KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Orinta Tekoriūtė

Informacinės sistemos reikalavimų modelio atvaizdavimas į projektą, remiantis RUP

Magistro darbas

Darbo vadovas

Lekt. dr. Lina Čeponienė

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Orinta Tekoriūtė

Informacinės sistemos reikalavimų modelio atvaizdavimas į projektą, remiantis RUP

Magistro darbas

Recenzentas

doc. dr. A. Lenkevičius

2009-01-08

Vadovas

lekt. dr. L. Čeponienė
2009-01-12

Atliko

IFM-3/4 gr. stud.
Orinta Tekoriūtė

2009-01-12

Kaunas, 2009

Turinys

Santrauka

Šiame darbe siekiama padidinti informacijos sistemų kūrimo proceso automatizavimo laipsnį, taip palengvinant IS kūrėjų darbą. Darbe tobulinamas IS reikalavimų analizės ir transformavimo į projektą metodas EMDA (angl. *Extended Model Driven Approach*), kuris automatizuoja transformaciją iš reikalavimų į projekto modelį. Ši transformacija skirta paslaugų informacinėms sistemoms ir remiasi vienu pasiūlytu architektūriniu būsenų koordinatoriaus šablonu, pagal kurį atlieka transformaciją ir reikalavimų į projektą. Šiame darbe EMDA metodas tobulinamas, sukuriant transformaciją iš reikalavimų į projekto modelį, remiantis kitu informacinių sistemų kūrimo metodu RUP (angl. *Rational Unified Process*) bei MVC (angl. *Model-View-Controller*) architektūriniu šablonu. Darbe aprašomas siūlomas transformavimo algoritmas ir jo realizacija, panaudojant CASE įrankį *MagicDraw*. Sukurtas sistemos vedlys (angl. *Wizard*) leidžia išvengti klasių diagramos kūrimo rankiniu būdu, taip automatizuojant diagramos (ir jos elementų) kūrimą.

Summary

Mapping Information System Requirements to Design, According to RUP Principles

This work aims to increase the degree of automation of information system development process, thus facilitating the work of developers. *Extended Model Driven Approach (EMDA*) automates the transformation from the requirements to the design model, based on only one architectural pattern (State Coordinator) and performs the transformation using this pattern. The method proposed in this work was developed by creating a transformation of the requirements to the project model, based on other information systems development approach RUP (Rational Unified Process) and MVC (Model-View-Controller) architectural pattern. The proposed transformation algorithm was implemented in as an extension to CASE tool MagicDraw. Developed system wizard automates the creation of class diagram (and its elements).

1. Įvadas

Informacijos sistemų kūrimo procese būtų labai naudinga kuo labiau padidinti proceso automatizavimo laipsnį ir taip palengvinti informacijos sistemų kūrėjų darbą. L.Čeponienės pasiūlytas IS reikalavimų analizės ir transformavimo į projektą metodas EMDA (angl. *Extended Model Driven Approach*) [1] automatizuoja transformaciją iš reikalavimų į projekto modelį, tačiau jis remiasi tik vienu pasiūlytu architektūriniu šablonu ir atlieka transformaciją pagal šį šabloną. Todėl šiame darbe siekiama patobulinti EMDA metodą, sukuriant transformaciją iš reikalavimų į projekto modelį, remiantis kitu informacinių sistemų kūrimo metodu bei šablonu.

Kuriama metodika turi koncentruotis į transformaciją iš reikalavimų, kurie jau yra aprašyti EMDA metode, į projektą. Daroma prielaida, jog yra gaunamas jau surinktas, išanalizuotas ir patikrintas reikalavimų modelis, kurį reikia transformuoti į projekto modelį. Tam, kad aprašyti tokią metodiką, atliekama projektavimo metodų ir šablonų (angl. *design pattern*) analizė ir pasirenkamas konkretus šablonas.

Tyrimo tikslas - patobulinti reikalavimų transformavimo į projektą metodą EMDA, papildant jį transformacija, sudaryta remiantis RUP (angl. *Rational Unified Process)* ir pasirinkto architektūrinio šablono principais.

Tyrimo uždaviniai:

- atlikti informacijos sistemų kūrimo metodų analizę;
- išanalizuoti UML (angl. *Unified Modelling Language*) kalbą, pagrindinius MDA (angl. *Model Driven Architecture)* principus;
- išanalizuoti architektūrinių projektavimo šablonų reikalavimus ir pasirinkti konkretų šabloną;
- atlikti UML CASE (angl. *Computer Aided Software Engineering*) įrankių analizę ir pasirinkti algoritmo realizavimui tinkamą CASE įrankį;
- aprašyti duomenų (reikalavimų) ir rezultatų (projekto) metamodelį, atitinkantį pasirinkto architektūrinio projektavimo šablono principus;
- sukurti reikalavimų modelio atvaizdavimo į projektą algoritmą pagal pasirinkto architektūrinio projektavimo šablono principus;
- išbandyti algoritmo taikymą pasirinktai informacijos sistemai.

Šį darbą sudaro įvadas, keturi pagrindiniai skyriai, išvados, literatūros sąrašas bei vienas priedas.

Pirmajame darbo skyriuje nagrinėjama modeliais grindžiama architektūra MDA; IBM/Rational organizacijos programų sistemų kūrimo procesas RUP; ribinio programavimo, programų kūrimo procesas (XP) bei ICONIX kūrimo procesas. Analizuojami architektūriniai projektavimo šablonai, CASE įrankių automatizavimo galimybės.

Transformavimo iš informacinės sistemos reikalavimų modelio į projekto modelį algoritmas aprašomas antrajame darbo skyriuje. Speficikuojami transformavimo algoritmo panaudojimo atvejai.

Trečias darbo skyrius aprašo siūlomo transformavimo algoritmo realizaciją, panaudojant MagicDraw priemones: klasių diagramos vedlį (angl. *Class Diagram Wizard*) ir modelių transformavimo priemonę „*AnytoAny*“, kuri transformavimo metu kopijuoja sistemos elementus: kopija yra identiška originalui, o transformuoti elementai virsta savarankiškais. Klasių diagramos vedlio pagrindu sukurtas naujas vedlys „*Transformation from DIM to PIM*“, realizuojantis transformavimo iš IS reikalavimų modelio į projekto modelį algoritmą. Vedlys kuria naują klasių diagramą, kuomet visos klasės ir ryšiai tarp jų jau yra sukurti ir nurodyti analitiko. Kuriama priemonė leidžia išvengti klasių diagramos kūrimo rankiniu būdu, taip automatizuojant diagramos (ir jos elementų) kūrimą. Vedlys per keletą žingsnių surenka informaciją iš sistemos projektuotojo ir automatiškai sugeneruoja klasių diagramą su projektuotojo nurodytais elementais ir ryšiais tarp jų.

