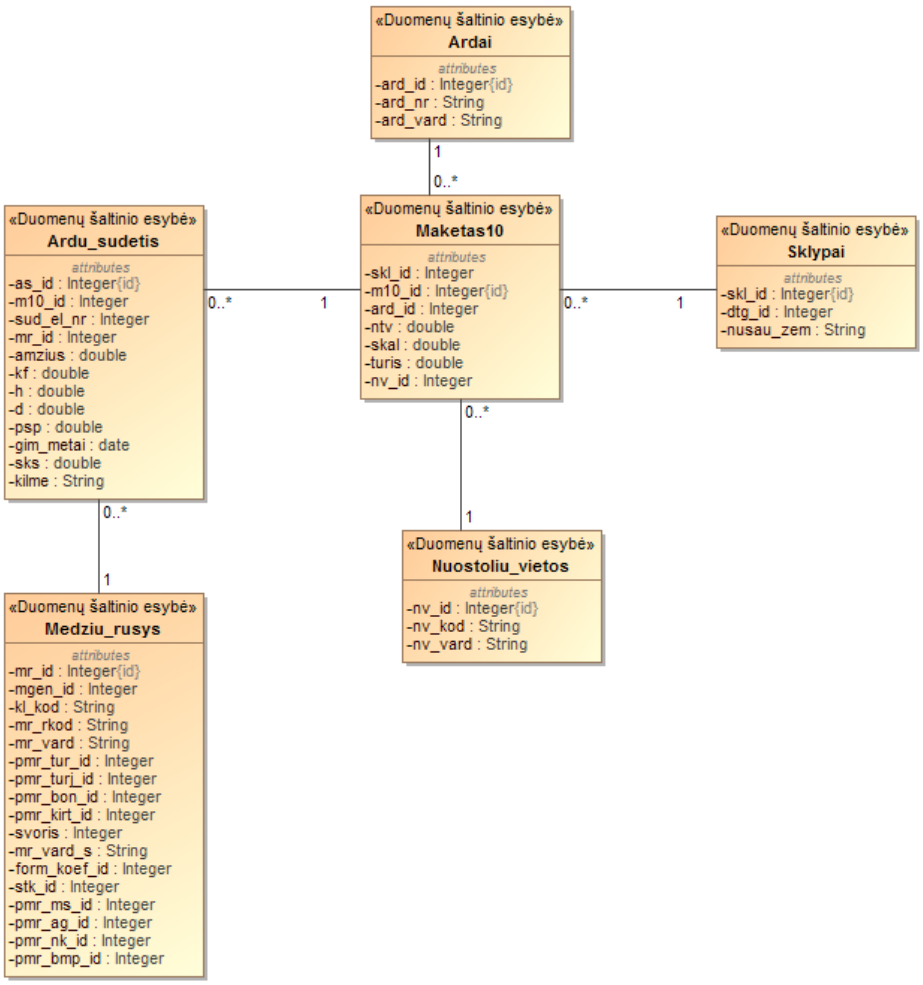
Valstybinė miškų tarnyba turi daug liktinių sistemų, kuriomis iki šiol naudojosi, ir projektas vykdomas siekiant praplėsti sistemų funkcionalumą ir atsinaujinti, todėl pagrindinis duomenų šaltinis yra esama duomenų bazė, kurioje buvo saugomi duomenys. Remiantis šia duomenų baze buvo sudarytos duomenų šaltinių specifikacijos, kurių pagalba gali būti formuojami rezultatai.

Pirmieji du duomenų šaltiniai vadinami „Maketas1“ ir „Maketas10“, šių duomenų šaltiniu pagrindu formuojamos pažymos ir atliekami skaičiavimai pažymų duomenims. Maketas1 duomenų šaltinio specifikacija pateikiama 5.5 paveiksle.



5.5 pav. Maketas1 duomenų šaltinio specifikacija

Maketas10 duomenų šaltinio specifikacija pateikiama 5.6 paveiksle.



5.6 pav. Maketas10 duomenų šaltinio specifikacija

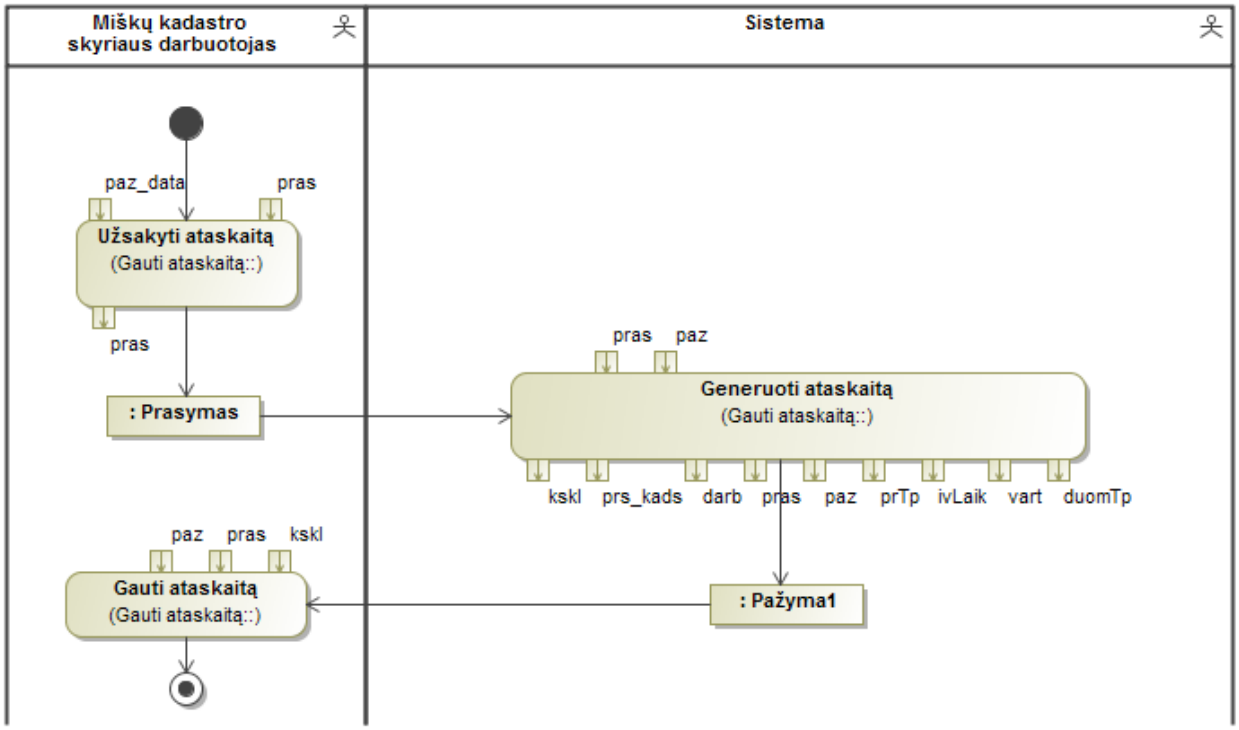
Kitų duomenų šaltinių specifikacijos pateikiamos prieduose, 8.1 skyriuje.

5.2.4. Duomenų srautai

Trečiasis *ODRES* metodo proceso etapas yra duomenų srautų specifikavimas. Kiekvienam rezultatui specifikuojami duomenų srautai ateinantys iš duomenų šaltinio ar šaltinių. Duomenų srautams specifikuoti yra du būdai. Pirmasis būdas, kai kuriamos keturios diagramos. Veiklos - duomenų srautams ir operacijoms tarp kurių tie duomenų srautai keliauja atvaizduoti. Komponentų - skirta atvaizduoti komponentui, kuris realizuojamas interfeisais, kurių pagalba atliekamos operacijos. Taip pat esant didesniam interfeisų specifikavimo detalumo lygiui, kiekvienam interfeisui galima kurti klasių diagramą, kurioje vaizduojamas interfeisas ir jo naudojamos duomenų esybės. Klasių diagrama skirta kiekvieno interfeiso specifikacijai atvaizduoti. Taip pat papildomai kiekvienam duomenų srautui vaizduojamam tarp operacijų veiklos diagramoje kuriama klasių diagrama, specifikuojama duomenų srauto struktūra. Antrasis būdas kai vienam rezultatui kuriamos dviejų tipų diagramos, paketų ir klasių. Paketų diagrama skirta vaizduoti paketus, kurie simbolizuoja duomenų šaltinį arba rezultatą, ir tarp jų vaizduojamas duomenų srauto ryšys, kuriam kuriama nuoroda į klasių diagramą detalizuojančiaą duomenų srautą ir realizuojančią atributų atsekamumą.

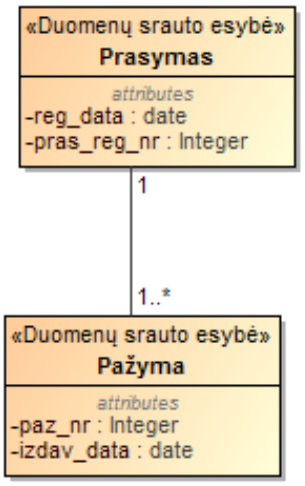
Duomenų srautams specifikuoti buvo pasirinktas pažymų ataskaitos rezultatas, nes turėjo didžiausią kiekį esybių ir atributų. Pradžioje paveiksluose pažymėtuose numeriais 5.6-5.10 bus pateikiamos diagramos duomenų srautų, kurie buvo specifikuoti pirmuoju būdu. Antrojo būdo duomenų srautų specifikavimo diagramos pateikiamos 5.11-5.12 paveiksluose.

Pažymų ataskaitos rezultato duomenų srautų veiklos diagrama pateikiama 5.6 paveiksle. Pradžioje miškų kadastro skyriaus darbuotojas turi užsakyti ataskaitą, užsakymo metu nurodo parametrus ataskaitai. Iš šios operacijos keliauja duomenų srautas prašymas, kurio specifikacija pavaizduota 5.7 paveiksle. Tada vykdoma operacija „Generuoti ataskaitą“, operacijos metu sugeneruojama ataskaitos duomenys, pagal gautus duomenų srauto „Prašymas“ duomenis. Iš šios operacijos duomenų srautu „Pažyma1“, kurios specifikacija pateikta 5.8 paveiksle duomenys perduodami operacijai „Gauti ataskaitą“ ir suformuojamas pažymų ataskaitos rezultatas.



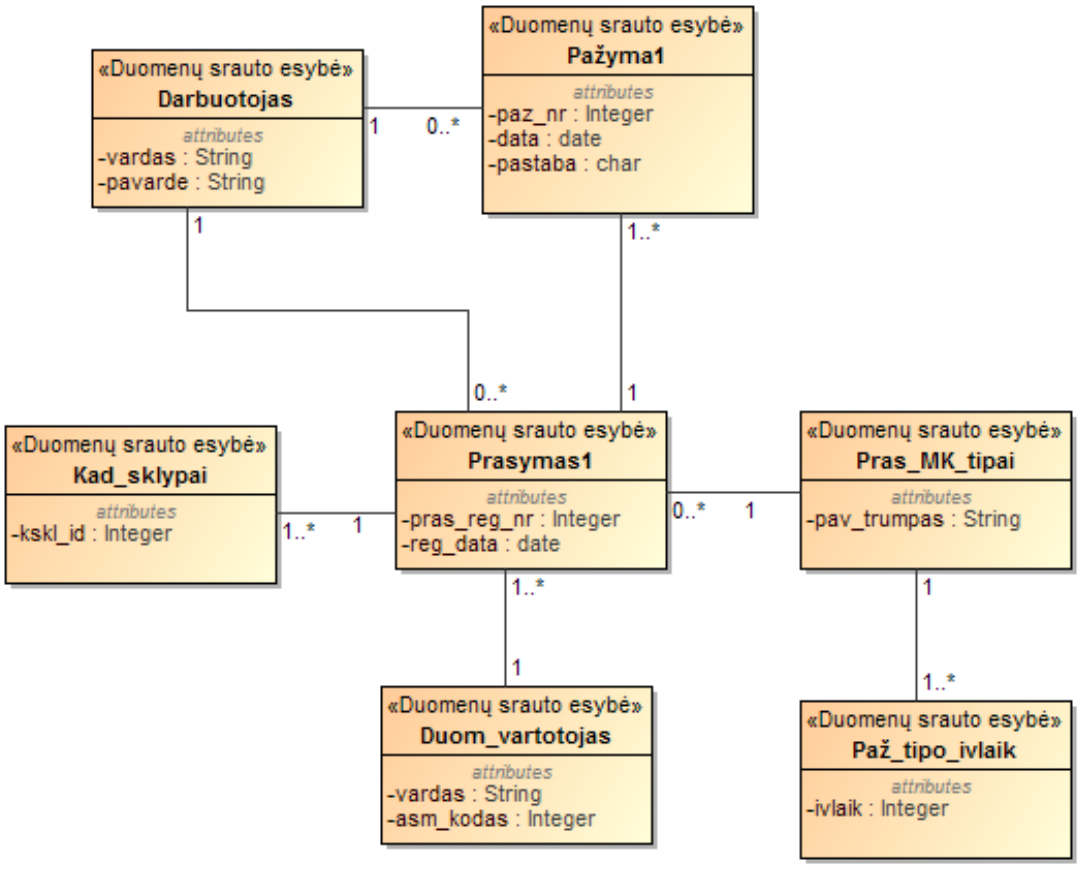
5.7 pav. Pažymų ataskaitos duomenų srautų veiklos diagrama

Duomenų srauto „Prašymas“ klasių diagrama, kurioje matomos duomenų srauto esybės ir jų atributai, pateikiama 5.8 paveiksle.



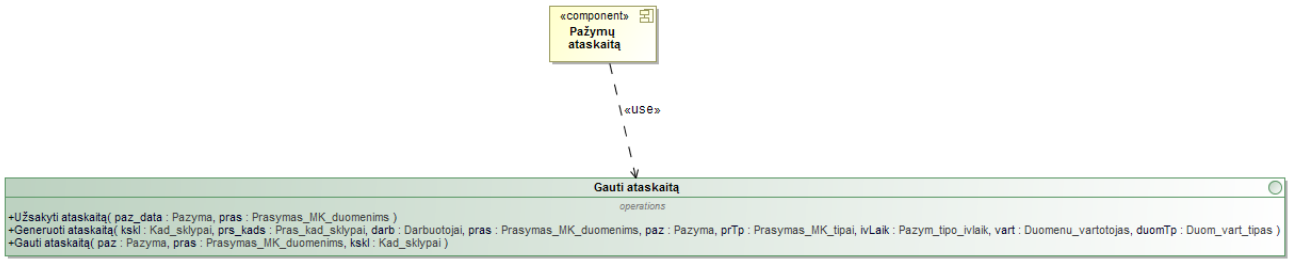
5.8 pav. Prašymas duomenų srautas

Duomenų srauto „Pažyma1“ klasių diagrama, kurioje matomos duomenų srauto esybės, jų atributai, tarpusavio ryšiai, pateikiama 5.9 paveiksle.



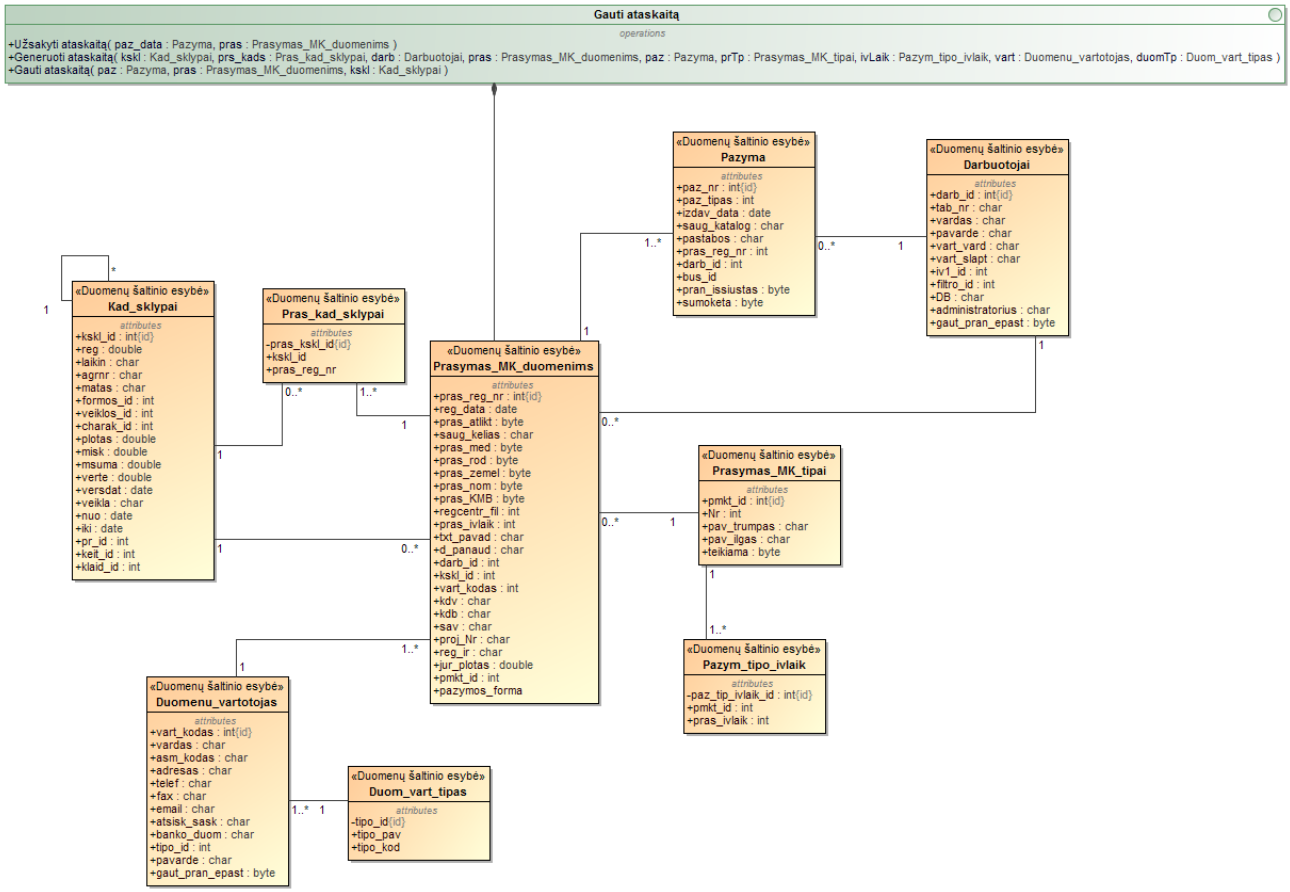
5.9 pav. Pažyma1 duomenų srautas

Pažymų ataskaitai realizuoti reikalingas vienas komponentas „Pažymų ataskaita“. Komponentas ir jam priklausantys interfeisai, šiuo atveju tik vienas interfeisas „Gauti ataskaitą“, pavaizduoti 5.10 paveiksle. „Gauti ataskaitą“ interfeisas realizuoja operacijas, kurios vaizduojamos pažymų ataskaitos duomenų srautų diagramoje.