Eksperimentinis transformavimo algoritmo tyrimas aprašomas ketvirtajame darbo skyriuje. Pirmiausia sukurtas algoritmas išbandytas rankiniu būdu, projektuojant skaitmeninių video diskų (DVD) nuomos sistemą. CASE įrankiu realizuoto algoritmo išbandymui naudotas elektroninio mokyklos dienyno pavyzdys. Aprašyti reikalavimų (duomenų) ir projekto (rezultatų) modeliai.

Darbo rezultatai apibendrinami išvadose. Darbo priede pateikiama realizuoto transformavimo algoritmo vartotojo instrukcija.

2. IS kūrimo metodų ir architektūrinių šablonų analizė

2.1. Analizės tikslas

Informacijos sistemos kūrimo metodų ir architektūrinių šablonų analizės tikslas:

- atlikti informacijos sistemų kūrimo metodų analizę, išanalizuoti esamas galimybes RUP principams taikyti, pagrindinius MDA principus, UML kalbą;
- atlikti UML CASE įrankių analizę;
- išanalizuoti architektūrinius projektavimo šablonus.

2.2. Tyrimo sritis, objektas ir problema

Tyrimo sritis – informacijos sistemų kūrimo proceso automatizavimo galimybės.

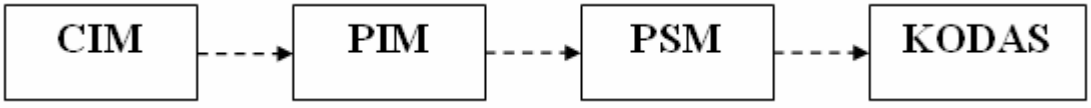
Tyrimo objektas – informacijos sistemų kūrime taikomi metodai, šablonai ir jų sudarymas.

Tyrimo problema – egzistuoja nedidelis informacijos sistemų (IS) kūrimo proceso automatizavimo laipsnis, dėl kurio sistemų kūrėjų darbas remiasi daugiausia intuicija ir vyksta neefektyviai. EMDA metodas siekia padidinti kūrimo proceso automatizavimo laipsnį, tačiau jame aprašyta transformacija iš reikalavimų į projektą remiasi tik vienu pasiūlytu architektūriniu šablonu. Šį metodą reikėtų patobulinti pritaikant jam RUP projektavimo principus ir pasirinktą architektūrinį projektavimo šabloną, bet tam reikia sukurti šablono principus atitinkantį projekto metamodelį ir aprašyti algoritmą, kaip, taikant šį šabloną reikalavimų modeliui, atlikti transformaciją į projekto modelį.

2.3. MDA architektūros analizė

MDA architektūroje IS kūrimas grindžiamas modeliais ir transformacijomis tarp jų. MDA išskiria nuo skaičiavimų nepriklausomą modelį CIM (angl. *Computation Independent Model*), kuris aprašo sistemos logiką nuo skaičiavimų nepriklausomu būdu. Modeliuojant veiklą aprašoma visa nagrinėjama sistema, nepriklausomai nuo to, ar ji bus kompiuterizuota ar ne. Veiklos modelis apima daugiau informacijos ir yra didesnis už sistemos reikalavimų modelį. Reikalavimų analizės ir specifikavimo etapas modeliais grindžiamoje architektūroje nėra detaliai apibrėžtas. Dalykinės srities modelis yra grindžiamas UML diagramų pavidalu. Projektuojamos sistemos UML aprašas nėra apribojamas konkrečia programavimo aplinka arba kūrimo platforma. Šis aprašas yra aiškus ir suprantamas projektuotojams, todėl jis, esant poreikiui, gali būti redaguojamas. Sistemos realizavimui dažniausiai naudojamas nuo realizavimo platformos nepriklausomas modelis PIM (angl. *Platform Independent Model*).

MDA architektūros principai vaizduojami 1 pav.