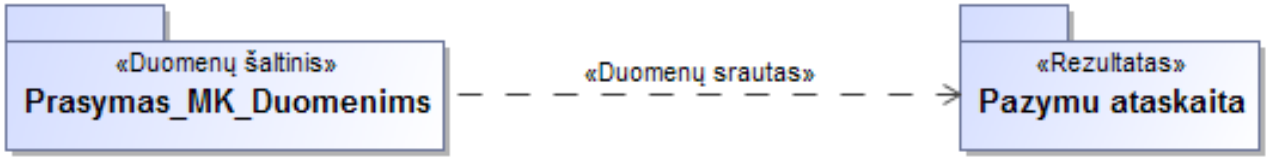
5.10 pav. Pažymų ataskaitos komponentų diagrama

Sudarytoje komponentų diagramoje pavaizduoti interfeisai specifikuojami kiekvienas atskirai, kiekvienam sukuriant atskirą klasių diagramą. Diagramoje vaizduojamas interfeisas ir jo naudojamos rezultato ar duomenų šaltinio esybės, jų struktūra. „Gauti ataskaitą interfeiso specifikacija pateikiama 5.11 paveiksle.



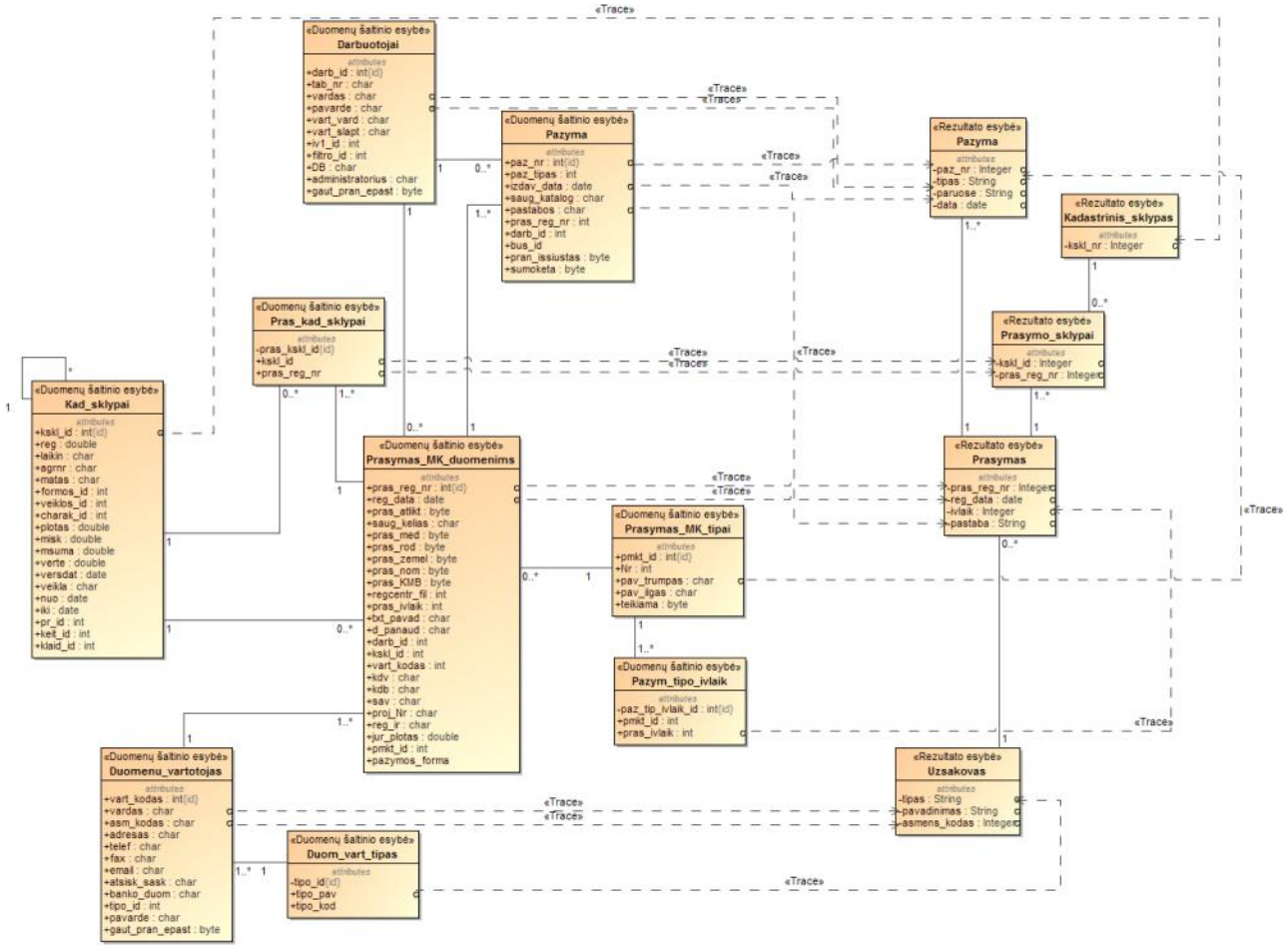
5.11 pav. Gauti ataskaitą interfeiso specifikacija

Antrojo duomenų srautų vaizdavimo būdo diagrama, pateikianti rezultato ir duomenų šaltinius, bei tarp jų esančius duomenų srautus, pateikiama 5.11 paveiksle. Diagramoje vaizduojamas „Pažymų ataskaitos“ rezultatas ir „Prasymas_MK_Duomenims“ duomenų šaltinis. Tarp jų yra duomenų srauto ryšys, kuriam pridėta nuoroda į atributų atsekamumo klasių diagramą kuri vaizduojama 5.12 pveiksle.



5.12 pav. Pažymų ataskaitos duomenų srautas tarp rezultato ir duomenų šaltinio

Norint pavaizduoti specifikuojamo rezultato ir duomenų šaltinio esybių atributų atsekamumą, t.y. kuris rezultato esybės atributas gaunamas iš vieno ar daugiau duomenų šaltinio esybių atributų, kuriama klasių diagrama. Joje vaizduojamos visos rezultato esybės ir visos reikiamos duomenų šaltinio esybės, bei jų atributai. Atsekamumui vaizduoti naudojamas „*Trace*“ ryšys, kuriuo jungiami duomenų šaltinio esybės atributai su rezultatų esybių atributais. Pažymų ataskaitos rezultato atributai gaunami iš duomenų šaltinio “Prasymas_MK_duomenims“ esybių atributų, jų atsekamumo diagrama pateikiama 5.13 paveiksle.



5.13 pav. Pažymų ataskaitos duomenų srautai atributų lygmenyje

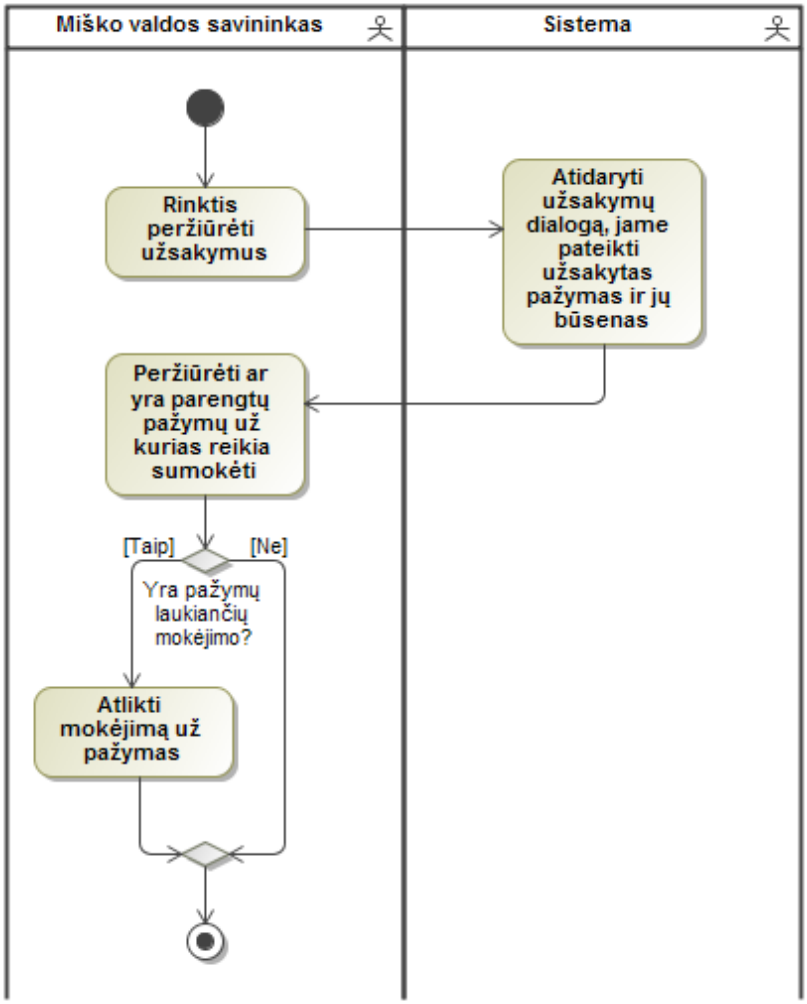
Atlikus duomenų srautų specifikavimą abiem būdais pastebėta, kad pirmuoju būdu reikia skirti žymiai daugiau laiko vieno rezultato duomenų srautams specifikuoti, artruoju būdu duomenų srautai specifikuojami daug greičiau. Pirmąjį specifikavimo variantą geriau naudoti kai reikia didelio detalumo vaizduojant duomenų srautus ir nėra svarbu kiek laiko bus sugaišta duomenų srautų specifikavimui.

5.2.5. Duomenų šaltinių apdorojimo etapai

Ketvirtojo *ODRES* metodo proceso specifikavimo etapo metu specifikuojami duomenų šaltinių apdorojimo etapai. Duomenų šaltiniai apdorojami funkcijomis suspecifikuotomis pirmojo etapo metu, todėl norint specifikuoti duomenų šaltinių apdorojimo etapus reikia specifikuoti kiekvieną funkcijų hierarchijos žemiausio lygio funkciją, jai sudarant veiklos diagramą. Specifikuodami funkcijas apimame visų duomenų šaltinių įvairias apdorojimo ir tuo pačiu rezultatų gavimo ar išvedimo veiklas.

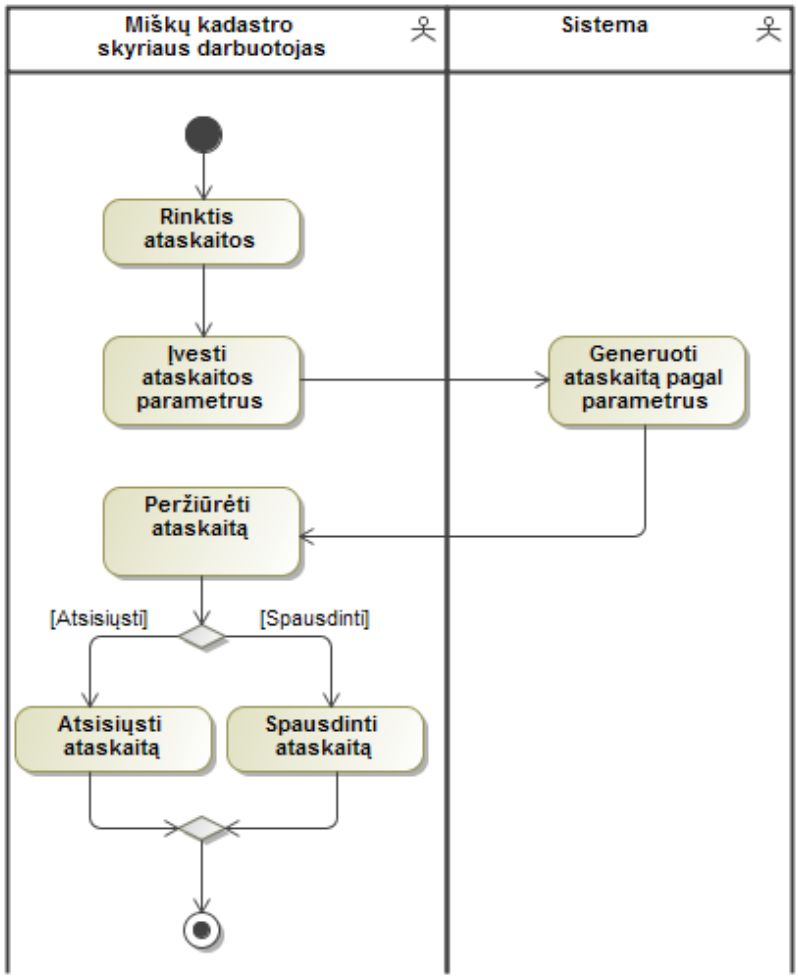
Pirmoji specifikuota funkcija yra „Apmokėti už paslaugas“, jos veiklos diagrama pateikiama 5.14 paveiksle. Pradžioje miško valdos savininkas renkasi peržiūrėti užsakymus. Sistema atidaro užsakymų peržiūros dialogą, kuriame pateikia asmens užsakytas pažymas ir jų būsenas. Jeigu

miško valdos savininkas randa parengtų ir neapmokėtų užsakymų, jis gali atlikti mokėjimą už pažymas, kad vėliau jas galėtų peržiūrėti ar atsisiųsti.



5.14 pav. Funkcijos „Apmokėti už paslaugas“ veiklos diagrama

Funkcijos „Gauti ataskaitą“ veiklos diagrama pateikiama 5.15 paveiksle. Miškų kadastro skyriaus darbuotojas turi pasirinkti ataskaitos užsakymą, tada suveda ataskaitos parametrus, pagal kuriuos bus generuojama ataskaita. Sistema sugeneruoja ataskaitą ir pateikia ją peržiūrai. Asmuo tada gali ataskaitą atsisiųsti arba atsispausdiniti, pagal savo norus.

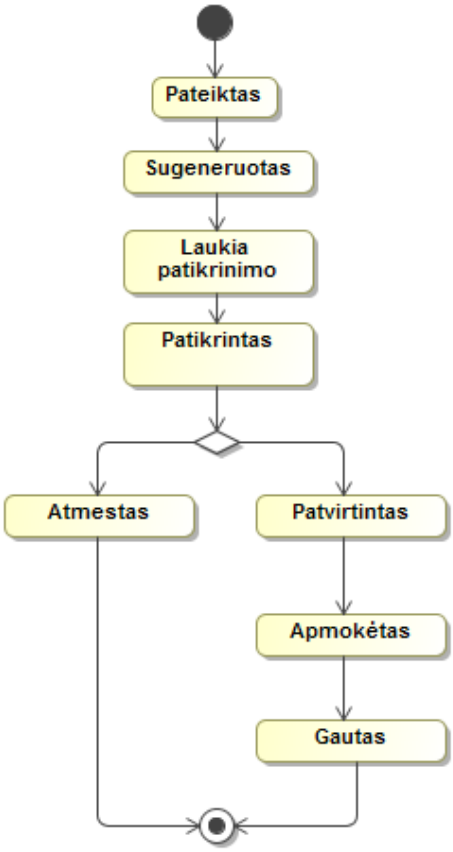


5.15 pav. Funkcijos „Gauti ataskaitą“ veiklos diagrama

Visų kitų funkcijų esančių pirmo etapo metu sudarytos funkcijų hierarchijos žemiausiame lygyje veiklos diagramos pateikiamos šio darbo prieduose, 8.1 skyriuje.

5.2.6. Būsenų kaita

Penktasis *ODRES* metodo proceso etapas yra busenų kaitos specifikavimas. Šio etapo metu svarbu specifikuoti sistemos objektų būsenas, kurios įtakoja duomenų pasikeitimus kiekvienos būsenos metu. MKDTP specifikavimo metu surasta tik vienas svarbus objektas - „Užsakymas“, kurio būsenų kaitos metu vyksta duomenų pasikeitimai. Pradžioje miško valdos savinikas pateikia užsakymą pažymoms gauti. Pateikus užsakyma pradedamas pažymų generavimas. Kai pažymos sugeneruojamos, jas turi patikrinti miškų kadastro skyriaus darbuotojas. Kai darbuotojas pažymas patikrina, jis jas gali atmesti arba patvirtinti. Patvirtintas pažymas turi apmokėti miško valdos savininkas, kuris tas pažymas užsakė, ir tik gavus apmokėjimą, pažymas gauna miško valdos savininkas. Užsakymo būsenų kaitos diagrama pateikiama 5.16 paveiksle.

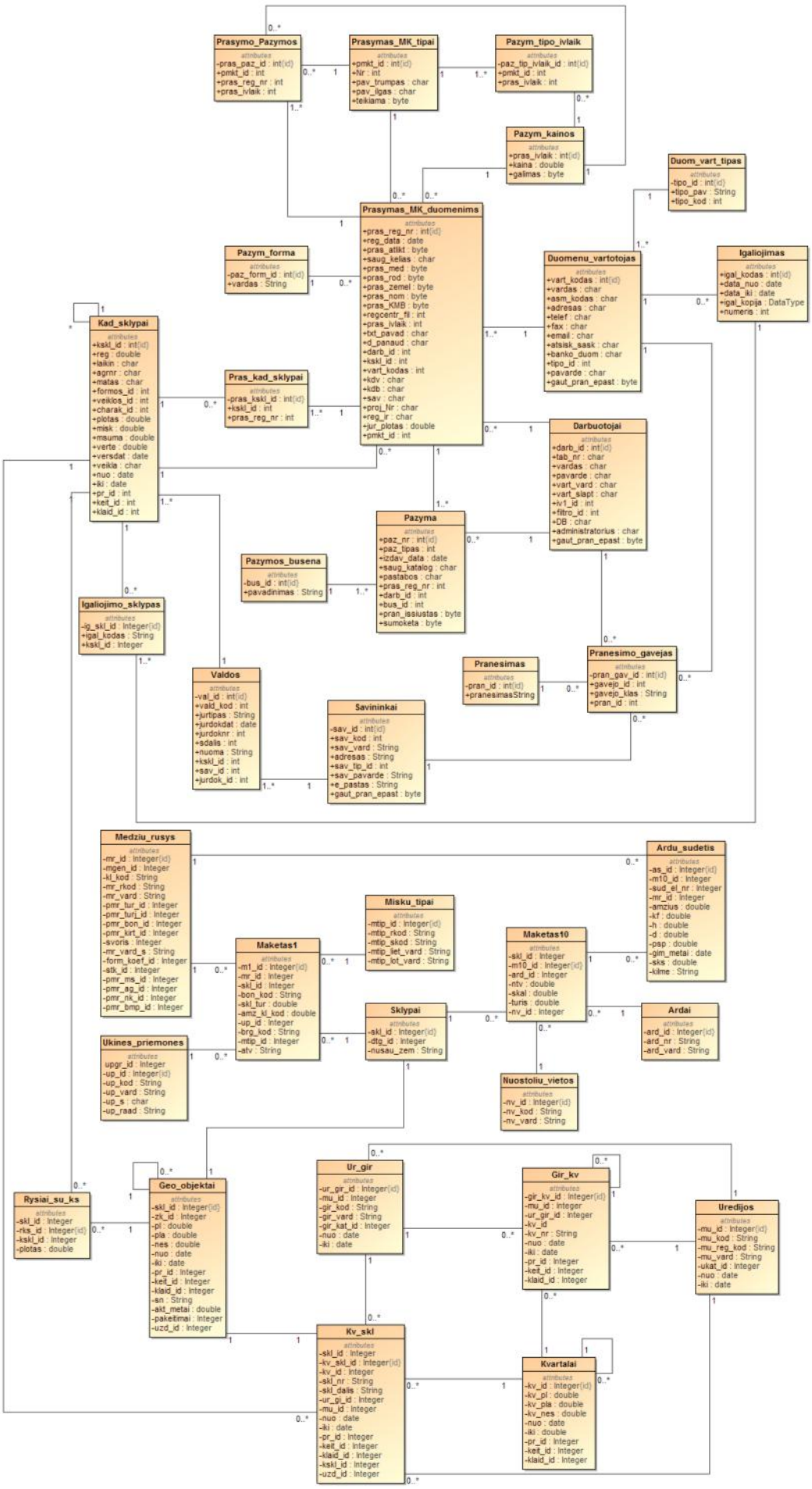


5.16 pav. Užsakymo būsenų kaitos diagrama

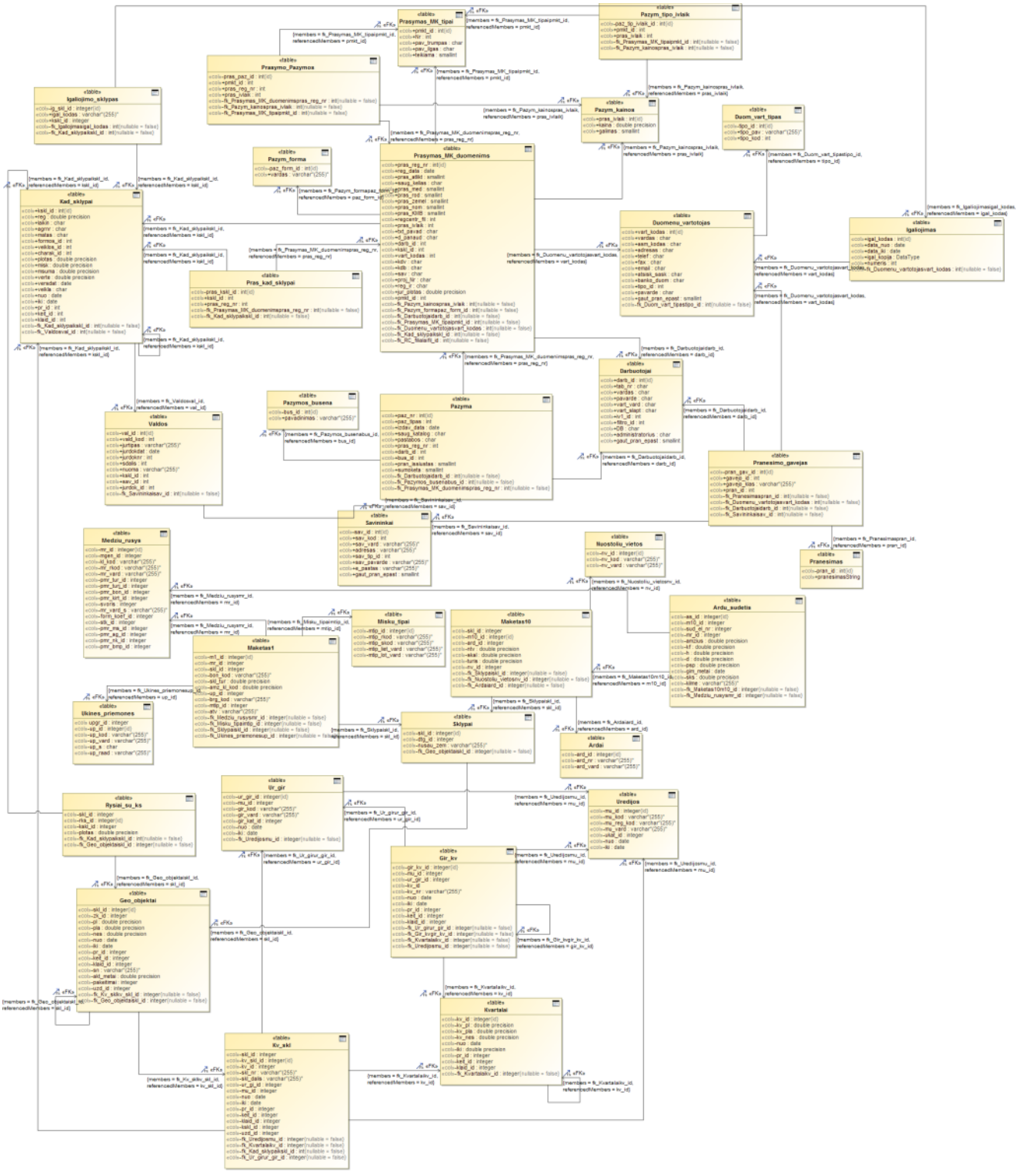
5.2.7. Duomenų modelis

Atlikus visus informacinės sistemos reikalavimų specifikavimo etapus pradedamas sistemos projektavimas. Projektuoti pradedama nuo sistemos duomenų modelio. Kadangi modeliuojant duomenų srautus buvo nustatyta, kad visi rezultatų esybių atributai turi juos formuojančius reikiamus duomenų šaltinių esybių atributus, duomenų modelis formuojamas remiantis tik duomenų šaltinio specifikacijom. Priešingu atveju pradžioje reikia remtis duomenų šaltinių specifikacijom, ir tada į duomenų modelį įtraukti trūkstamus režultatų esybių atributus ar pačias esybes. Buvo pasirinktas duomenų šaltinis turintis daugiausiai duomenų šaltinio esybių ir iš jo suformuotas pradinis duomenų modelis. Vėliau imami po vieną kiekvienas kitas duomenų šaltiniai ir buvo jungiamos kiekvieno duomenų šaltinio esybės po vieną į bendrą duomenų modelį. Jungiant pradžioje ieškoma ar yra sutampanti esybė, jeigu nėra, duomenų šaltinio esybė įtraukiama į bendrą duomenų modelį su visais savo atributais. Jeigu esybės sutampa, tada lyginami esybių atributai ir į duomenų esybę įtraukiami nesutampantys duomenų šaltinio esybės atributai. Suformuotas bendras miškų kadastro integruotos informacinės sistemos duomenų teikimo posistemio duomenų modelis pateikiamas 5.17 paveiksle.

Turintpilnai paruoštą duomenų modelio klasių diagramą atliekama transformacija į *SQL* duomenų bazės struktūrą. Transformacija atliekama *MagicDraw* įrankio pagalba, kuris turi klasių diagramos transformavimo į įvairių duomenų bazių schemas galimybę. Transformuojant automatiškai sukuriami papildomi atributai duomenų lentelėse, reikalingi duomenų bazės realizavimui. Transformuotos duomenų bazės struktūra pateikiama 5.18 paveiksle.



5.17 pav. Duomenų modelis

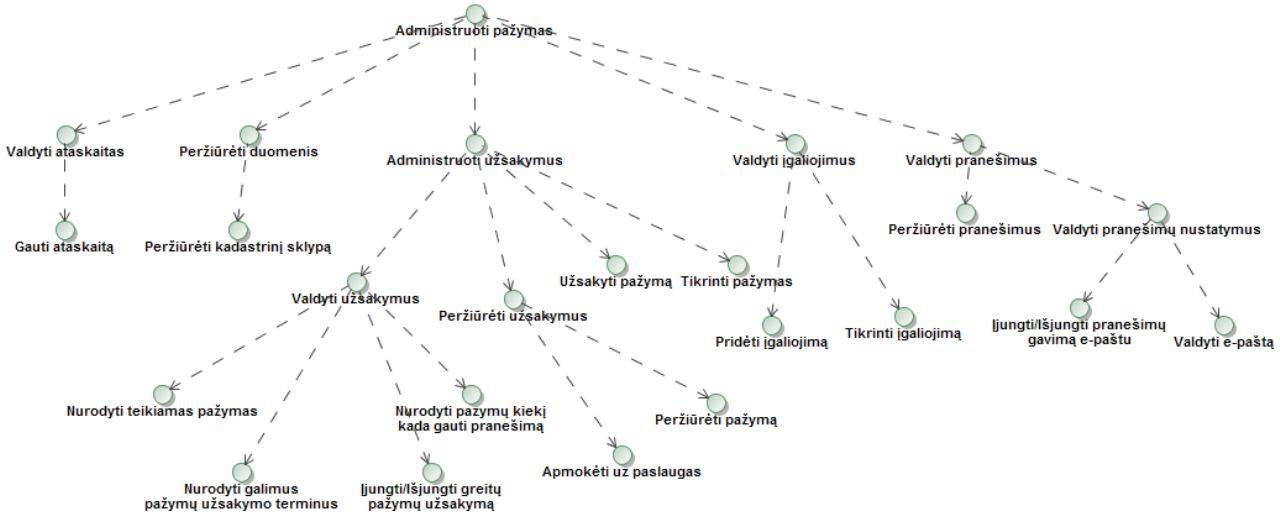


5.18 pav. Duomenų bazės schema

5.2.8. Sistemos meniu struktūra

Antrasis projektavimo etapas yra sistemos meniu struktūros sudarymas. Projektuojamos sistemos meniu struktūra buvo sudaryta remiantis sistemos specifikavimo metu sudaryta funkcijų hierarchija. Realizuojant sistemą bus penki pagrindiniai meniu punktai: valdyti ataskaitas, peržiūrėti duomenis, administruoti užsakymus, valdyti įgaliojimus, valdyti pranešimus. Kiekvienas meniu elementas turi jį realizuojančius elementus, taigi realizuojant sistemą pagrindiniai meniu punktai bus išskleidžiami ir jie turės papildomo funkcionalumo meniu punktus, kurie taipogi gali būti

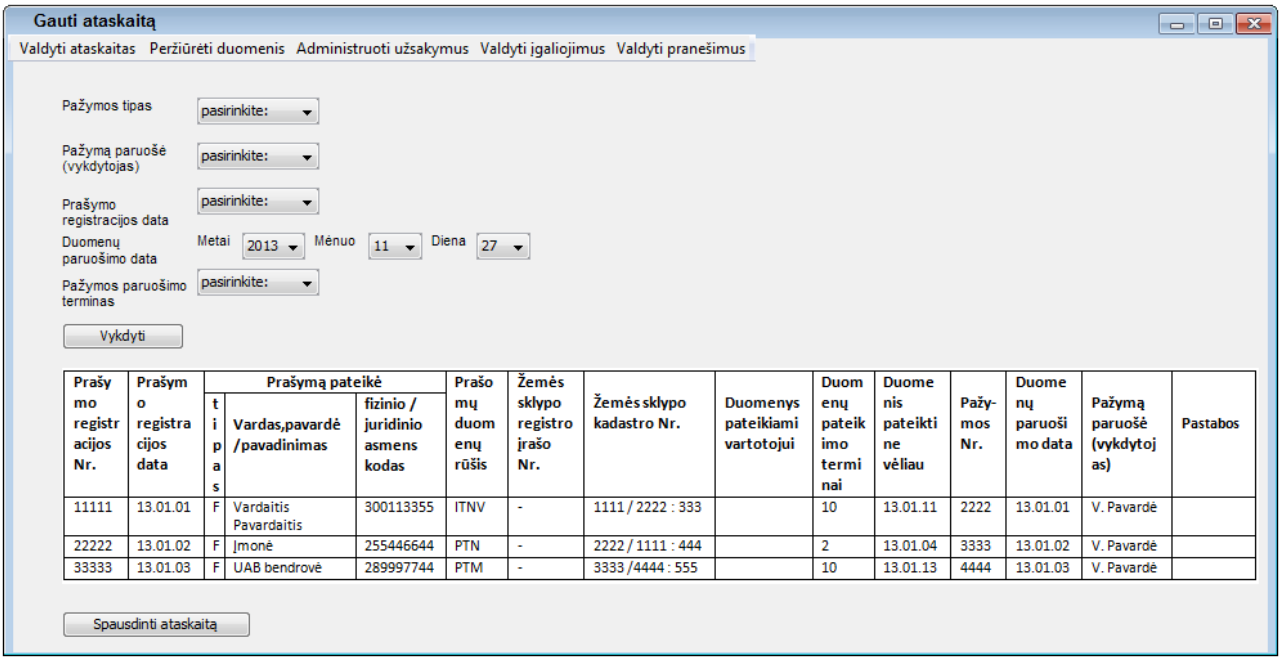
detalizuojami žemesnio lygio meniu elementais. Sistemos meniu struktūra pateikiama 5.19 paveiksle.



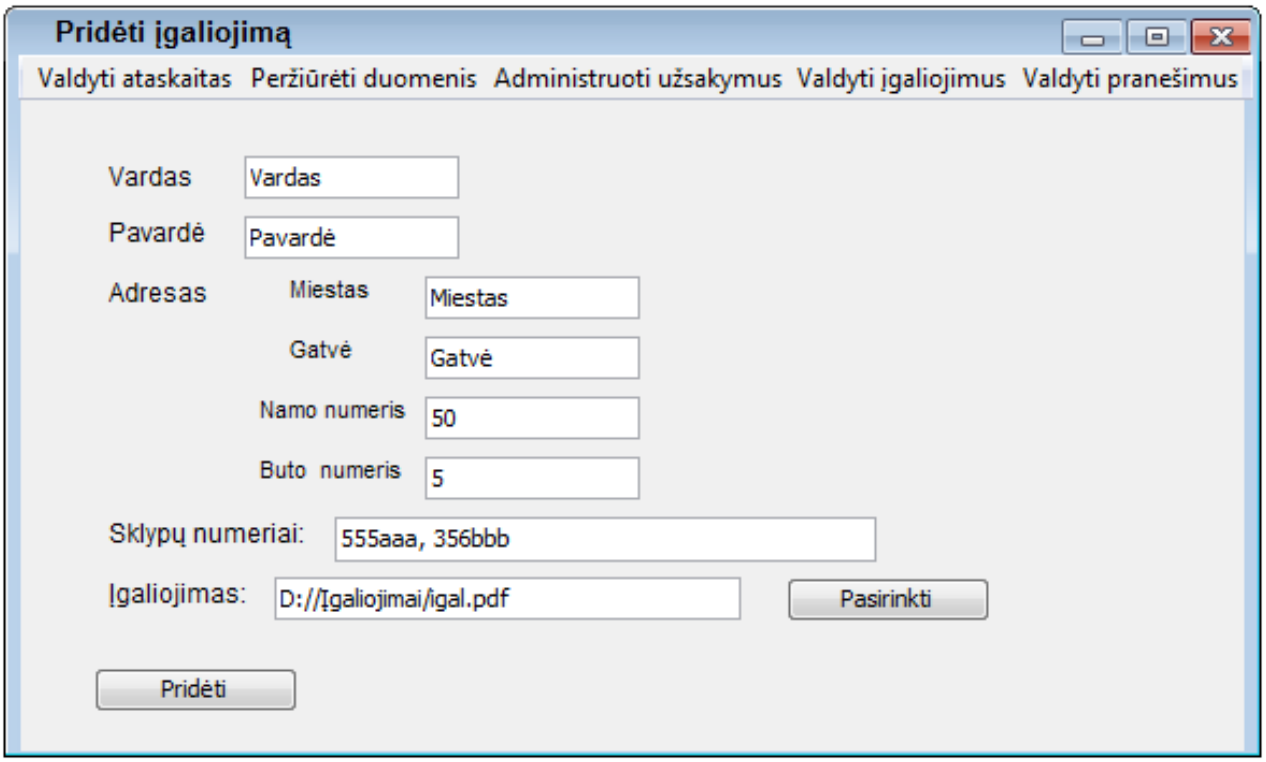
5.19 Sistemos meniu struktūra

5.2.9. Naudotojo sąsajos prototipai

Trečiasis sistemos projektavimo etapas yra vartotojo sąsajos prototipų projektavimas. Šio etapo metu sudaromi sistemos naudotojų sąsajos prototipai, su visais elementais ir pavyzdiniais duomenimis. Prototipai sudaryti remiantis duomenų šaltinių apdorojimo etapų specifikacijomis, duomenų modelio esybėmis ir jų atributais bei sudaryta meniu struktūra. Naudotojo sąsajos prototipai sudaryti žemiausio lygmens meniu struktūros elementams, kadangi jie atspindi pagrindines veiklas ir funkcijas, kurias galės atlikti naudotojas sistemoje. Paveiksluose pažymėtuose numeriais 5.20 ir 5.21 pateikiami atitinkamai „Gauti ataskaitą“ ir „Pridėti įgaliojimą“ meniu elementų funkcijoms vykdyti naudotojo sąsajos prototipai. Kiti sudaryti naudotojo sąsajos prototipai pateikiami šio darbo prieduose, 8.1 skyriuje.