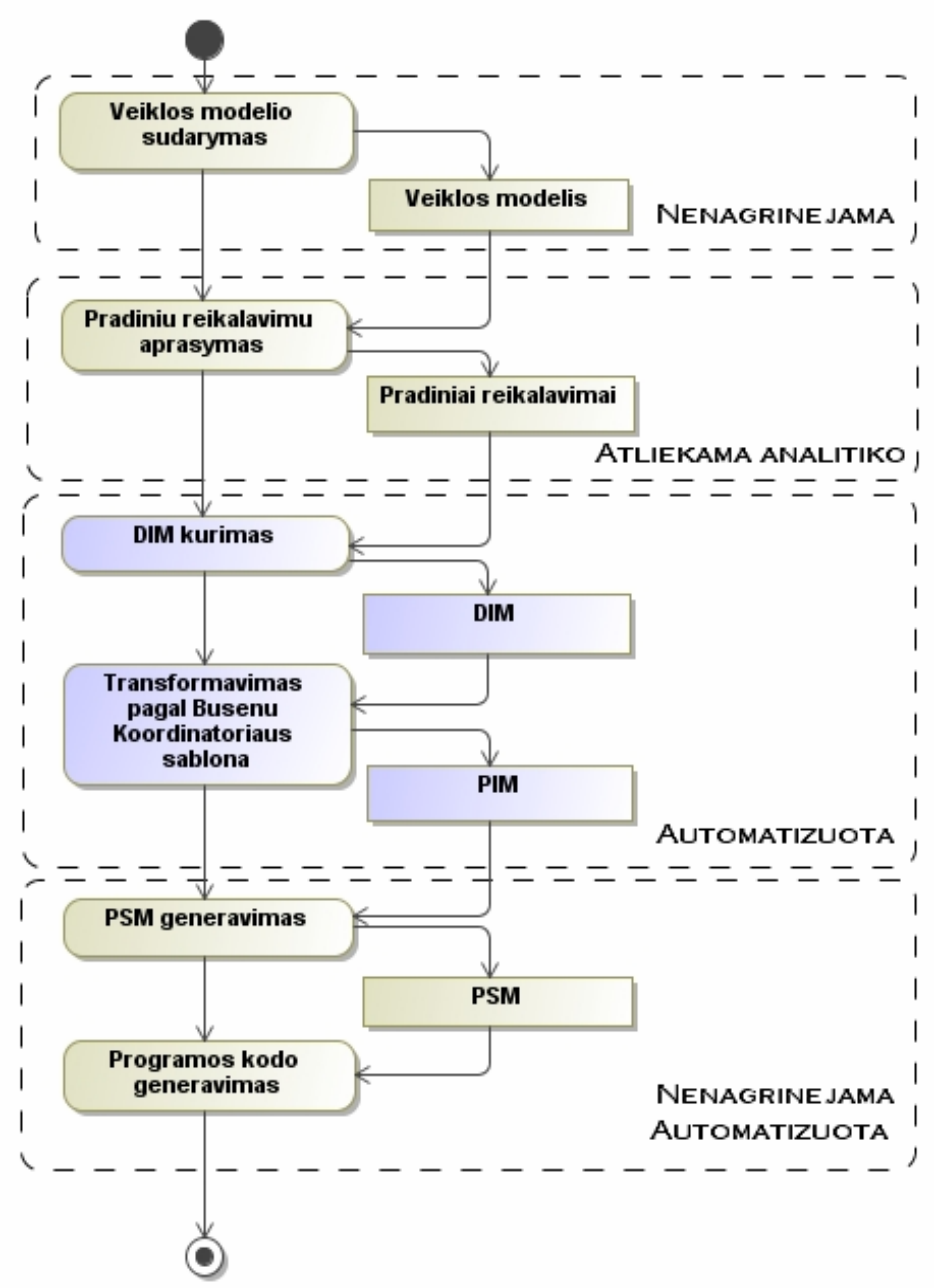


1 pav. MDA architektūros principinė schema

Perėjimas nuo PIM prie PSM galimas keliais būdais: projektuotojas gali naudoti paruoštus šablonus; projektuotojui pasirinkus reikiamą platformą, CASE įrankiai automatiškai atlieka PIM perėjimą į PSM. CASE įrankis gali būti nuolat tobulinamas [5].

2.4. EMDA analizė

EMDA – metodas (2 pav.), kuris leidžia detaliai aprašyti ir suderinti kuriamos operacinės sistemos reikalavimų modelį, automatizuoti perėjimą nuo reikalavimų lygmens modelio prie projekto [1]. Šis modelis aprašo informacinės sistemos struktūrą ir elgseną, nepriklausomai nuo architektūros ar projektavimo metodo, tačiau svarbu naudojant UML diagramas suderinti DIM ir PIM modelius.



2 pav. Pagrindiniai EMDA metodo etapai [1]

EMDA metodo etapai:

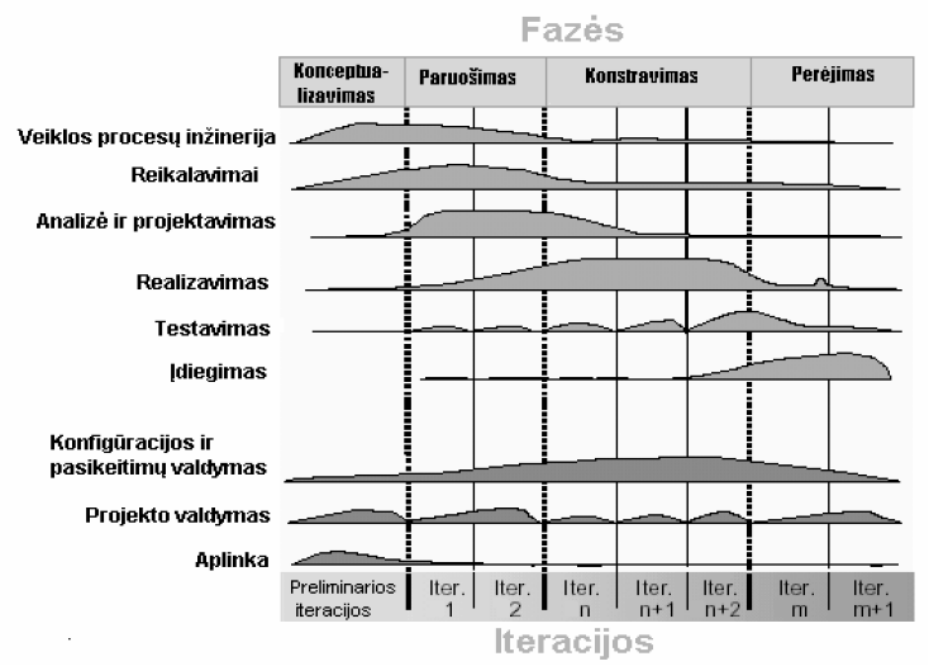
1. *Pradinių reikalavimų aprašymas.* Aprašymą sudaro panaudojimo atvejų bei dalykinės srities modeliai. Abu modeliai tarpusavyje yra susieti ir suderinti grįžtamuoju ryšiu, iš kurio gaunamos panaudojamo atvejų ir dalykinės srities klasių diagramos.
2. *DIM modelio sudarymas.* Šį etapą galima suskirstyti į DIM specifikacijos derinimą (sekos ir būsenos diagramų kūrimas) ir DIM specifikacijos kūrimą (aktorių/vartotojų modelių kūrimas, interfeisų modelio kūrimas, dalykinės srities modelio papildymas). Iš šio etapo gaunamos DIM interfeisų ir aktorių klasių diagrama bei DIM esybių klasių diagrama.
3. *PIM modelio generavimas.* Etapą sudaro aktorių paslaugų modelio generavimas; paslaugų modelio generavimas; dalykinės srities modelio generavimas.

PIM suderinamumas yra užtikrinamas savaime, nes PIM yra automatiškai gaunamas iš DIM. DIM nuo PIM skiriasi tuo, kad PIM turi būti papildytas reikalingomis klasėmis ar komponentais pagal pasirinktos architektūros taisykles. EMDA aprašyti reikalavimai yra paskirstomi architektūros elementams pagal pasirinktą šabloną. Šis metodas automatizuoja transformaciją iš reikalavimų į projekto modelį, tačiau transformacija realizuota tik pagal vieną pasiūlytą architektūrinį šabloną; metodas skirtas paslaugų sistemoms modeliuoti. Būtų naudinga patobulinti EMDA metodą, sukuriant transformaciją iš reikalavimų į projekto modelį, remiantis kitu informacinių sistemų kūrimo metodu bei šablonu. Tai projektuotojui suteiktų galimybę pasirinkti projektavimo šabloną iš kelių siūlomų, taip nusprendžiant kokiu principu remiantis atlikti transformaciją.

2.5. IS kūrimo metodų analizė

2.5.1. RUP metodas

RUP yra iteracinis, pagrįstas panaudojimo atvejais ir akcentuojantis architektūrą programinės įrangos kūrimo procesas [2, 7]. Iteraciniai metodai leidžia atsiradusius nesusipratimus nustatyti ankstyvojoje kūrimo stadijoje, vartotojas gali detaliau išsiaiškinti realius sistemos reikalavimus, o projekto komandos gali koncentruotis į svarbiausius tikslus. Pakartotiniai testavimai leidžia objektyviau įvertinti projekto būseną. Reikalavimų, projektavimo ir realizavimo netikslumai atrandami anksčiau, be to projekto komandos apkrovimas yra tolygiau paskirstomas, pati komanda gali mokytis iš ankstesnių iteracijų ir gerinti kūrimo procesą. RUP procese yra 4 etapai, kurių metu turi būti atliktos 9 veiksmų sekos (3 pav.):



3 pav. RUP fazės ir iteracijos [7]

RUP etapai:

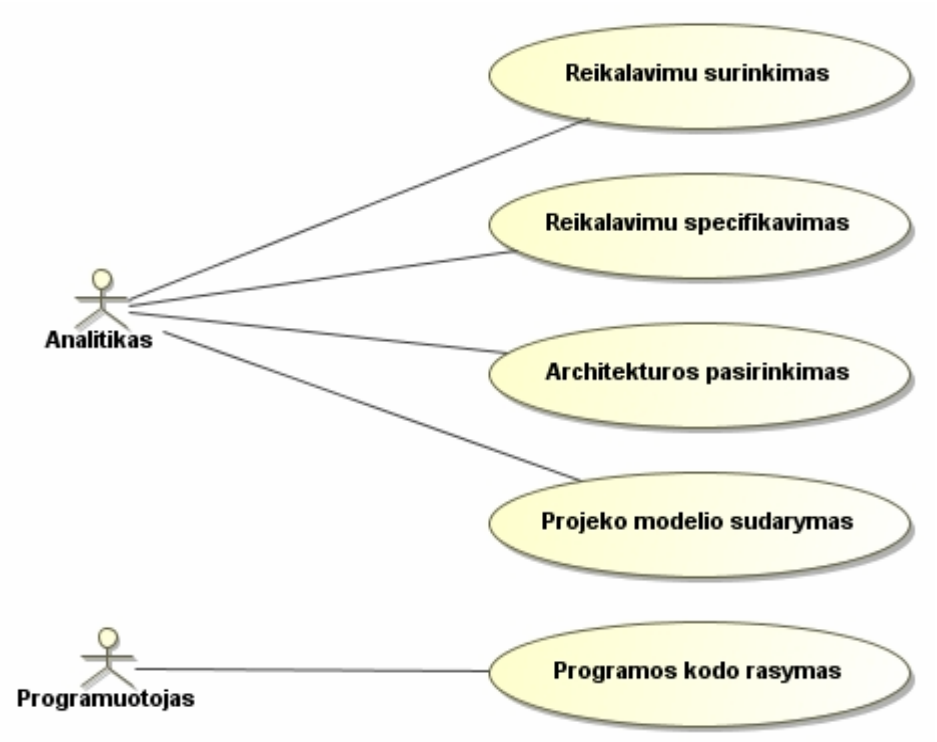
1. *Konceptualizavimas*. Šiame etape atliekama veiklos procesų inžinerija, atrenkami pradiniai reikalavimai, atliekama minimali analizė ir projektavimas; sukuriama konfigūracija bei pasikeitimų valdymas; atliekamas projektavimo valdymas; sukuriama pradinė aplinka. Visa tai vykdoma pradiniame iteracijos žingsnyje.

2. *Paruošimas.* Atliekamos mažiausiai 2 iteracijos, kurių metu tikslinami veiklos procesai ir reikalavimai; atliekama maksimali analizė ir projektavimas; įvertinamas minimalus realizavimas; kiekvieno žingsnio metu atliekamas sistemos testavimas. Pagal gautus rezultatus keičiama konfigūracija, pertikrinamas projekto valdymas (į aplinką praktiškai nekreipiant dėmesio).

3. *Konstravimas.* Minimaliai redaguojama veiklos procesų inžinerija; tikslinami reikalavimai; iš pasiruošimo metu gautos analizės ir projektavimo daromi atitinkami koregavimai. Realizavimo fazė šiame etape yra maksimali. Kiekvieno žingsnio metu atliekamas naujas testavimas. Įdiegimas pradedamas paskutiniame fazės etape. Ypatingai atkreipiamas dėmesys į konfigūraciją ir pasikeitimų valdymą. Projekto valdymas tobulinamas kiekviename žingsnyje.

4. *Perėjimas.* Šiame etape daugiausia dėmesio skiriama modelio įdiegimui. Konfigūracijos ir pasikeitimų valdymas supaprastėja.

Principinė IS kūrimo naudojant RUP proceso metu atliekamų panaudojimo atvejų diagrama patekta 4 paveiksle.



4 pav. Pagrindiniai RUP panaudojimo atvejai IS kūrime

Panaudojimo atvejis *„Reikalavimų surinkimas“*: analitikas pagal vartotojo poreikius surenka reikalavimus. (Analitikas bet kuriuo laiko momentu gali nutraukti reikalavimų rinkimą ir pratęsti darbą vėliau).

Panaudojimo atvejis *„Reikalavimų specifikavimas“*: analitikas atlieka probleminės srities analizę, specifikuoja reikalavimus. (Analitikas bet kuriuo laiko momentu gali papildyti reikalavimų specifikacijas).

Panaudojimo atvejis *„Architektūros pasirinkimas“*: analitikas pasirenka architektūrinį projektavimo šabloną. (Jei analitikas nepasirinko konkretaus architektūrinio šablono – projekto modelio sudarymas negali būti vykdomas).

Panaudojimo atvejis *„Projekto modelio sudarymas“*: analitikas UML diagramomis apibrėžia projektuojamos sistemos struktūrą ir elgseną.

Panaudojimo atvejis „*Programos kodo rašymas*“: programuotojas rašo programos kodą, jį testuoja, šalina atsiradusias klaidas, yra atsakingas už projekto vizualųjį sprendimą. (Programuotojas bet kuriuo momentu gali ištrinti duomenis ir įvesti naujus).