5.20 Ataskaitos gavimo naudotojo sąsajos prototipas



5.21 pav. Įgaliojimo pridėjimo naudotojo sąsajos prototipas

5.3. Sprendimo veikimo ir savybių analizė, kokybės kriterijų įvertinimas

Eksperimento metu buvo atlikta pasirinktos dalykinės srities informacinės sistemos specifikacija remiantis *ODRES* metodo procesu ir projektavimas remiantis sudaryta specifikacija. Visus *ODRES* metodo proceso žingnius pavyko sėkmingai realizuoti specifikuojant sistemą. Specifikuojant duomenų srautus nustatyta, kad geriausia naudoti antrąjį duomenų srautų specifikavimo variantą. Vėliau buvo sudarytas sistemos projektas pagal specifikaciją. Visų kriterijų sąrašas ir jų įvertinimai pateikiami 5.1 lentelėje.

5.1 lentelė. Eksperimentinio tyrimo kriterijų įvertinimas

Kriterijus	Įvertinimas
Funkcijų hierarchijos sudarymas	Sudaryti funkcijų hierarchiją pakankamai lengva, svarbiausia skaidant funkcijas iki žemiausio lygio, neišskaidyti iki atominių vienos operacijos funkcijų.
Funkcijų priklausomumo aktoriams sudarymas	Funkcijų priklausomumo aktoriams diagrama naudinga norint detalizuoti aktorių ar jų grupių funkcijas, taip lengva nustatyti prie kokių duomenų gali prieiti vartotojas, kokias funkcijas sistemoje galės atlikti.
Rezultatų specifikavimas	Kiekvienas rezultatas lengvai ir pakankamai greitai specifikuojamas standartine klasių diagrama.
Duomenų šaltinių specifikavimas	Kiekvienas duomenų šaltinis lengvai ir pakankamai greitai specifikuojamas standartine klasių diagrama.
Duomenų srautų specifikavimas (1 variantas)	Pakankamai sudėtingas, ilgai užtrunka vieno rezultato duomenų srautams specifikuoti, lengva padaryti klaidų, rekomenduojama naudoti tik išskirtiniais atvejais, kai reikia labai nuodugniai ir detaliai specifikuoti duomenų srautus, yra svarbi sistemos klaidos galimybė, kritinė sistema.
Duomenų srautų specifikavimas (2 variantas)	Lengvesnis negu pirmasis duomenų srautų specifikavimo variantas, atlikus specifikavimą lengva atsekti reikalingus

	duomenų šaltinių esybių atributus, pakankamai greitai specifikuojama., rekomenduojama naudoti dėl greito ir patogaus specifikavimo.
Duomenų šaltinių apdorojimo etapų specifikavimas	Atlikus duomenų šaltinių apdorojimo etapų specifikavimą aprašomas sistemos veikimas, lengva surasti ar yra trūkstamų funkcijų.
Būsenų kaitos specifikavimas	Atlikus būsenų kaitos specifikaciją lengva nustatyti ar nėra praleistų funkcijų.
Duomenų modelio projektavimas	Pagal sudarytą duomenų modelį galima realizuoti kuriamos informacinės sistemos duomenų bazę.
Sistemos meniu struktūros projektavimas	Meniu struktūra sudaroma remiantis funkcijų hierarchija, jeigu specifikuojant funkcijų hierarchija nepadaryta klaidų, meniu struktūra padaroma labai lengvai.
Naudotojo sąsajos prototipų projektavimas	Projektuojant vartotojo sąsajos prototipus galima surasti ar nepadaryta klaidų sudarant duomenų modelį, specifikuojant duomenų šaltinių apdorojimo etapus.

5.4. Sprendimo taikymo rekomendacijos

Rekomendacijos, kurių reikėtų laikytis specifikuojant informacinę sistemą naudojantis *ODRES* metodo procesu:

- Aktorių priklausomumo funkcijoms diagrama sudaroma kai sistema naudosis du ar daugiau skirtingų aktorių.
- Specifikuojant duomenų srautus reikia naudoti tik vieną duomenų srautų specifikavimo variantą, taip lengviau ir aiškiau bus skaitoma specifikacija.
- Duomenų srautams sudaryti rekomenduojama naudoti antrąjį variantą, taip duomenų srautai specifikuojami greičiau ir lengva atsekti ar yra visi reikalingi atributai rezultato esybėms formuoti.
- Sudarant duomenų modelį pradinių esybių itraukimui geriausiai pasirinkti duomenų šaltinį ar rezultatą turintį daugiausiai esybių.
- Busenų kaitos diagramas rekomenduojama sudaryti tik tiems specifikuojamos sistemos objektams, kurių būsenų kaitos metu vyksta ir duomenų pasikeitimai sistemoje.

6. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Kadangi *ODRES* metodas neturi standartizuotos notacijos ir nėra tinkamas bei patogus specifikavimo metodas skirtas šiuolaikiniams įrankiams, šiame darbe siekiama surasti standartizuotos modeliavimo kalbos notacijos atitikmenis *ODRES* metodo notacijai, kad būtų galima specifikuoti IS naudojantis *ODRES* metodo procesu.
2. Buvo atlikta trijų modeliavimo kalbų analizė: *IDEF*, *BPMN* ir *UML*. Modeliavimo kalbos buvo palygintos ir pasirinkta *UML* modeliavimo kalba *ODRES* metodo pritaikymui. *IDEF* kalba nuvo atmesta dėl mažo populiarumo ir modeliavimo įrankių trūkumo, *BPMN* buvo atmesta dėl per daug siauros taikymo srities, kas neleistų realizuoti visus *ODRES* metodo procesą žingsnius.
3. Darbo metu buvo išanalizuotas *ODRES* metodas, jo procesas, kas leido suprasti metodo galimybes. *ODRES* metodo procesą sudaro šeši etapai: funkcijų hierarchijos sudarymas, duomenų šaltinių ir rezultatų struktūros specifikavimas, duomenų srautų specifikavimas, duomenų šaltinių apdorojimo etapų specifikavimas, būsenų kaitos specifikavimas, ryšių tarp duomenų šaltinių detalizavimas ir ryšių tarp duomenų šaltinių būsenų specifikavimas. Metodą buvo nuspręsta pritaikyti *UML* modeliavimo kalboje. Visgi buvo nuspręsta atsisakyti šeštojo proceso žingsnio, kadangi labai detalus būsenų, duomenų šaltinių ir ryšių tarp jų specifikavimas reikalingas retais atvejais, specifikuojant labai sudėtingas sistemas.
4. Atlikta modeliavimo įrankių *MagicDraw* ir *Visual Paradigm* analizė. *ODRES* metodo pritaikymui pasirinktas *MagicDraw* modeliavimo įrankis dėl šiek tiek platesnių modeliavimo galimybių, dėl galimybių paruošti dokumentų generavimo šablonus, taip pat įrankis yra naudojamas mokymo tikslais Kauno technologijos universitete, todėl jo galimybės yra puikiai žinomos.
5. Sudaryti ir aprašyti reikalavimai kaip *ODRES* metodo proceso pagalba reikia atlikti sistemos specifikavimą. Pateikiamas pats procesas, kiekvienam proceso žingsniui tinkamos diagramos ir jų elementai. Specifikuojant naudojamos klasių, panaudojimo atvejų, veiklos, paketų, komponentų ir būsenų diagramos.
6. Atliktas pasirinktos dalykinės srities Valstybinės miškų tarnybos „Miškų kadastro integruotos informacinės sistemos“ posistemio specifikavimas pagal *ODRES* metodo proceso schemą naudojantis *UML* notacija ir sugalvotu sprendimo aprašu. Daugiausiai problemų sudarė duomenų srautų specifikavimas, nes norint specifikuoti vieno rezultato duomenų srautus reikia modeliuoti eilę diagramų, kas kainuoja daug analitiko laiko. Visgi buvo sugalvoti du duomenų srautų specifikavimo variantai. Rekomenduojama naudoti antrąjį, kuomet specifikuojama paketų diagrama, kurioje paketai vaizduoja duomenų srautus ir rezultatus, o ryšiai tarp jų duomenų srautus. Taipogi duomenų srautų ryšiai detalizuojami klasių diagramomis detalizuojant atributų atsekamumą.
7. Sudaryti ir aprašyti reikalavimai kaip atlikti sistemos projektavimą remiantis sudaryta specifikacija trims sistemos projektavimo aspektams: duomenų modelio sudarymo, sistemos meniu struktūros sudarymo, naudotojo sąsajos prototipų sudarymo. Nustatyta, kad šių aspektų projektavimui reikalingos klasių ir komponentų *UML* diagramos, o vartotojo sąsajos prototipai projektuojami naudotojo sąsajos modeliavimo diagrama , kuri nėra *UML* kalbos ir notacijos dalis.
8. Atlikus eksperimentinį tyrimą buvo sudarytos ir pateiktos *ODRES* metodo gairės bei metodo naudojimo gerosios praktikos rekomendacijos, kaip geriausia specifikuoti sistemą naudojantis *ODRES* metodo procesu *UML* aplinkoje.

7. LITERATŪRA

[1] J. Valacich and C. Schneider, "Information Systems Today," in *Information Systems Today*, 5 ed., Prentice Hall, 2011, p. 576.

[2] R. Butkienė ir R. Butleris, „The Approach for the User Requirements Specification," *5th East-European conference ADBIS'2001,* pp. 225-240, 2001.

[3] R. Butkienė ir R. Butleris, „Verification Rules of Computerised Information System Model with Respect to Data Resource Processing," *Informatica,* t. 12, pp. 347-372, 2001.

[4] M. Pečiulaitis, *Magistro darbas „Automatizuotas internetinių IS vartotojų sąsajų kūrimas",* Kaunas, 2006.

[5] A. Aleksandravičienė, T. Danikauskas, R. Butleris ir K. Šidlauskas, „Duomenų modelio automatizuoto sudarymo prototipo funkcionalumo tyrimas," *Informacijos mokslai,* t. 32, pp. 118-127, 2005.

[6] A. Aleksandravičienė, R. Butleris ir T. Danikauskas, „Duomenų modeliavimas informacijos srautų specifikacijos pagrindu," *Informacinės technologijos,* pp. 473-478, 2004.

[7] T. Danikauskas ir R. Butleris, „Approach for IS workspace design based on output driven requirements specification," *Information Technology and Control,* t. 35, pp. 144 - 156, 2006.

[8] „IDEF metodų dokumentacija," [Tinkle]. Available: http://www.idef.com/Downloads.htm. [Kreiptasi 13 05 2015].

[9] C. H. Kim, R. H. Weston, A. Hodgson ir K. H. Lee, „The complementary use of IDEF and UML modelling approaches," *Computers in Industry,* t. 50, pp. 35-56, 2003.

[10] „OMG BPMN 2.0 specifikacija," [Tinkle]. Available: http://www.omg.org/spec/BPMN/2.0/. [Kreiptasi 13 05 2015].

[11] „OMG UML 2.4.1 specifikacija," [Tinkle]. Available: http://www.omg.org/spec/UML/. [Kreiptasi 13 05 2015].

[12] „No Magic produktų ir dokumentacijos tinklalapis," No Magic, [Tinkle]. Available: http://www.nomagic.com/products/magicdraw.html. [Kreiptasi 13 05 2015].

[13] „Visual Paradigm produkto ir dokumentacijos tinklalapis," Visual Paradigm International, [Tinkle]. Available: http://www.visual-paradigm.com. [Kreiptasi 13 05 2015].

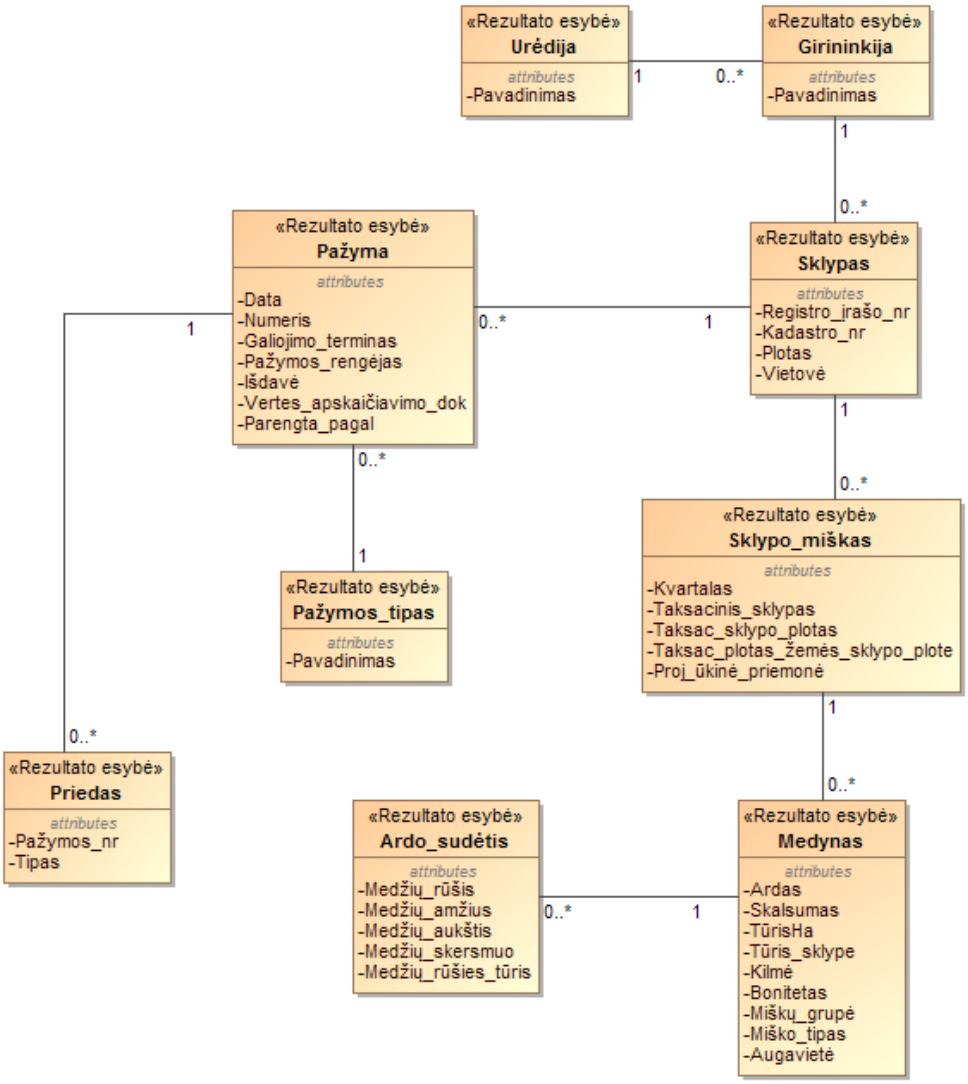
[14] C. V. Geambasu, „BPMN VS. UML activity diagram for business process modelling," *Accounting and Management Information Systems,* t. 11, pp. 637-651, 2012.

8. PRIEDAI

8.1. Priedas. Sistemos specifikacija pagal ODRES metodą

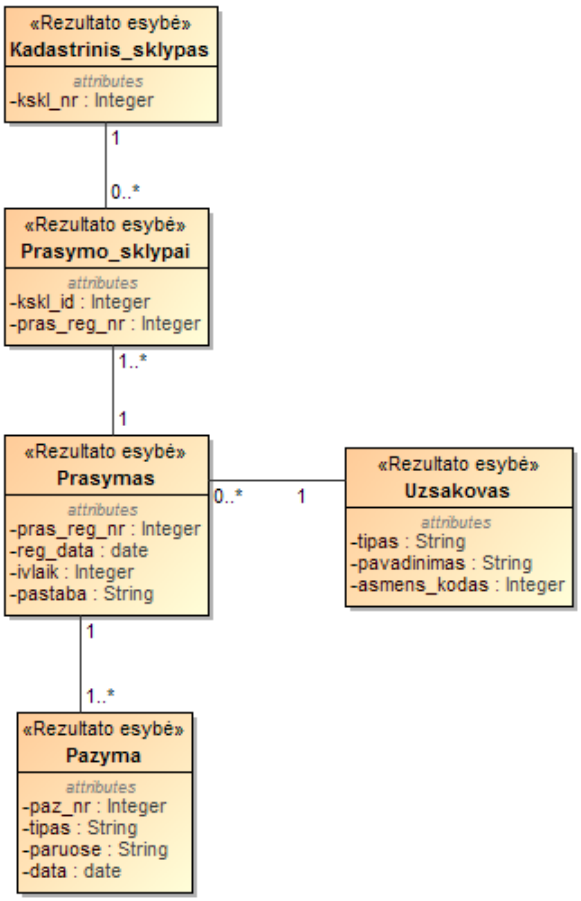
Rezultatai

Trečioji pažyma - taksaciniai rodikliai, šios pažymos struktūros specifikacija vaizduojama 8.1 paveiksle.



8.1 pav. Pažymos taksaciniai rodikliai specifikacija

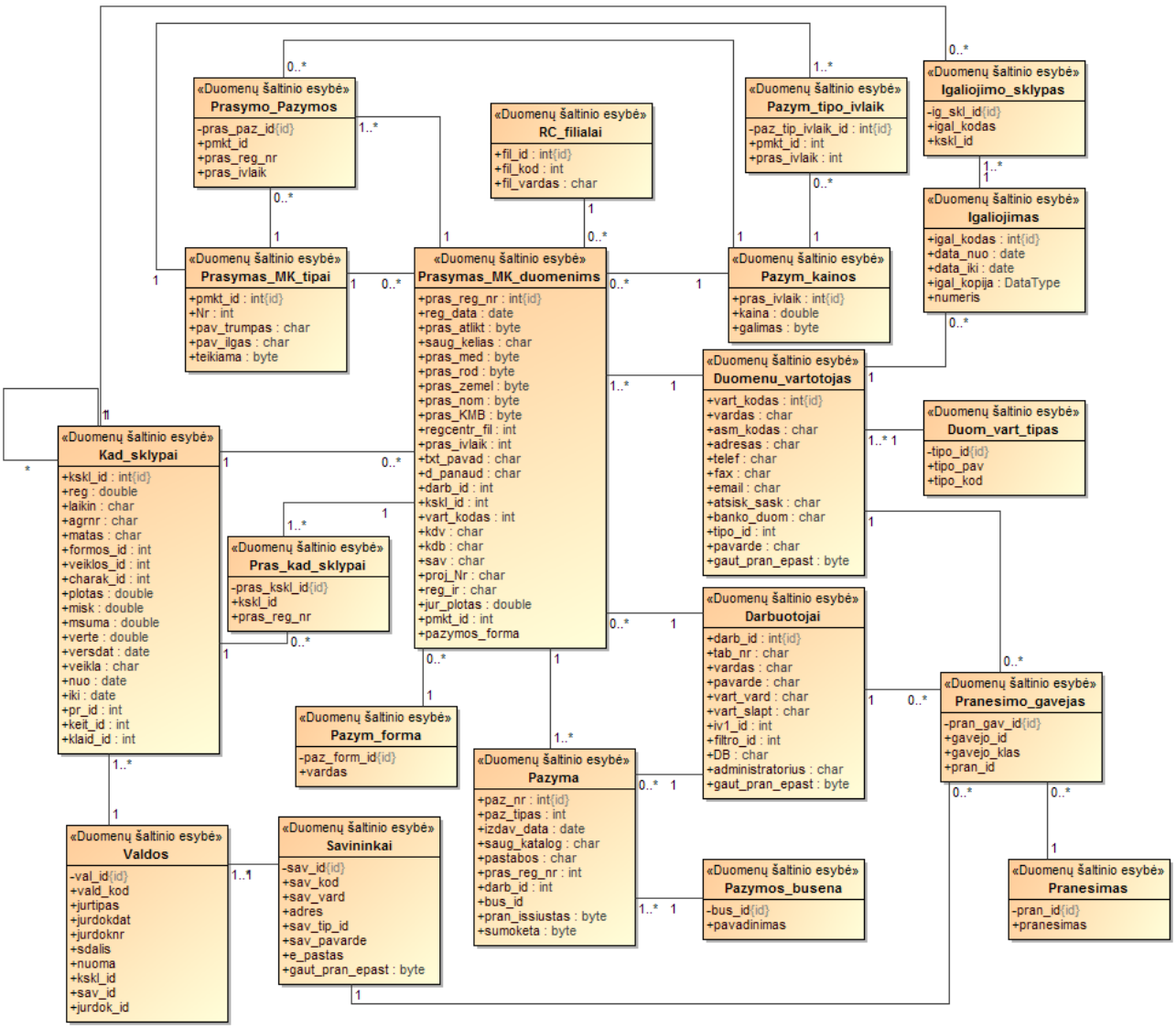
Miškų kadastro skyriaus darbuotojai nori gauti informaciją apie pažymų teikimą, taigi pagal pateiktą popierinę pažymų ataskaitą sumodeliuotas rezultatas pažymų ataskaitoms realizuoti. Šių ataskaitų specifikacija pateikiama 8.2 paveiksle.



8.2 pav. Pažymų ataskaitos specifikacija

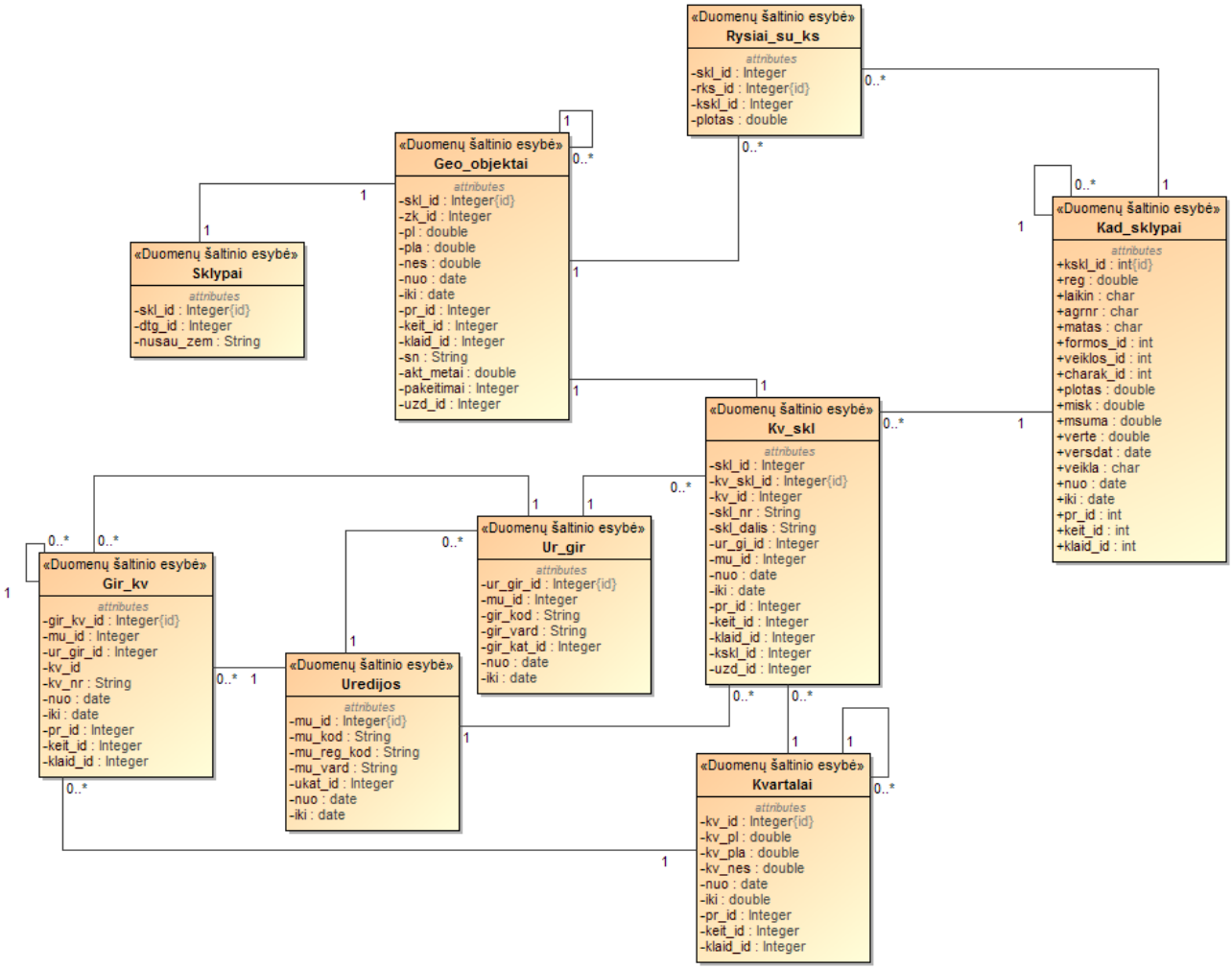
Duomenų šaltiniai

Užsakant pažymas sistemoje teikiamas prašymas su kuriuo nurodomi kadastriniai sklypai, kuriems užsakomos pažymos, pažymų tipai, kas užsako pažymas, per kiek laiko jas prašoma pagaminti ir dar keletas papildomų charakteristikų. Taip pat pažymų ataskaitoje pateikiama informacija apie prašymus, jų pažymas ir susijusi informacija. Todėl trečiojo duomenų šaltinio specifikacija sudaryta remiantis duomenų bazės lentele „Prašymas_MK_duomenims“ ir su šia lentele susijusiomis duomenų lentelėmis. Prasymas_MK_duomenims specifikacija pateikiama 8.3 paveiksle.



8.3 pav. Prasymas_MK_duomenims duomenų šaltinio specifikacija

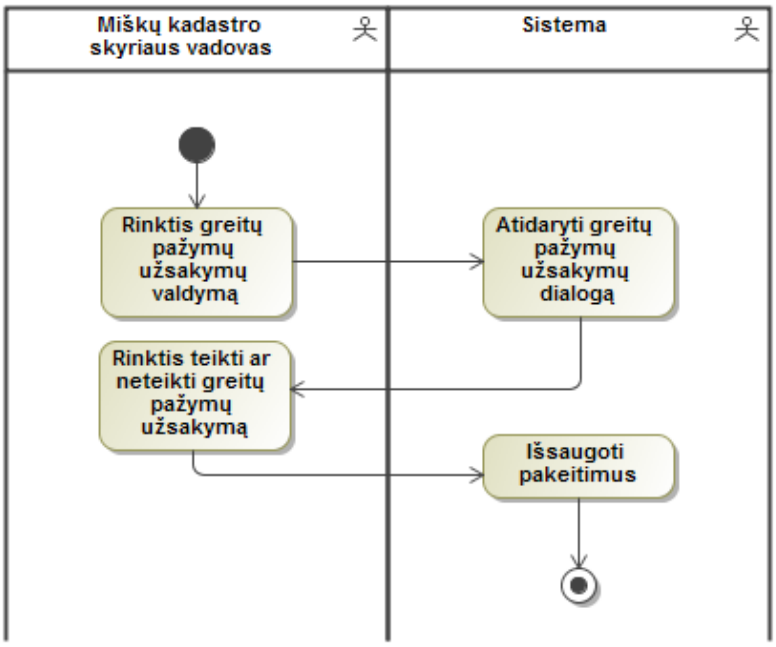
Ketvirtas duomenų šaltinis skirtas duomenims apie geografinius objektus: sklypus, kadastrinius sklypus, urėdijas, girininkijas, kvartalus, ir ryšiams tarp jų realizuoti. Ketvirtasis duomenų šaltinis pateikiamas 8.4 paveiksle.



8.4 pav. Geografinių objektų duomenų šaltinių specifikacija

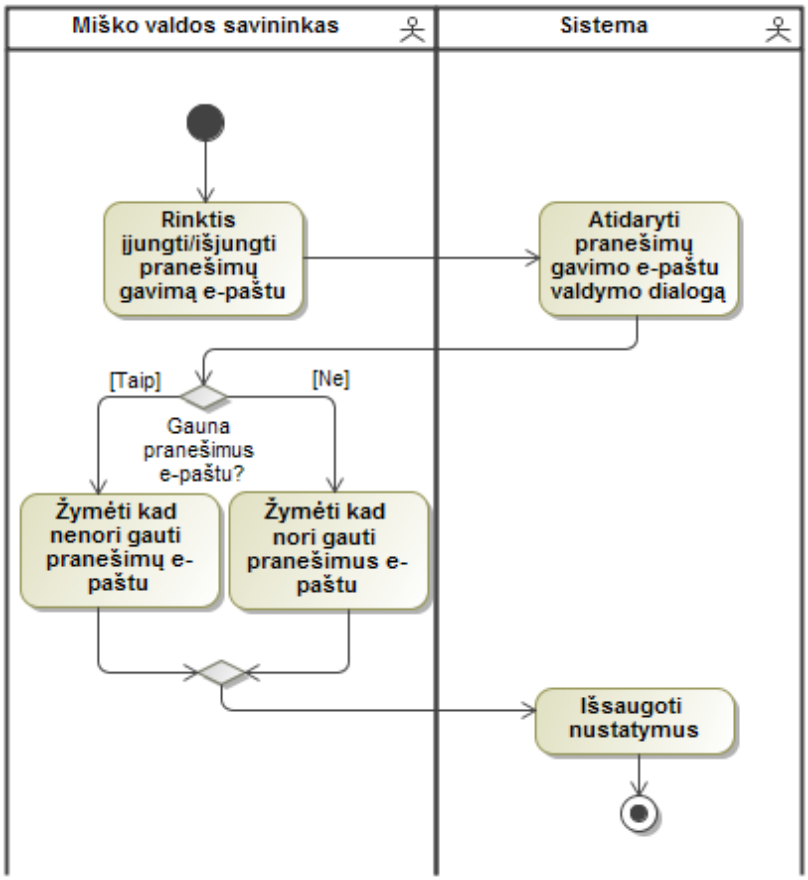
Duomenų šaltinių apdorojimo etapai

Funkcijos „Įjungti/Išjungti greitų pažymų užsakymą“ veiklos diagrama pateikiama 8.5 paveiksle. Miškų kadastro skyriaus vadovas renkasi greitų pažymų užsakymo valdymą. Sistema atidaro greitų pažymų užsakymo valdymo dialogą. Asmuo pasirenka teikti ar neteikti greitas pažymas ir sistema išsaugo pakeitimus.



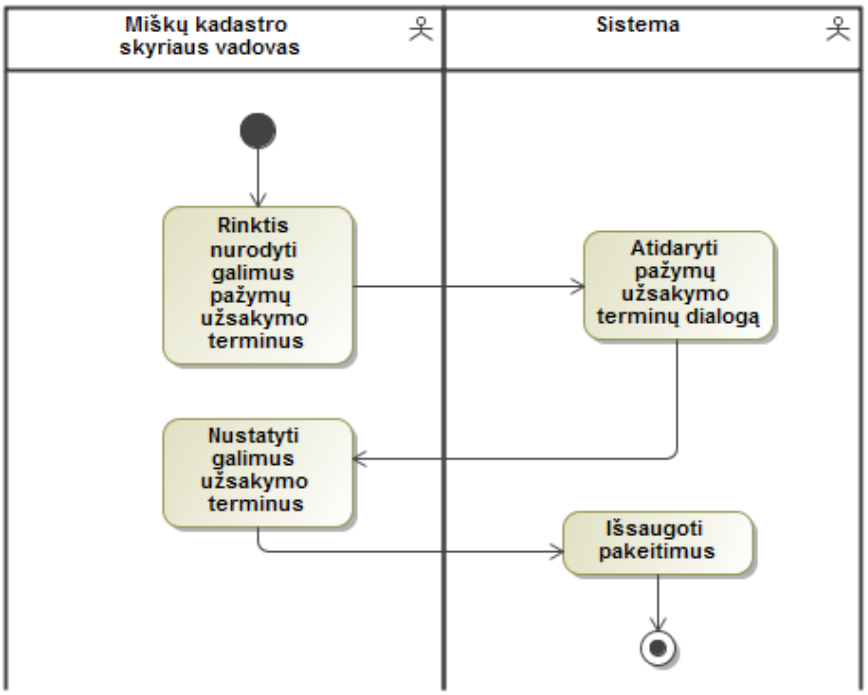
8.5 pav. Funkcijos „Įjungti/Išjungti greitų pažymų užsakymą“ veiklos diagrama

Funkcijos „Įjungti/Išjungti pranešimų gavimą e-paštu“ veiklos diagrama pateikiama 8.6 paveiksle. Miško valdos savininkas renkasi vykdyti funkciją „Įjungti/Išjungti pranešimų gavimą e-paštu“. Sistema atidaro pranešimų gavimo e-paštu valdymo dialogą. Tada asmuo nustato ar nori gauti, ar nenori gauti pranešimų e-paštu, o sistema išsaugo nustatymus.



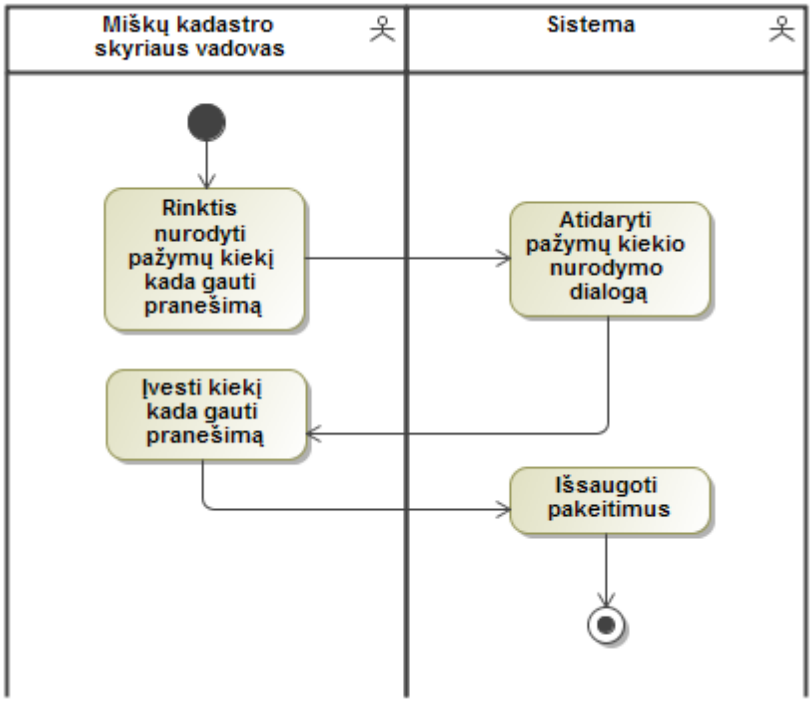
8.6 pav. Funkcijos“ Įjungti/Išjungti pranešimų gavimą e-paštu“ veiklos diagrama

Funkcijos „Nurodyti galimus pažymų užsakymo terminus“ veiklos diagrama pateikiama 8.7 paveiksle. Miškų kadastro skyriaus vadovas renkasi Nurodyti galimus pažymų užsakymo terminus, sistema atidaro dialogą jo pasirinktai funkcijai. Asmuo nurodo prie pažymų tipų kokius gavimo terminus ši gali turėti ir sistema išsaugo pakeitimus.



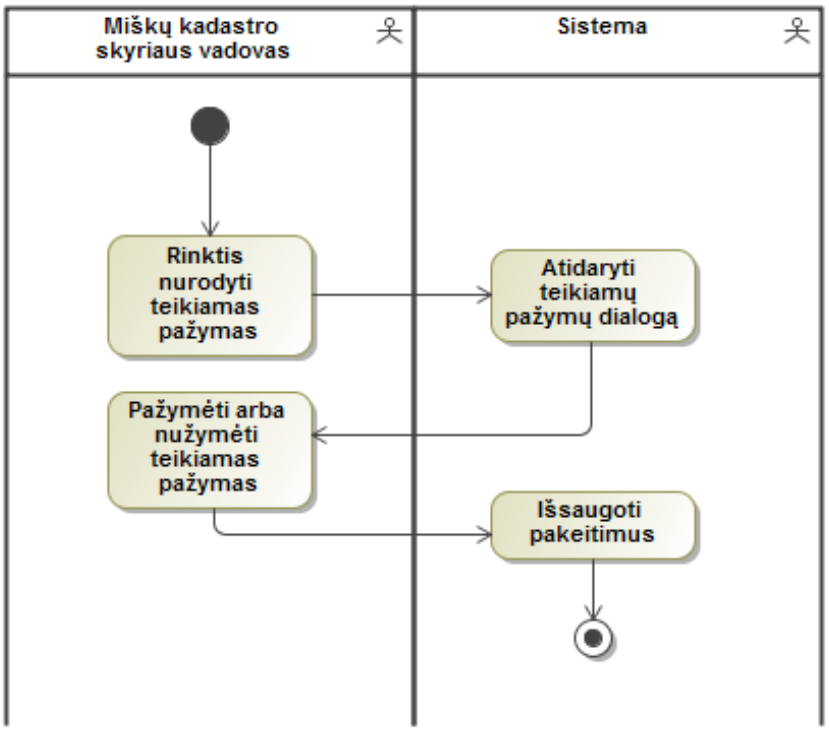
8.7 pav. Funkcijos „Nurodyti galimus pažymų užsakymo terminus“ veiklos diagrama

Funkcijos „Nurodyti pažymų kiekį kada gauti pranešimą“ veiklos diagrama pateikiama 8.8 paveiksle. Miškų kadastro skyriaus vadovas renkasi funkciją „Nurodyti pažymų kiekį kada gauti pranešimą“, sistema atidaro funkcijos dialogą, asmuo įveda pažymų skaičių ir sistema išsaugo pakeitimą.