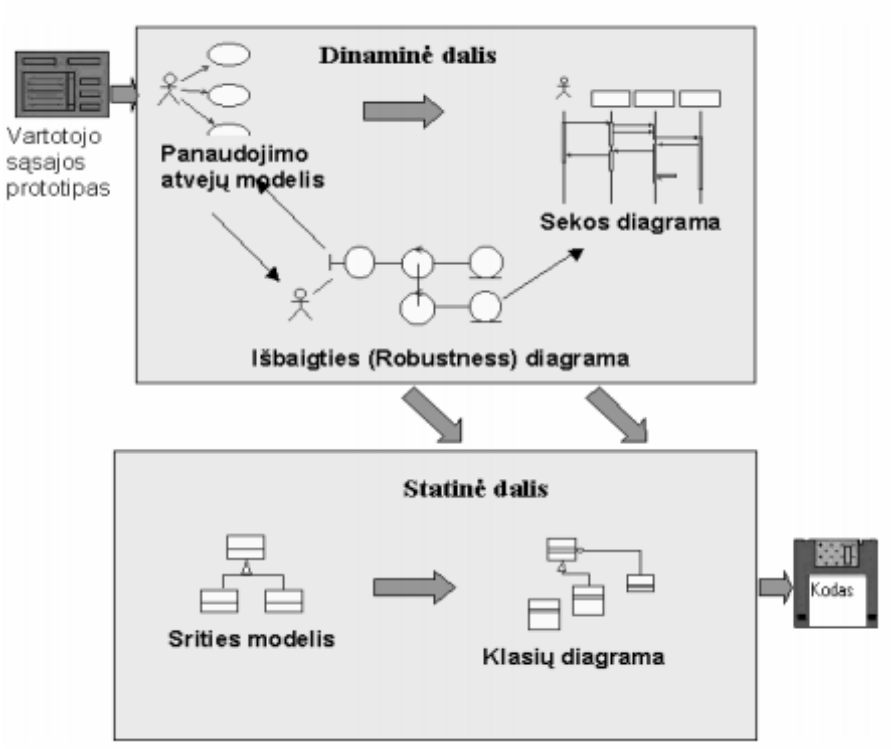
2.5.2. XP metodas

XP (angl. *Extreme Programming*) – ekstremalaus programavimo metodas, akcentuojantis kuo greitesnį rezultato gavimą, minimizuojant reikalingos dokumentacijos kiekį bei jos ruošimo laiką. Jis sudarytas iš paprastų, tarpusavyje persipinančių taisyklių, o šių taisyklių visuma turi žymiai didesnį svorį nei kiekviena taisyklė atskirai [24]. Tai yra siauras ir tik į programos kodą orientuotas metodas, kuriame privaloma viena diagrama. Šiame procese siekiama kuo greičiau pereiti nuo projektavimo prie programos kodo rašymo. XP galima laikyti mažu RUP poaibiu. Kadangi faktoriai, veikiantys projektavimą, sparčiai kinta, XP procesu siekiama atidėti projektavimo sprendimų priėmimą iki to momento, kuomet reikia pradėti programuoti. Naudojant XP, supratimas apie reikalavimų detales gaunamas tiesiogiai bendraujant su užsakovu. Pats XP projektas ciklišk ai pristato tarpines veikiančios programinės įrangos versijas. Tarpinės versijos pristatomos tam, kad sulaukti vartotojų atsiliepimų.

2.5.3. ICONIX metodas

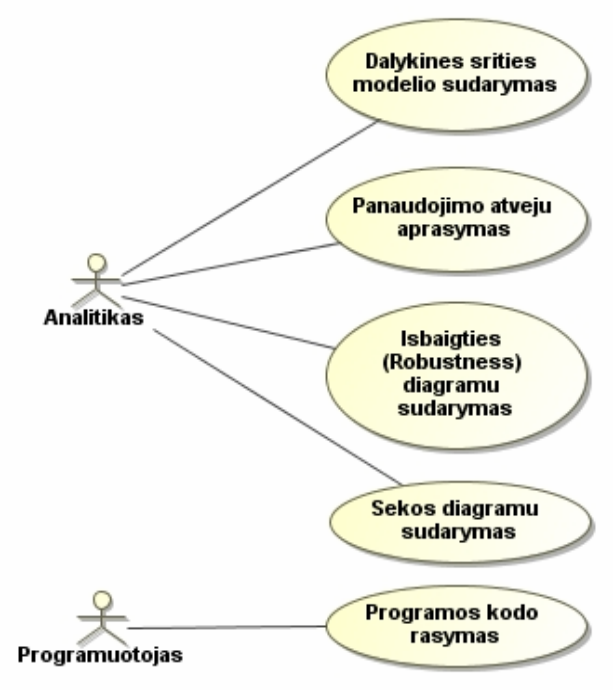
Programinės įrangos kūrimo metodas **ICONIX** pagrįstas minimaliu UML diagramų kiekiu [19]. Šis kūrimo procesas remiasi panaudojimo atvejų diagramomis. Pagrindinis proceso rezultatas yra sekos diagramos, kurios naudojamos kodo rašymui. ICONIX procesą sudaro 4 tipų diagramos (vaizduojamos 5 pav.):

1. klasių diagramos;
2. panaudojimo atvejų diagramos;
3. išbaigties (angl. *robustness)* diagramos;
4. sekos diagramos.



5 pav. ICONIX proceso struktūra [19]

ICONIX procese pagrindinis dėmesys skiriamas reikalavimų apibrėžimui. Reikalavimų specifikacijos išsamumą ir išbaigtumą kiekvienam panaudojimo atvejui užtikrina sekų ir išbaigties diagramos. Pagrindiniai šio metodo panaudojimo atvejai, vykdomi IS kūrimo metu, vaizduojami 6 pav.



6 pav. Pagrindiniai ICONIX metodo panaudojimo atvejai IS kūrime

Panaudojimo atvejis *„Dalykinės srities modelio sudarymas“*: analitikas atlieka probleminės srities aprašymą – sukuria srities modelį, kuris yra pradinis klasių diagramos variantas.

Panaudojimo atvejis *„Panaudojimo atvejų aprašymas“*: analitikas sudaro glaustai ir tiksliai aprašytas panaudojimo atvejų diagramas, kuriose pateikiami funkciniai reikalavimai kuriamai sistemai.

Panaudojimo atvejis *„Išbaigties (angl. Robustness) diagramų sudarymas“*: analitikas atlieka išbaigtumo analizę, kurios metu tikrinamas ir redaguojamas panaudojimo atvejų aprašymo glaustumas.