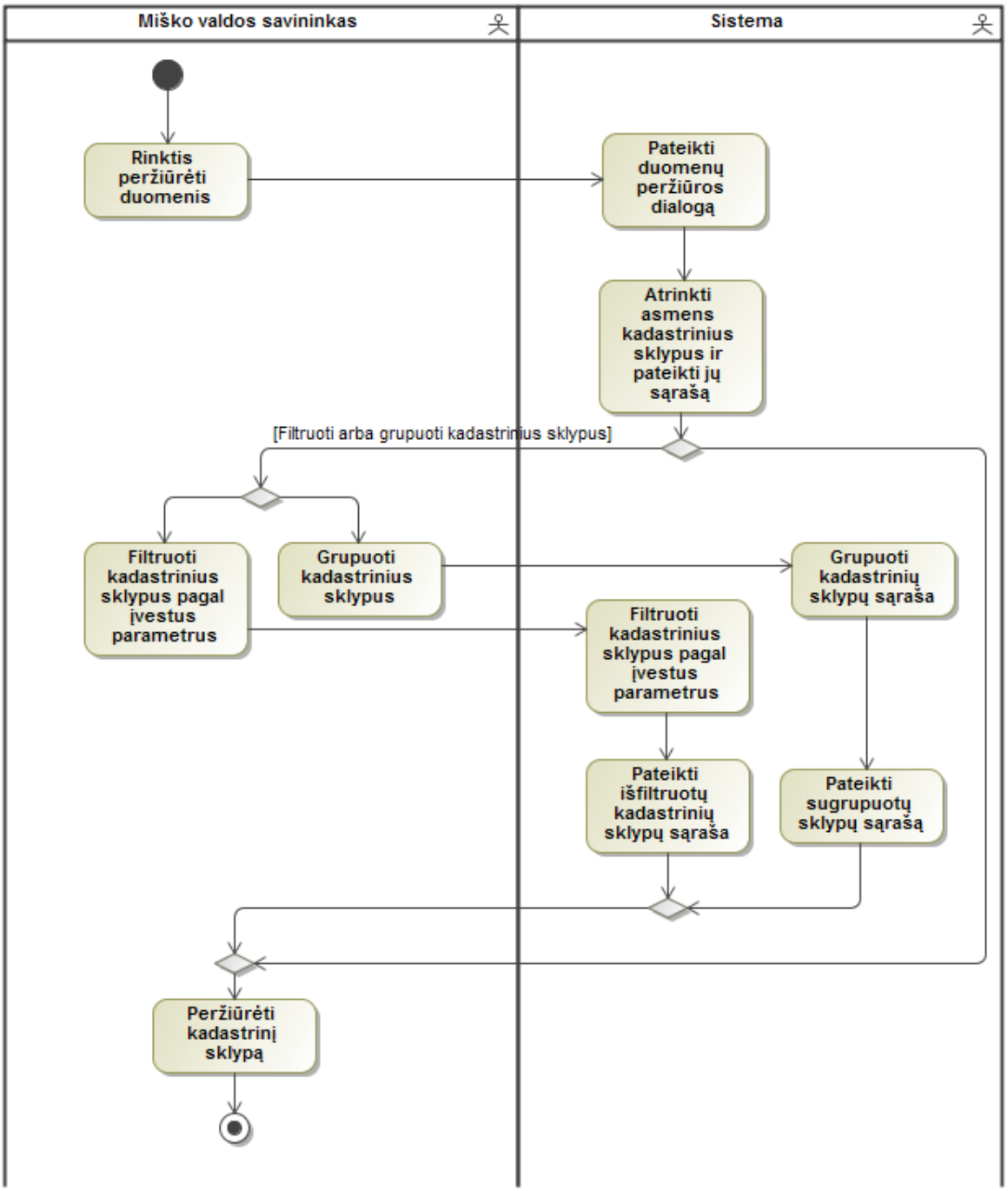
8.8 pav. Funkcijos „Nurodyti pažymų kiekį kada gauti pranešimą“ veiklos diagrama.

Funkcijos „Nurodyti teikiamas pažymas“ veiklos diagrama pateikiama 8.9 paveiksle. Miškų kadastro skyriaus vadovas renkasi nurodyti teikiamas pažymas. Sistema atidaro teikiamų pažymų dialogą, kuriame asmuo pažymi arba nužymi teikiamas pažymas, visi pakeitimai sistemos išsaugomi.



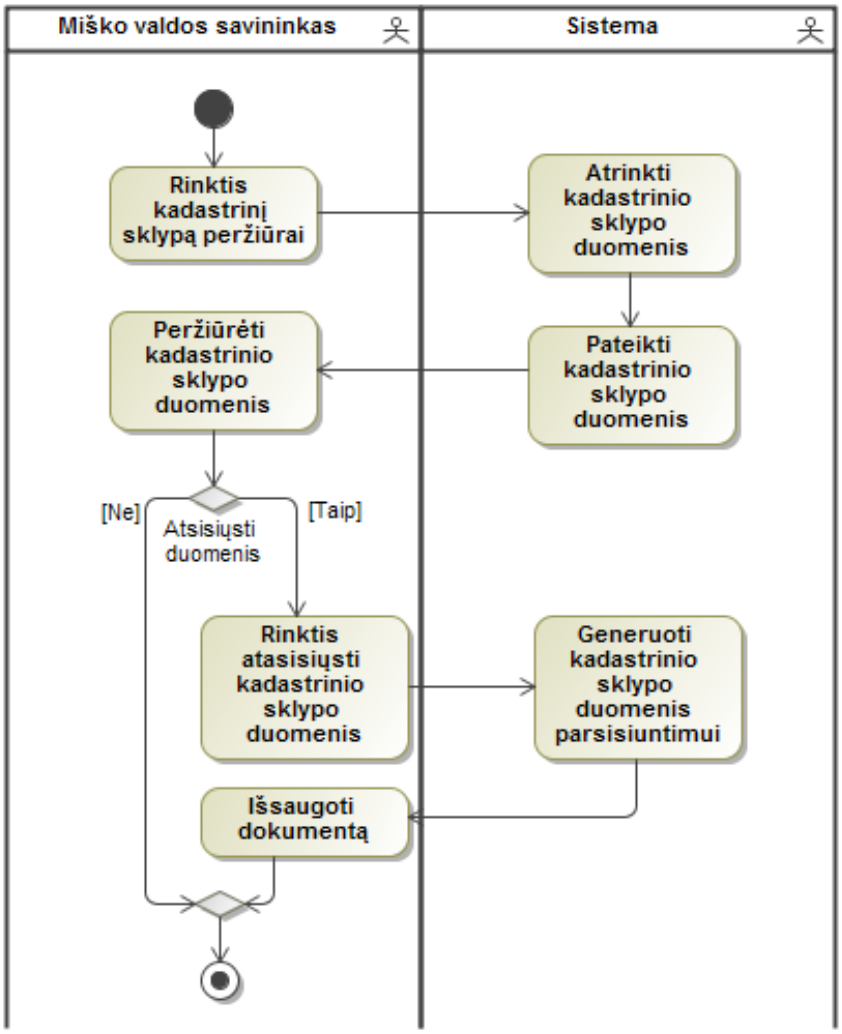
8.9 pav. Funkcijos „Nurodyti teikiamas pažymas“ veiklos diagrama

Funkcijos „Peržiūrėti duomenis“ veiklos diagrama pateikiama 8.10 paveiksle. Miško valdos savininkas renkasi peržiūrėti duomenis. Sistema atidaro duomenų peržiūros dialogą, kuriame pateikia asmeniui priklausančių kadastrinių sklypų sąrašą. Jeigu asmuo nori gali filtruoti arba grupuoti savo kadastrinių sklypų sąraša. Filtruojant asmuo įveda filtravimo parametrus pagal kuriuos pateikiamas kadatrinių sklypų priklausančių asmeniui sąrašas. Pasirinkus grupuoti kadastriniai sklypai sugrupuojami pagal sistemoje nustatytus parametrus ir pateikiamas sugrupuotas sąrašas. Arba iš filtruoto, arba iš sugrupuoto, arba iš pradžioje pateikto bendro sklypų sąrašo asmuo pasirenka kadastrinį sklypą ir gali peržiūrėti informaciją ir duomenis apie jį.



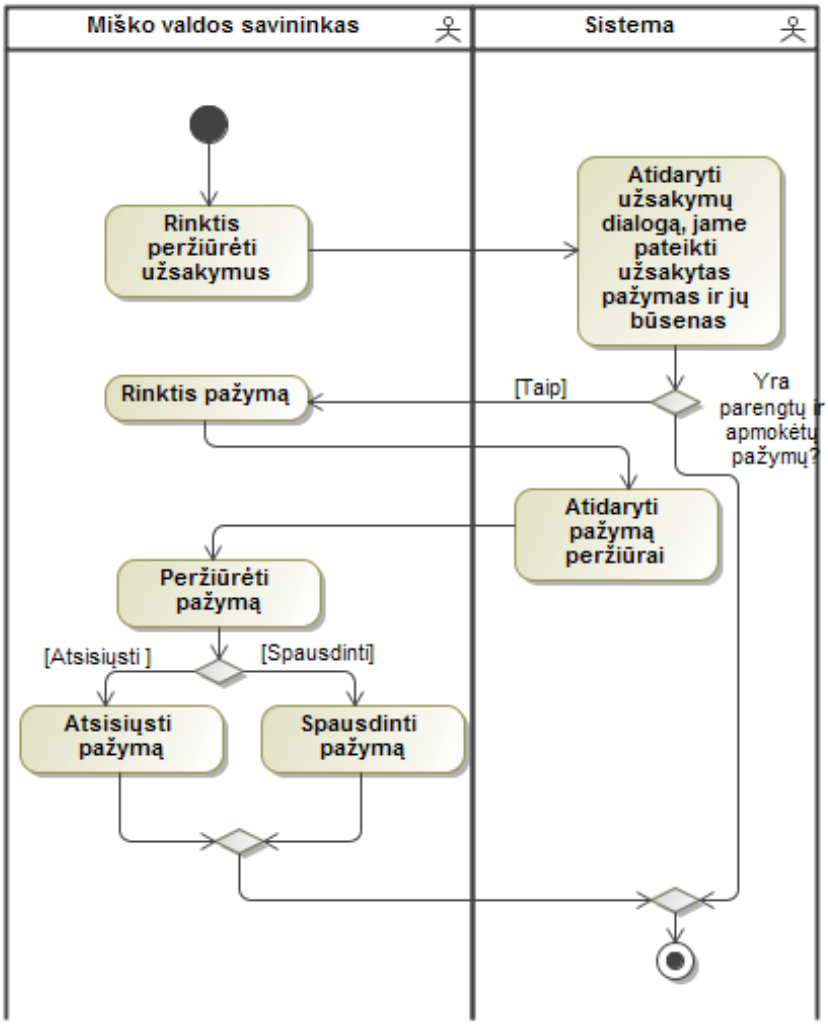
8.10 pav. Funkcijos „Peržiūrėti duomenis“ veiklos diagrama

Funkcijos „Peržiūrėti kadastrinį sklypą“ veiklos diagrama pateikiama 8.11 paveiksle. Miško valdos savininkas pasirenka peržiūrėti kadastrinio sklypo duomenis. Sistema atrenka ir pateikia turimus duomenis apie kadastrinį sklypą. Asmuo peržiūri duomenis ir gali nuspręsti ar nori parsisiųsti kadastrinio sklypo duomenis. Jeigu nori asmuo renkasi atsisiųsti duomenis, sistema sugeneruoja dokumentą su duomenimis, o asmuo tada gali sugeneruotą dokumentą parsisiųsti.



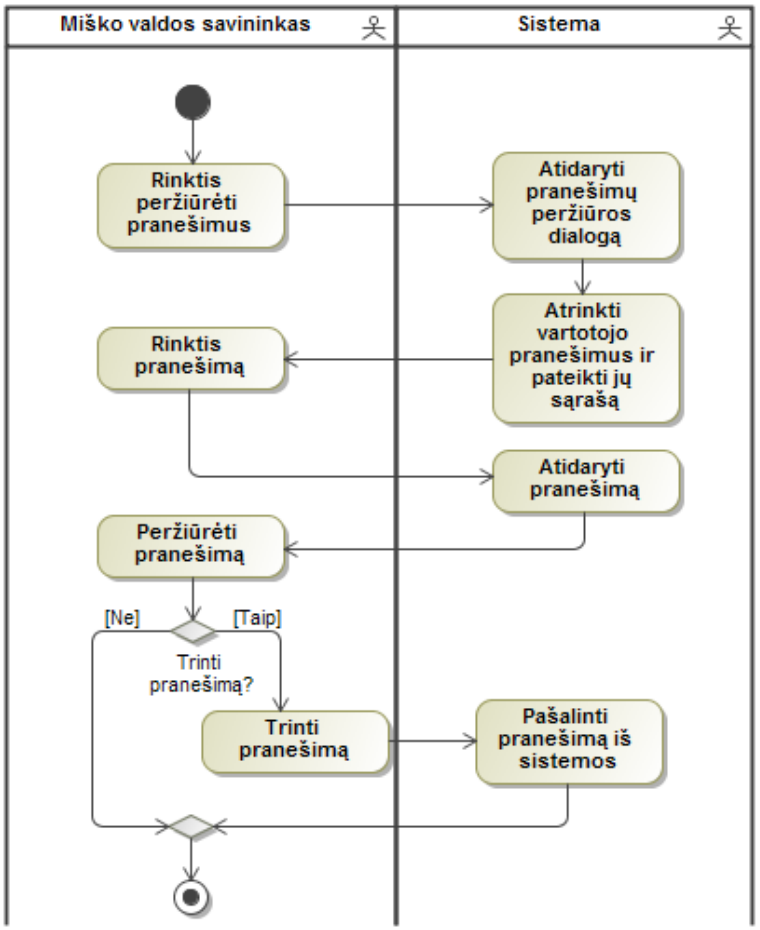
8.11 pav. Funkcijos „Peržiūrėti kadastrinį sklypą“ veiklos diagrama

Funkcijos „Peržiūrėti pažymą“ veiklos diagrama pateikiama 8.12 paveiksle. Miško valdos savininkas renkasi peržiūrėti užsakymus, sistema atidaro užsakymų peržiūros dialogą, kuriame pateikia asmens užsakytas pažymas ir jų būsenas. Jeigu tarp asmens užsakytų ir parengtų pažymų yra parengtų ir apmokėtų, miško valdos savininkas gali rinktis peržiūrėti pažymą. Sistema atidaro pažymą peržiūrai. Asmuo peržiūrėjęs pažymą gali ją atsisiųsti arba atsispausdinti.



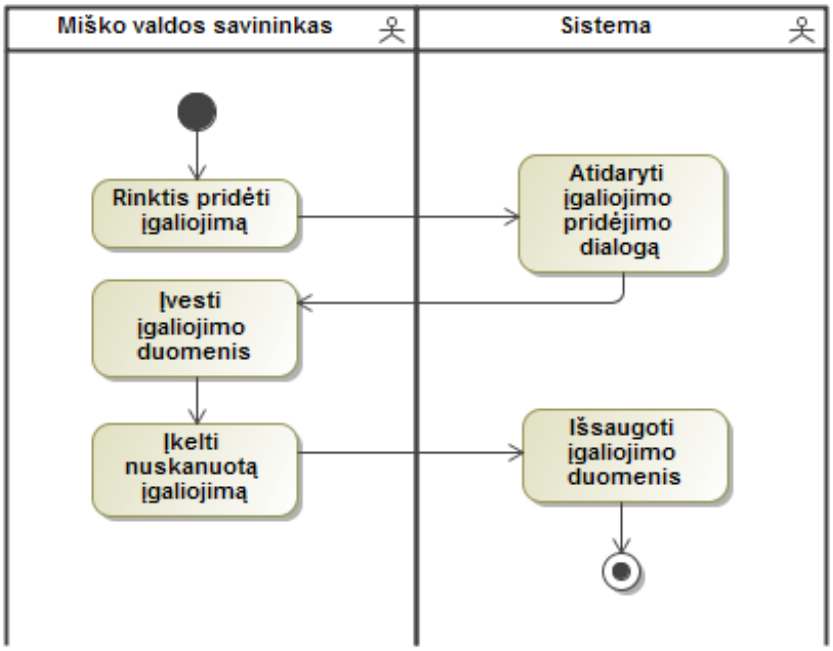
8.12 pav. Funkcijos „Peržiūrėti pažymą“ veiklos diagrama

Funkcijos „Peržiūrėti pranešimus“ veiklos diagrama pateikiama 8.13 paveiksle. Miško valdos savininkas renkasi peržiūrėti pranešimus. Sistema atidaro pranešimų peržiūros dialogą, kuriame pateikia asmens gautus pranešimus. Asmuo pasirenka pranešimą iš sąrašo, o sistema atidaro pasirinktą pranešimą peržiūrai. Miško valdos savininkas peržiūri pranešimą ir gali rinktis trinti pranešimą. Jeigu pasirenka trinti, sistema pašalina pranešimą.