Panaudojimo atvejis *„Sekos diagramų sudarymas“*: analitikas sudaro sekos diagramas, kuriose aprašo panaudojimo atvejų modelyje aprašytą sistemos objektų elgesį.

Panaudojimo atvejis *„Programos kodo rašymas“*: programuotojas rašo programos kodą, jį testuoja, šalina atsiradusias klaidas, yra atsakingas už projekto vizualųjį sprendimą.

2.5.4. Analizuotų metodų palyginimas

RUP yra platus ir sudėtingas, o XP – minimalus, orientuotas į kodo rašymą. Iš IS projektavimo metodų galima išskirti ICONIX procesą, kuris pagrįstas minimaliu UML diagramų skaičiumi ir turi metodiką, kurios dėka kūrimo procesas ,,nuo panaudojimo atvejų iki kodo" [19] yra greitas ir efektyvus. Kaip ir RUP, jis remiasi panaudojimo atvejų diagramomis, bet nėra toks platus. Pagrindinis ICONIX proceso rezultatas – sekos diagramos, kurios kartu su statine klasių diagrama toliau naudojamos kodo rašymui.

Kadangi XP ir ICONIX metodai yra RUP poaibiai, pasirinktas RUP metodas, nes jame aiškiai atskirti reikalavimų analizės ir projektavimo etapai. Pats metodas pagrįstas užduotimis (angl. *Use - Case Driven*), per kurias išreiškiami reikalavimai. RUP procese akcentuojama architektūra (angl. *Architecture - centric*), projektas gali būti skaldomas į smulkesnes dalis - subprojektus. RUP leidžia iteratyviai kurti programinę įrangą, valdyti reikalavimus, naudoti komponentais paremtas (angl. *component-based*) architektūras, vizualiai modeliuoti programinę įrangą (naudojant diagramas), nuolat tikrinti programinės įrangos kokybę ir kontroliuoti programinės įrangos pasikeitimus. Pagrindinis trūkumas – RUP procesas daugeliu atvejų yra per daug formalus: nėra aiškaus ir konkretaus aprašymo kaip metodą būtų galima naudoti automatizavimui transformacijai iš reikalavimų modelio į projektą. Tam reikalingi papildymai ir projektuotojo intuicija. 1 lentelėje pateikiamas nagrinėtų metodų palyginimas.

1 lentelė. IS kūrimo metodų palyginimas

Palyginimo kriterijai	**RUP**	**ICONIX**	**XP**
Naudojami diagramų tipai	Klasių diagrama; panaudojimo atvejų diagrama; išbaigties (angl. *robustness)* diagrama; sekų diagrama; veiklos diagrama; bendradarbiavimo diagrama; būsenų diagrama; paketų diagrama; navigacijos diagrama; diegimo diagrama.	Klasių diagrama; panaudojimo atvejų diagrama; išbaigties diagrama; sekų diagrama.	Klasių diagrama; panaudojimo atvejų diagrama.
Kokias MDA transformacijas galima realizuoti	PIM->PSM->KODAS	Projektas->programos kodas	-

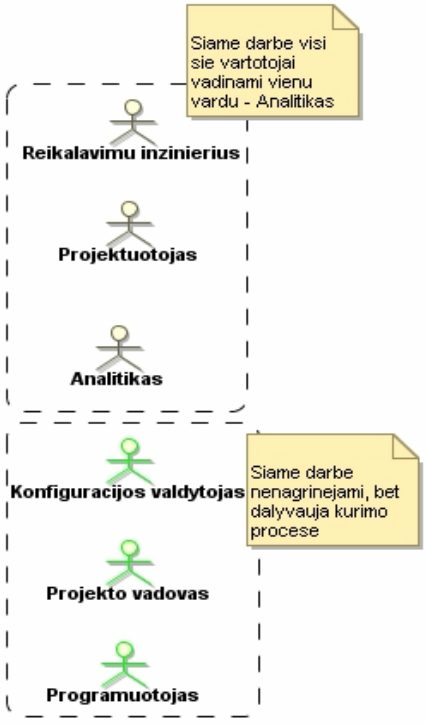
Ar reikalingas metodo pritaikymas konkrečiu atveju	Taip, nes RUP metodas yra labai platus, todėl reikia prisitaikyti pagal savo poreikius.	Nereikalingas.	Taip, nes nėra taisyklių, yra tik bendri darbo principai.
Ar galima atlikti transformavimą iš reikalavimų modelio į projekto modelį	Taip.	Ne, etapai nėra aiškiai atskirti.	Ne, nes nėra iš ko į ką transformuoti.
Ar aiškiai atskirti reikalavimų ir projekto etapai	Taip.	Ne, etapai nėra aiškiai atskirti.	Ne.
Dalykinė sritis aprašoma	Klasių diagrama.	Klasių diagrama.	Gali būti aprašoma klasių diagrama.
Vartotojo reikalavimai aprašomi	Panaudojimo atvejų diagrama.	Panaudojimo atvejų diagrama.	Panaudojimo atvejų diagrama.
Elgsena aprašoma	Būsenų diagrama; sekų diagrama; bendradarbiavimo diagrama; išbaigties diagrama; veiklos diagrama.	Sekų ir išbaigties (angl. *Robustness*) diagrama.	Elgsena neaprašoma - iškart rašomas programos kodas.
Etapų skaičius	4 etapai (konceptualizavimas; paruošimas; konstravimas; perėjimas); 9 veiksmų sekos; iteracijų skaičius neribojamas	4 etapai (vartotojo prototipas; statinė dalis – probleminės srities aprašymas; dinaminė dalis - panaudojimo atvejų aprašymas, išbaigties analizė, sekos diagramų sudarymas; programos kodas)	Ciklišai pristatomos tarpinės veikiančios PĮ versijos

RUP metode iteracijų skaičius yra neribojamas, todėl IS proceso projektavimo ir kūrimo supaprastinimui ir pagreitinimui naudojami projektavimo šablonai. Dažniausiai IS sistemos projektuotojas pasirenka tam tikrą šabloną ir jį pritaiko konkrečiai užduočiai spręsti. Kiekvienas projektavimo šablonas turi savo dalyvaujančias klases ir objektus, kurie turi savo roles (vaidmenis), bendradarbiauja tarpusavyje. Klasikiniai projektavimo šablonai – viena iš projektavimo priemonių, kuri konkretizuoja ir palengvina IS projektavimo procesą.