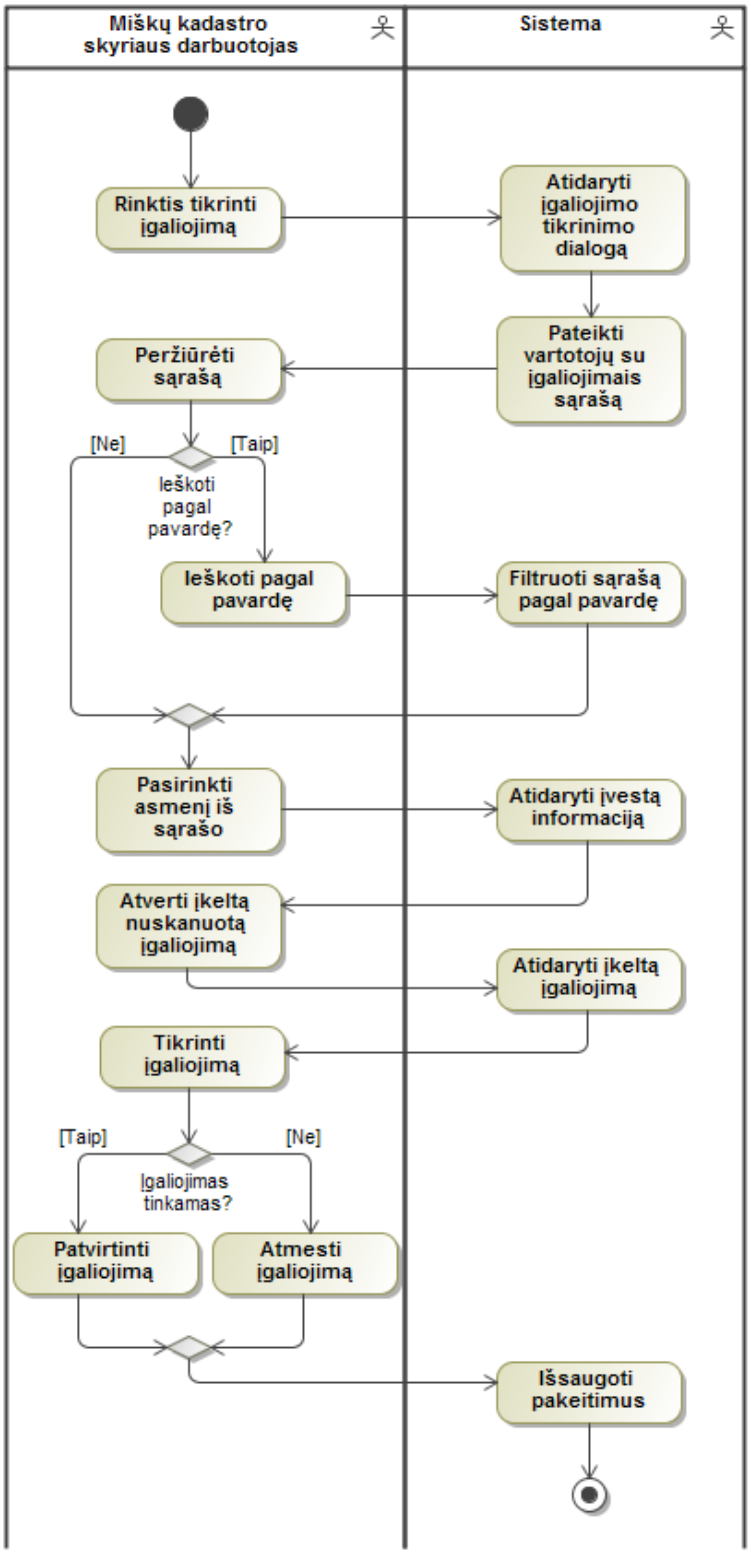
8.13 pav. Funkcijos „Peržiūrėti pranešimus“ veiklos diagrama

Funkcijos „Pridėti įgaliojimą“ veiklos diagrama pateikiama 8.14 paveiksle. Miško valdos savininkas renkasi pridėti įgaliojimą. Sistema atidaro įgaliojimo pridėjimo dialogą, kuriame asmuo įveda reikiamus įgaliojimo duomenis ir tuo pačiu įkelia nuskanuotą įgaliojimo kopiją į sistemą. Sistema išsaugo duomenis ir nuskanuotą įgaliojimą.



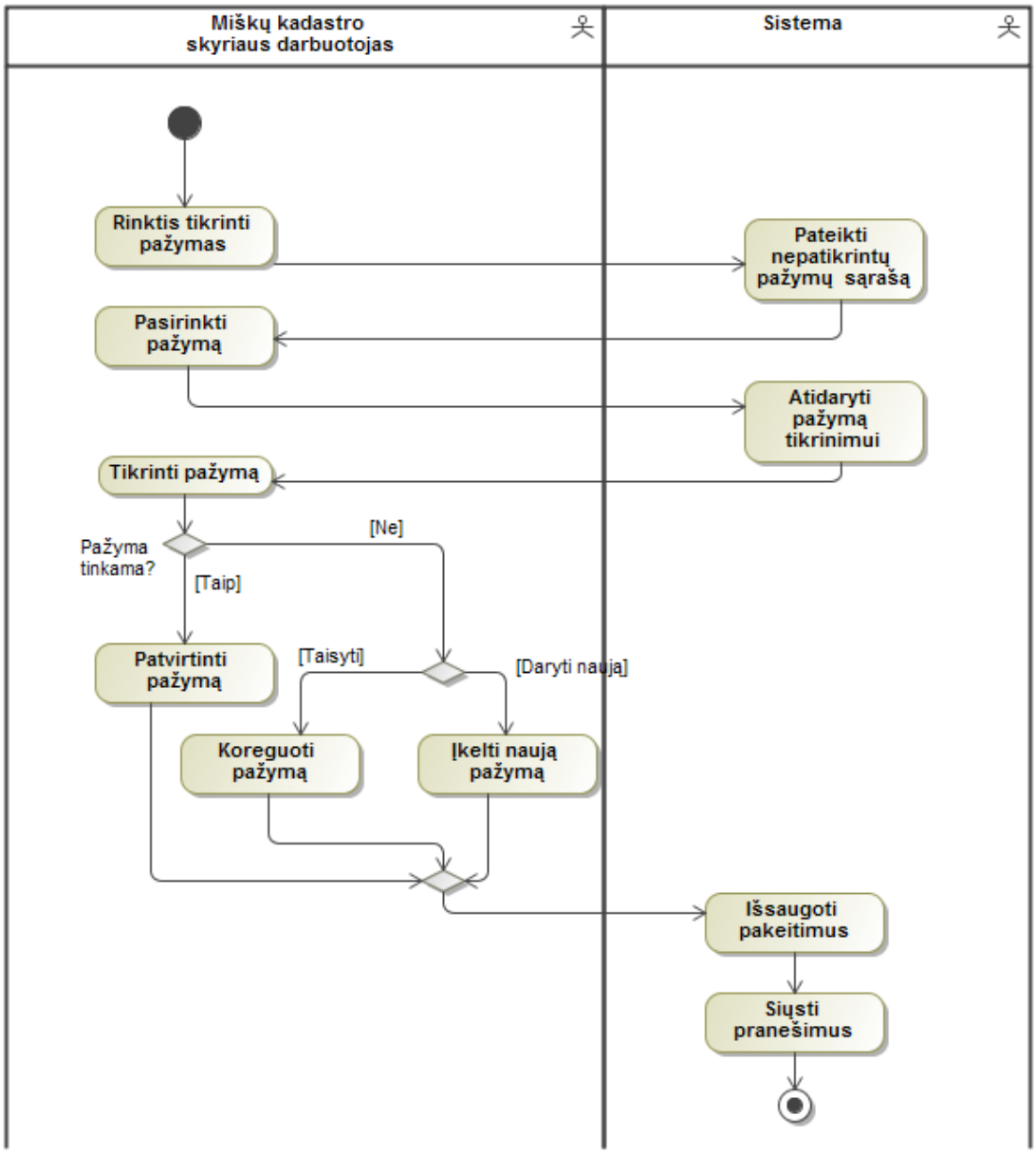
8.14 pav. Funkcios „Pridėti įgaliojimą“ veiklos diagrama

Funkcijos „Tikrinti įgaliojimą“ veiklos diagrama pateikiama 8.15 paveiksle. Miškų kadastro skyriaus darbuotojas renkasi tikrinti įgaliojimą, o sistema atidaro įgaliojimų tikrinimo dialogą, kuriame pateikia vartotojų su įgaliojimais sąrašą. Asmuo gali ieškoti vartotojo pagal pavardę ir sistema pateiks išfiltruotą sąraša, arba tiesiog rinktis vartotoją iš pateikto sąrašo. Pasirinkus vartotoją atidaroma informaciją apie jo įgaliojimą. Tada darbuotojas turi atsidaryti įkeltą nuskanuotą įgaliojimą ir patikrinti ar įgaliojimas tinkamas ir duomenys tvarkingi. Jeigu viskas gerai įgaliojimą patvirtina, priešingu atveju atmeta. Galiausiai sistema išsaugo atliktus pakeitimus.



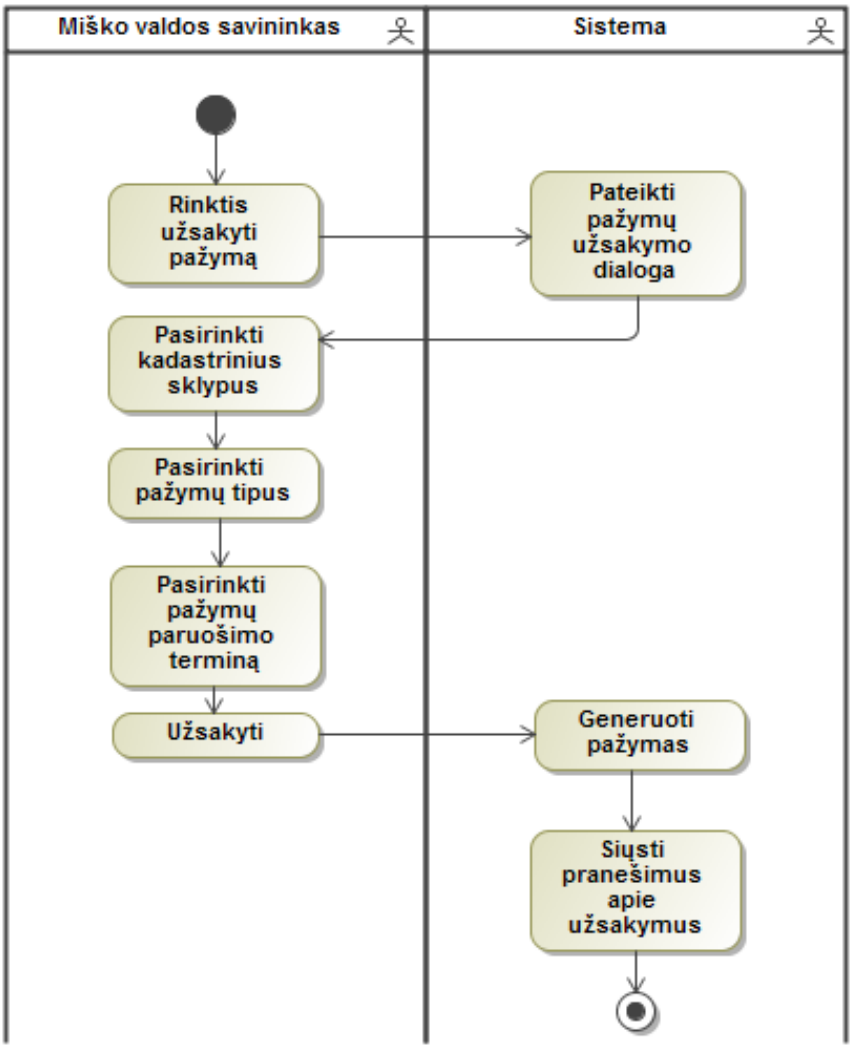
8.15 pav. Funkcijos „Tikrinti įgaliojimą“ veiklos diagrama

Funkcijos „Tikrinti pažymas“ veiklos diagrama pateikiama 8.16 paveiksle. Miškų kadastro skyriaus darbuotojas renkasi tikrinti pažymas. Sistema atidaro pažymų tikrinimo dialogą, kuriame pateikia nepatikrintų pažymų sąrašą. Asmuo pasirenka pažymą, o sistema ją atveria patikrinimui. Jeigu pažyma tinkama darbuotojas patvirtina pažymą, priešingu atveju, jeigu įmanoma pataiso pažymą, jeigu ji netinkama, įkelia naują paties padarytą pažymą į sistemą vietoje netinkamos. Galiausiai sistema išsaugo duomenų pakeitimus ir išsiunčia pranešimą užsakovui apie patvirtintą ir padarytą pažymą.



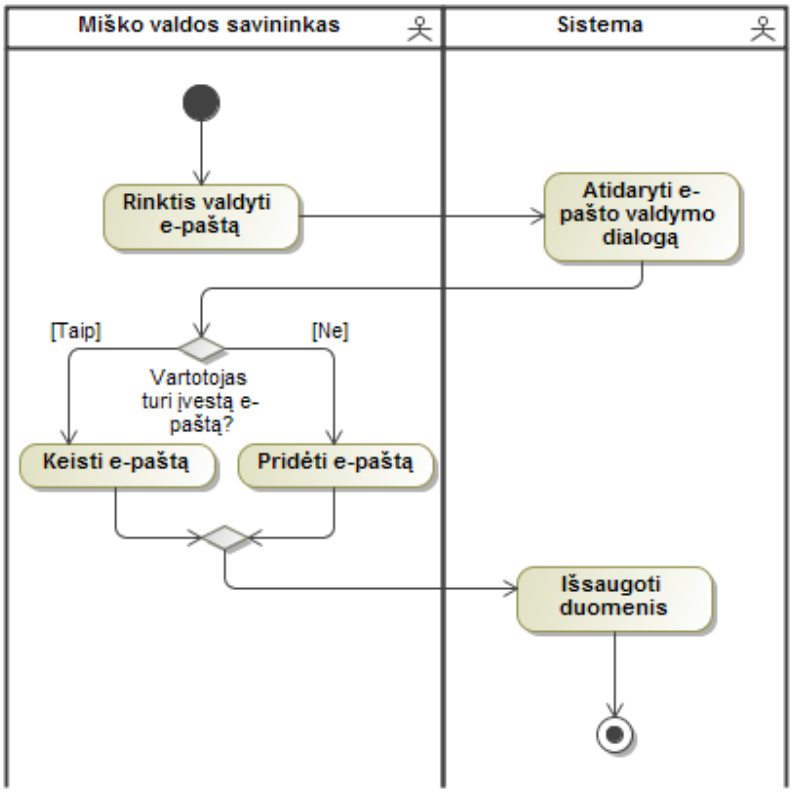
8.16 pav. Funkcijos „Tikrinti pažymas“ veiklos diagrama

Funkcijos „Užsakyti pažymą“ veiklos diagrama pateikiama 8.17 paveiksle. Miško valdos savininkas renkasi užsakyti pažymą. Sistema atidaro pažymos užsakymo dialogą. Asmuo pasirenka kadastrinius sklypus, kuriems nori užsakyti pažymą, pasirenka pažymų tipus, kokias pažymas nori užsakyti, tada pasirenka pažymų paruošimo laiką, per kiek nori, kad pažymos jam būtų paruoštos, ir užsako. Užsakius sistema automatiškai generuoja pažymas pagal duomenų bazėje turimus duomenis, ir išsiunčia pranešimą asmeniui, kad pažyma arba pažymos sėkmingai užsakytos.



8.17 pav. Funkcijos „Užsakyti pažymą“ veiklos diagrama

Funkcijos „Valdyti e-paštą“ veiklos diagrama pateikiama 8.18 paveiksle. Miško valdos savininkas renkasi e-pašto valdymą ir sistema atidaro e-pašto valdymo dialogą. Jeigu asmuo jau turi įvestą elektroninį paštą sistemoje, jis gali pasikeisti savo e-pašto adresą, jeigu dar neturi, gali įvesti savo e-paštą į sistemą. Galiausiai sistema išsaugo visus įvestus duomenis.

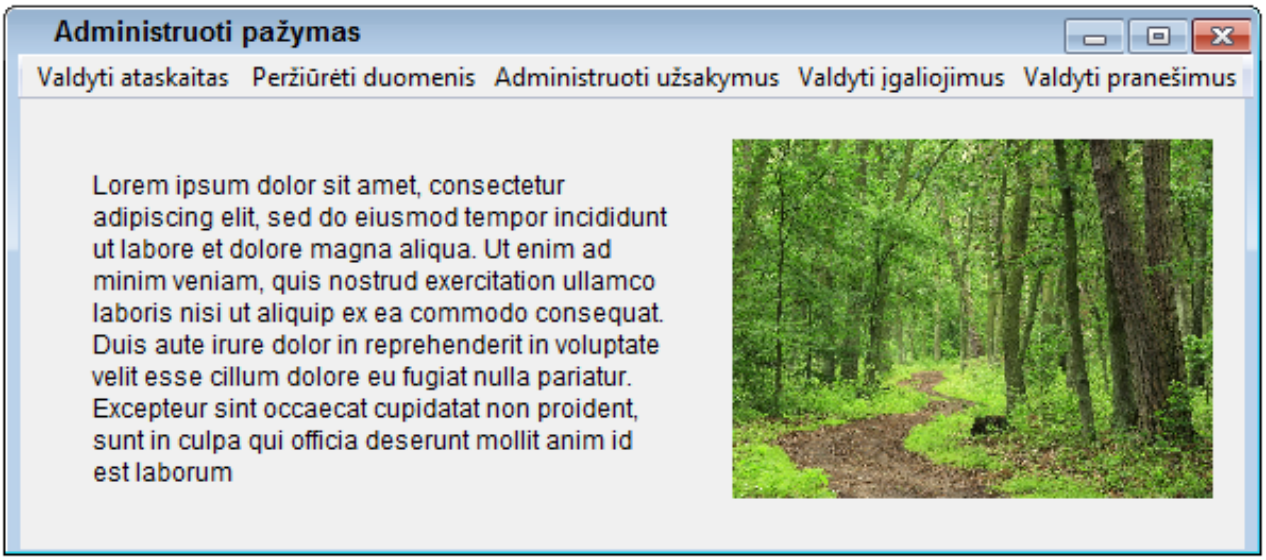


8.18 pav. Funkcijos „Valdyti e-paštą“ veiklos diagrama

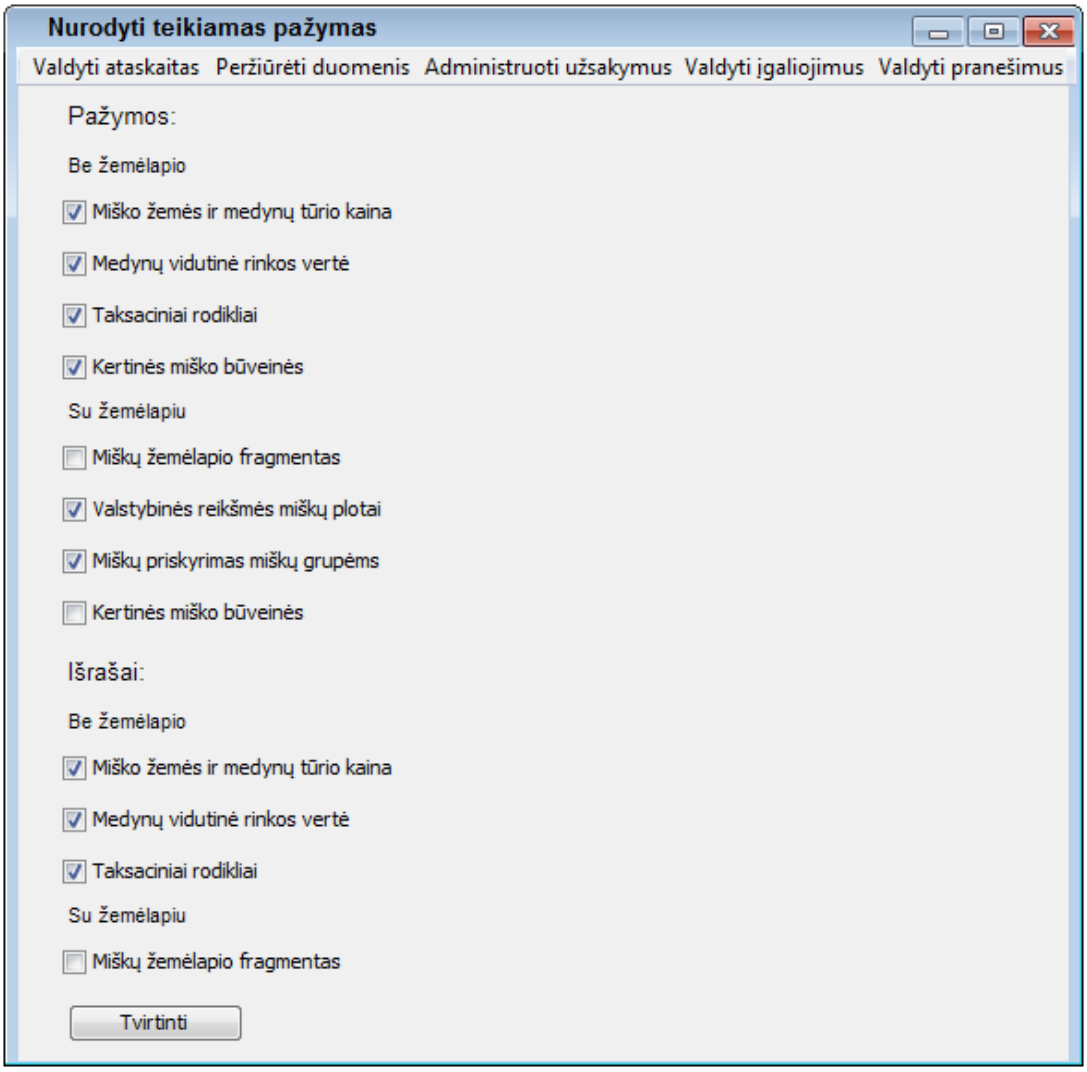
Naudotojo sąsajos prototipai

Toliau pateikiami suprojektuoti naudotojų sąsajos prototipai:

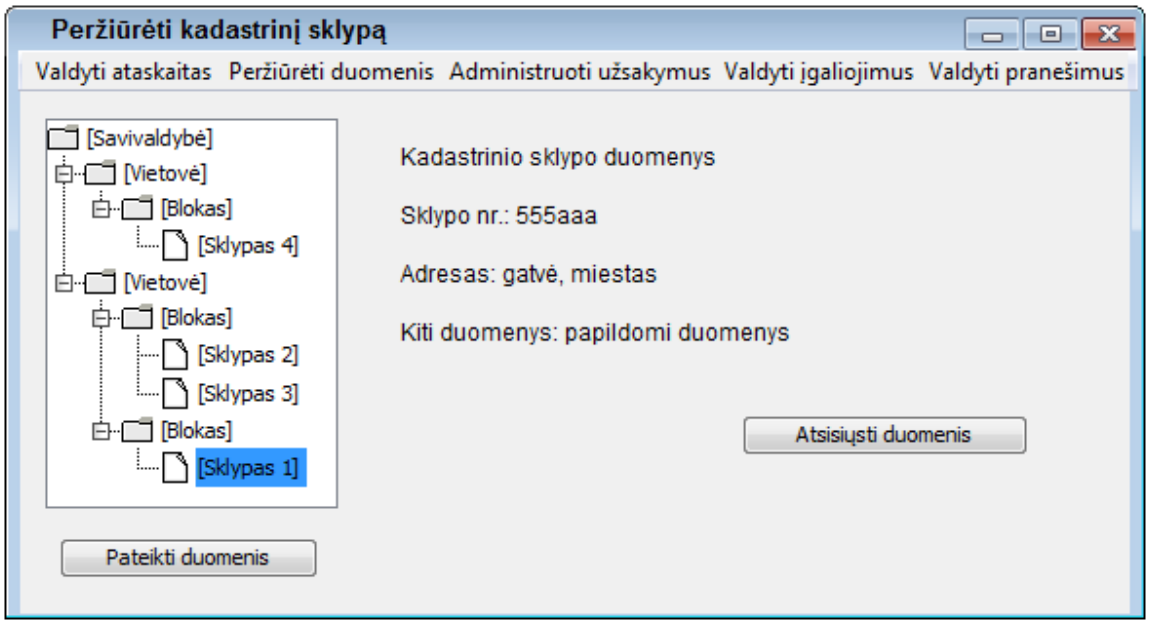
- Pradinio pažymų administravimo posistemio naudotojo sąsajos prototipas pateikiamas 8.19 paveiksle.
- Teikiamų pažymų nurodymo naudotojo sąsajos prototipas pateikiamas 8.20 paveiksle.
- Kadastrinio sklypo duomenų peržiūros naudotojo sąsajos prototipas pateikiamas 8.21 paveiksle.



8.19 pav. Pažymų administravimo pradinio sistemos naudotojo lango prototipas



8.20 pav. Teikiamų pažymų nurodymo naudotojo sąsajos prototipas



8.21 pav. Kadastrinio sklypo duomenų peržiūros naudotojo sąsajos prototipas

8.2. Priedas. Ataskaitos generavimo šablonai

8.2.1. Sistemos reikalavimų specifikavimo ataskaitos generavimo šablonas

```
#macro( getReportName $elem $name)#if(
$report.containsStereotype($elem, "documented") )#set( $name =
$report.getStereotypeProperty($elem, "documented",
"reportName").get(0) )#else#set( $name = $elem.getName() )#end#end
#macro( printDiagram $diag )#set( $diagName = ""
)#getReportName($diag, $diagName)
     $diag.documentation

                                   $diag.image
                          1.1      pav. $diag.name

#end
#macro( resolveDiag $diagQName $diag )#foreach( $testDiag in
$Diagram )#if( $report.getPackageQualifiedName($testDiag, ".") ==
$diagQName )#set( $diag = $testDiag )#end#end#end
#macro( getDiagramUseCases $diag $useCases )#set( $useCases =
$sorter.sort($report.filterElement($report.getDiagramElements($diag
), ["UseCase"]), "name") )#end
#macro( printUcDiagram $diag )#printDiagram($diag)#set( $diagUcs =
"" )#getDiagramUseCases($diag, $diagUcs)#foreach( $diagUc in
$diagUcs )#printUseCase($diagUc)#end#end
#macro( printActor $uc ) #set( $actors = $array.createHashSet()
)#set( $actors = $report.getRelativeActor($uc))#foreach( $act in
$actors )$act.getName()
#end#end
#macro( printUseCase $uc )
```

1.1 lentelė. *Panaudojimo atvejis "$uc.name"*

PA pavadinimas. *$uc.name*	
Aprašymas. *$uc.documentation*	
Aktoriu s.	*#printActor($uc)*

```
#end
#macro( getDiagramClasses $diag $diagClss )#set( $diagClss =
$sorter.sort($report.filterElement($report.getDiagramElements($diag
), ["Class", "Data Type", "Enumeration"]), "name") )#end

#macro( printStdClass $cls )#set( $baseClss =
$sorter.sort($cls.getGeneral(), "name") )
```

1.1 lentelė. Klasė "$cls.name"

Klasės pavadinimas. *$cls.name.*
Aprašymas. *$cls.documentation*

```
#end

#macro( printEnumClass $cls )#set( $enumLits =
$sorter.sort($cls.getOwnedLiteral(), "name") )
```

1.1 lentelė. *Reikšmių klasė "$cls.name"*

Klasės pavadinimas. *$cls.name.*

Aprašymas. *$cls.documentation*
Reikšmės
#if($enumLits.size() > 0)#set($cnt = 0)#foreach($enumLit in $enumLits)#set($cnt = $cnt + 1) **$cnt.** *$enumLit.name* **Aprašymas.** *$enumLit.documentation* *#end#else#**#Nėra#end*

```
#end
#macro( printClass $cls )#if(
$cls.getClass().getName().endsWith(".EnumerationImpl")
)#printEnumClass($cls)#else#printStdClass($cls)#end#end

      #macro(  printClsDiagram  $diag  )#printDiagram($diag)#set(
$diagClss  =  ""  )#getDiagramClasses($diag,  $diagClss)#foreach(
$diagCls in $diagClss )#printClass($diagCls)#end#end

#set( $basePkgName = "Bendras pradinis kelias" )

#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos " )
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printUcDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printDiagram($diag)

#set( $basePkgName = "Sistemos reikalavimų specifikacija" )
```

1. Funkcijų hierarchija

```
#set( $pkg = $basePkgName+".kelias iki diagramos " )
#set( $diag = $pkg+".funkcijų hierarchijos diagrama" )
#resolveDiag($diag, $diag)
#printUcDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".funkcijų priklausomumo aktoriams diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

2. Rezultatai

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Rezultato diagrama" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)
```

3. Duomenų šaltiniai

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Duomenų šaltinio diagrama" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)
```

4. Duomenų srautai

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Duomenų srautų diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

5. Duomenų šaltinių apdorojimo etapai

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Duomenų apdorojimo etapų veiklos diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

6. Būsenų kaita

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Būsenų kaitos diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

8.2.2. Sistemos projektavimo ataskaitos generavimo šablonas

```
#macro( getReportName $elem $name)#if(
$report.containsStereotype($elem, "documented") )#set( $name =
$report.getStereotypeProperty($elem, "documented",
"reportName").get(0) )#else#set( $name = $elem.getName() )#end#end
#macro( printDiagram $diag )#set( $diagName = ""
)#getReportName($diag, $diagName)
     $diag.documentation

                                $diag.image
                1.2       pav. $diag.name

#end
#macro( resolveDiag $diagQName $diag )#foreach( $testDiag in
$Diagram )#if( $report.getPackageQualifiedName($testDiag, ".") ==
$diagQName )#set( $diag = $testDiag )#end#end#end
#macro( getDiagramClasses $diag $diagClss )#set( $diagClss =
$sorter.sort($report.filterElement($report.getDiagramElements($diag
), ["Class", "Data Type", "Enumeration"]), "name") )#end
#macro( printStdClass $cls )#set( $baseClss =
$sorter.sort($cls.getGeneral(), "name") )
```

1.2 lentelė. Klasė "$cls.name"

Klasės pavadinimas. *$cls.name.*
Aprašymas. *$cls.documentation*

```
#end
```

```
#macro( printEnumClass $cls )#set( $enumLits =
$sorter.sort($cls.getOwnedLiteral(), "name") )
```

1.2 lentelė. *Reikšmių klasė "$cls.name"*

Klasės pavadinimas. *$cls.name.*
Aprašymas. *$cls.documentation*
Reikšmės
#if($enumLits.size() > 0)#set($cnt = 0)#foreach($enumLit in $enumLits)#set($cnt = $cnt + 1) **$cnt.** *$enumLit.name* **Aprašymas.** *$enumLit.documentation* *#end#else#**#Nėra#end*

```
#end
#macro( printClass $cls )#if(
$cls.getClass().getName().endsWith(".EnumerationImpl")
)#printEnumClass($cls)#else#printStdClass($cls)#end#end

      #macro(  printClsDiagram  $diag  )#printDiagram($diag)#set(
$diagClss  =  ""  )#getDiagramClasses($diag,  $diagClss)#foreach(
$diagCls in $diagClss )#printClass($diagCls)#end#end

#set( $basePkgName = "Bendras pradinis kelias" )

#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printDiagram($diag)

#set( $basePkgName = "Sistemos projektavimas" )
```

1. Duomenų modelis

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Duomenų modelio diagrama" )
#resolveDiag($diag, $diag
#printClsDiagram($diag)
```

2. Sistemos meniu struktūra

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Sistemos meniu struktūros diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

3. Naudotojo sąsajos prototipai

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Būsenų kaitos diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

8.2.3. Bendras ataskaitos generavimo šablonas

```
#macro( getReportName $elem $name)#if(
$report.containsStereotype($elem, "documented") )#set( $name =
$report.getStereotypeProperty($elem, "documented",
"reportName").get(0) )#else#set( $name = $elem.getName() )#end#end
#macro( printDiagram $diag )#set( $diagName = ""
)#getReportName($diag, $diagName)
     $diag.documentation

                                   $diag.image
                          1.3      pav. $diag.name

#end
#macro( resolveDiag $diagQName $diag )#foreach( $testDiag in
$Diagram )#if( $report.getPackageQualifiedName($testDiag, ".") ==
$diagQName )#set( $diag = $testDiag )#end#end#end
#macro( getDiagramUseCases $diag $useCases )#set( $useCases =
$sorter.sort($report.filterElement($report.getDiagramElements($diag
), ["UseCase"]), "name") )#end
#macro( printUcDiagram $diag )#printDiagram($diag)#set( $diagUcs =
"" )#getDiagramUseCases($diag, $diagUcs)#foreach( $diagUc in
$diagUcs )#printUseCase($diagUc)#end#end
#macro( printActor $uc ) #set( $actors = $array.createHashSet()
)#set( $actors = $report.getRelativeActor($uc))#foreach( $act in
$actors )$act.getName()
#end#end
#macro( printUseCase $uc )
```

1.2 lentelė. *Panaudojimo atvejis "$uc.name"*

PA pavadinimas. *$uc.name*	
Aprašymas. *$uc.documentation*	
Aktoriu s.	*#printActor($uc)*

```
#end
#macro( getDiagramClasses $diag $diagClss )#set( $diagClss =
$sorter.sort($report.filterElement($report.getDiagramElements($diag
), ["Class", "Data Type", "Enumeration"]), "name") )#end

#macro( printStdClass $cls )#set( $baseClss =
$sorter.sort($cls.getGeneral(), "name") )
```

1.3 lentelė. Klasė "$cls.name"

Klasės pavadinimas. *$cls.name.*
Aprašymas. *$cls.documentation*

```
#end
#macro( printEnumClass $cls )#set( $enumLits =
$sorter.sort($cls.getOwnedLiteral(), "name") )
```

1.3 lentelė. *Reikšmių klasė "$cls.name"*

Klasės pavadinimas. *$cls.name.*
Aprašymas. *$cls.documentation*

Reikšmės
#if($enumLits.size() > 0)#set($cnt = 0)#foreach($enumLit in $enumLits)#set($cnt = $cnt + 1) **$cnt.** $enumLit.name **Aprašymas.** $enumLit.documentation #end#else#**#Nėra#end

```
#end
#macro( printClass $cls )#if(
$cls.getClass().getName().endsWith(".EnumerationImpl")
)#printEnumClass($cls)#else#printStdClass($cls)#end#end

      #macro(  printClsDiagram  $diag  )#printDiagram($diag)#set(
$diagClss  =  ""  )#getDiagramClasses($diag,  $diagClss)#foreach(
$diagCls in $diagClss )#printClass($diagCls)#end#end

#set( $basePkgName = "Bendras pradinis kelias" )

#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos " )
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printUcDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".diagramos pavadinimas" )
#resolveDiag($diag, $diag)
#printDiagram($diag)

#set( $basePkgName = "Sistemos reikalavimų specifikacija" )
```

1. Sistemos reikalavimų specifikacija

1.1. Funkcijų hierarchija

```
#set( $pkg = $basePkgName+".kelias iki diagramos " )
#set( $diag = $pkg+".funkcijų hierarchijos diagrama" )
#resolveDiag($diag, $diag)
#printUcDiagram($diag)

#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".funkcijų priklausomumo aktoriams diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

1.2. Rezultatai

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Rezultato diagrama" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)
```

1.3. Duomenų šaltiniai

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Duomenų šaltinio diagrama" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)
```

1.4. Duomenų srautai

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Duomenų srautų diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

1.5. Duomenų šaltinių apdorojimo etapai

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Duomenų apdorojimo etapų veiklos diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

1.6. Būsenų kaita

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Būsenų kaitos diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

1.7. Duomenų modelis

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Duomenų modelio diagrama" )
#resolveDiag($diag, $diag)
#printClsDiagram($diag)

set( $basePkgName = "Sistemos projektavimas" )
```

2. Sistemos projektas

2.1. Duomenų modelis

```
#set( $pkg = $basePkgName+".kelias iki diagramos" )
#set( $diag = $pkg+".Duomenų modelio diagrama" )
#resolveDiag($diag, $diag
#printClsDiagram($diag)
```

2.2. Sistemos meniu struktūra

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Sistemos meniu struktūros diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```

2.3. Naudotojo sąsajos prototipai

```
#set( $pkg = $basePkgName+".kelias iki diagramos")
#set( $diag = $pkg+".Būsenų kaitos diagrama" )
#resolveDiag($diag, $diag)
#printDiagram($diag)
```