2.5.5. Vartotojų analizė

Vartotojų aibė, tipai ir savybės

Vartotojų tipai gali būti įvairūs: projektuotojai, projekto prižiūrėtojai, architektai, konfigūracijos valdytojai. Analizuojami IS kūrimo proceso vartotojų tipai (vaizduojami 7 pav.) – informacijos sistemų kūrėjai, analitikai, reikalavimų inžinieriai.



7 pav. Vartotojų tipai

Vartotojų tikslai ir problemos

Pagrindinis vartotojo tikslas – efektyvus darbas ir kokybiškas projekto modelis. Vartotojų problemos kyla dėl to, jog IS kūrimo procesas yra brangus ir neefektyvus, nes darbas – mažai automatizuotas ir dažnai remiasi IS kūrėjo intuicija.

2.6. Architektūrinių šablonų analizė

Projektavimo šablonas – tai pripažintas, dažnai pasitaikančios programinės įrangos projektavimo problemos sprendimo būdas [8]. Projektavimo pavyzdys tinka ne vienai probleminei sričiai – jis gali būti panaudotas įvairiose situacijose.

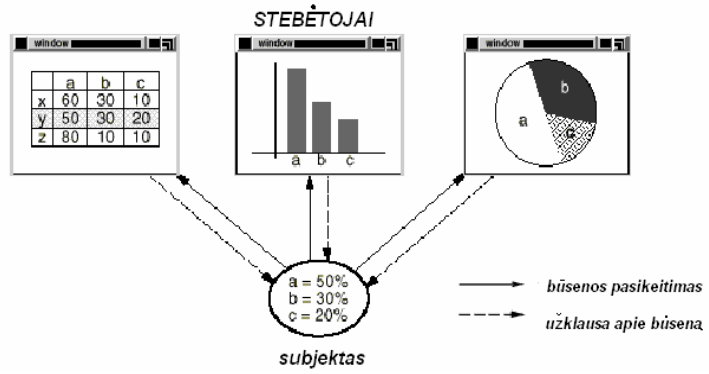
Gang of Four (GoF) projektavimo šablonai skirstomi į kūrimo (angl. *Creational*), struktūros (angl. *Structural*) ir elgsenos (angl. *Behavioral*) šablonų grupes [8]. Šiuos šablonus galima naudoti RUP procese [22].

Plačiau išnagrinėtas elgsenos šablonų taikymas transformacijos iš reikalavimų į projekto modelį realizacijai. Kadangi RUP procese dažnai rekomenduojama taikyti MVC šablono principus [22], iš GoF šablonų detaliau analizuotas ir taikymui transformacijoje pasirinktas Stebėtojo (angl. *Observer*) šablonas bei konkretesnis jo taikymas – MVC šablonas.

2.6.1. Architektūrinio Stebėtojo šablono analizė

Stebėtojo šablonas apibrėžia ir palaiko priklausomybę tarp objektų taip, kad, vienam objektui pakeitus būseną, visiems sistemoje esantiems objektams yra pranešama apie būsenos pasikeitimą ir kiekvienas jų automatiškai atnaujinamas [8].

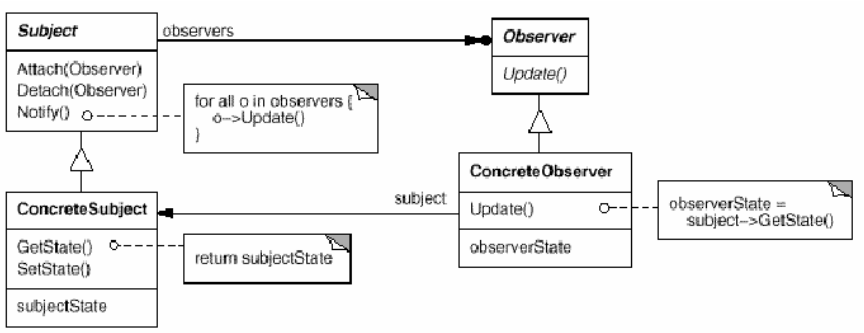
Šablono pagrindas – sistemos suskirstymas į tarpusavyje sąveikaujančias klases. Sistemoje dalyvaujančios klasės yra susietos tarpusavyje. Klasės, naudojančios ir atvaizduojančios duomenis, gali sąveikauti tarpusavyje arba nepriklausomai viena nuo kitos. Pavyzdžiui, objektais gali būti: skaičiuoklė, grafinė lentelė ar kiti objektai (pateiki 8 pav.). Šie objektai gali vaizduoti tuos pačius duomenis, naudodami skirtingus atvaizdavimo būdus.



8 pav. Stebėtojo šablono pavyzdys [8]

Skaičiuoklė ir grafinė lentelė neturi tiesioginio tarpusavio ryšio, todėl duomenų keitimui ir atvaizdavimui gali būti panaudota betkuri iš jų. Pakeitimai turi būti atvaizduoti kiekviename iš objektų. Vartotojui pakeitus informaciją skaičiuoklėje, grafinė lentelė iškart reaguoja į atsiradusius pakeitimus ir juos atvaizduoja (ir atvirkščiai).

Toks elgesys reiškia, kad skaičiuoklė ir grafinė lentelė yra priklausomos nuo duomenų objekto ir kiekvienas jo būsenos pasikeitimas turi būti fiksuojamas ir atvaizduojamas interfeisuose. Nepriklausomų objektų skaičiui nėra jokių apribojimų: gali būti bet koks skirtingų vartotojo interfeisų kiekis tiems patiems duomenims atvaizduoti. Stebėtojo šablonas (struktūra vaizduojama 9 pav.) nusako šių ryšių sudarymą.



9 pav. Stebėtojo šablono struktūra [8]

Pagrindiniai objektai – **subject** (*subjektas*) ir **observer** (*stebėtojas*). Subjektas gali turėti bet kokį nepriklausomų stebėtojų skaičių [8]. Visiems stebėtojams yra pranešama apie subjekto būsenos pasikeitimą. Atsakydamas į tai, kiekvienas stebėtojas sinchronizuoja savo būseną su subjekto būsena, siųsdamas objektui užklausą apie jo būseną.

Sąveikos rūšis – **publish-subscribe** (*pateikti-užsisakyti*). Subjektas yra pranešėjas: jis siunčia pranešimus apie būsenos pasikeitimą visiems stebėtojams. Kiekvienas stebėtojas siunčia užklausą subjektui apie jo būsenos pasikeitimus.

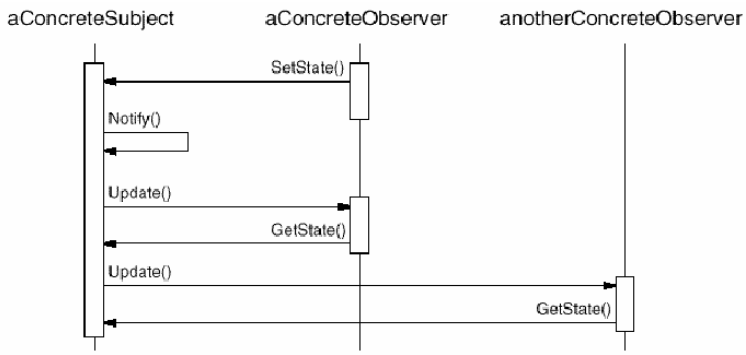
Klasė **Subject** *(Subjektas)* žino savo Stebėtojus . Betkoks Stebėtojų skaičius gali sekti Subjekto būsenas. Ši klasė turi pridėjimo, pranešimo ir išjungimo paslaugos interfeisus.

Klasė **Observer** *(Stebėtojas)* atnaujina objektų interfeisus pagal Subjekto būsenos pasikeitimo informaciją.

Klasė **ConcreteSubject** (*KonkretusSubjektas*) saugo KonkretausStebėtojo užklausos būseną ir siunčia pranešimą savo stebėtojui apie būsenos pasikeitimą.

Klasė **ConcreteObserver** (*KonkretusStebėtojas*) palaiko tiesioginį ryšį su klasės KonkretusSubjektas objektu ir saugo būseną, kuri susijusi su Subjektais. Ši klasė realizuoja Stebėtojo būsenos atnaujinimą.

Šablono veikimo principas gali būti aprašomas taip: konkretus Subjektas praneša savo Stebėtojams apie įvykusius būsenos pasikeitimus, kurie yra nesuderinti su jo esama būsena. Gavęs šią informaciją KonkretusStebėtojas ją panaudoja savo būsenos suderinimui su kitais Subjektais. 10 pav. pateikta diagrama vaizduoja tarpusavio ryšius tarp Subjekto ir dviejų Stebėtojų:



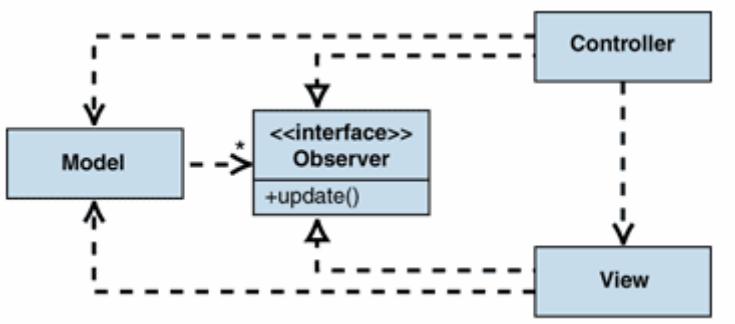
10 pav. Subjekto ir Stebėtojų tarpusavio ryšių diagrama

Stebėtojo objektas inicijuoja pasikeitimo užklausą ir atideda savo būsenos atnaujinimą iki tol, kol gauna patvirtinantį pranešimą iš Subjekto. Pranešimas taip pat gali ateiti iš bet kurio stebėtojo ar prisijungusio kitos rūšies objekto.

2.6.2. MVC šablono analizė

Klasikinis Stebėtojo šablono pavyzdys yra **MVC** (angl. *Model/View/Controller*) šablonas, kuriame, pasikeitus modelio būsenai, pokyčiai atvaizduojami kiekviename modelį sudarančiame interfeise [8]. RUP procese dažnai rekomenduojama naudoti MVC architektūrinį šabloną [22], todėl šiame darbe pasirinktas MVC šablono pritaikymas transformacijai iš DIM į PIM.

Šablono dalyviai - klasės ir objektai, dalyvaujantys projekto struktūroje. Struktūrinis santykis tarp trijų objektų parodytas 11 paveiksle:



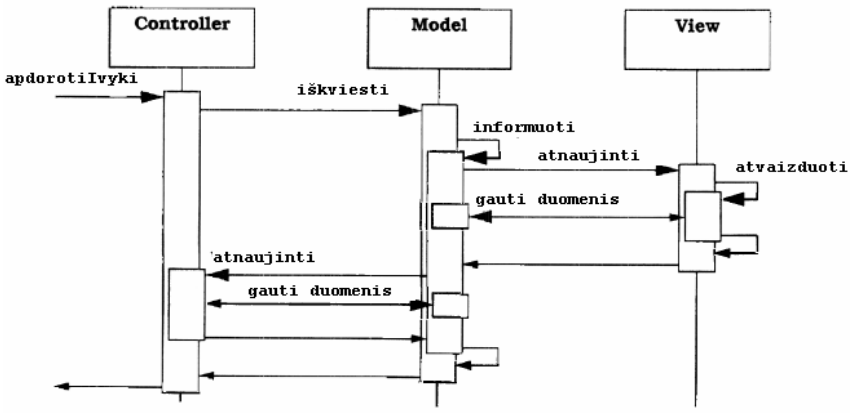
11 pav. MVC šablono struktūra

MVC šablonas susideda iš trijų objektų rūšių – modelio (angl. *Model*), vaizdo (angl. *View*) ir valdiklio (angl. *Controller*) [9; 13]. MVC principas pasireiškia logikos, prezentacijos ir turinio atskyrimu. Egzistuoja trys šablono dalys, kurios yra atsakingos už savo „veiklos sritį":

- **Modelis** (angl. *Model*) valdo dalykinės srities duomenis, atsako į *Vaizdo* užklausas dėl būsenos pakeitimo.
- **Vaizdas** (angl. *View*) valdo informacijos atvaizdavimą vartotojui [9], suteikia programos modeliui tokią išvaizdą, kokią supranta vartotojas. *Vaizdas* informaciją gauna iš *Modelio*.
- **Valdiklis** (angl. *Controller)* atsako už *Modelio* valdymą bei *Vaizdų* pateikimus pagal atitinkamus vartotojo veiksmus (mygtuko paspaudimas, teksto įvedimas). Valdiklis apibrėžia būdą, kuriuo vartotojo interfeisas reaguoja vartotojų įvesčiai. Šias įvestis Valdiklis traktuoja kaip įvykius [9].

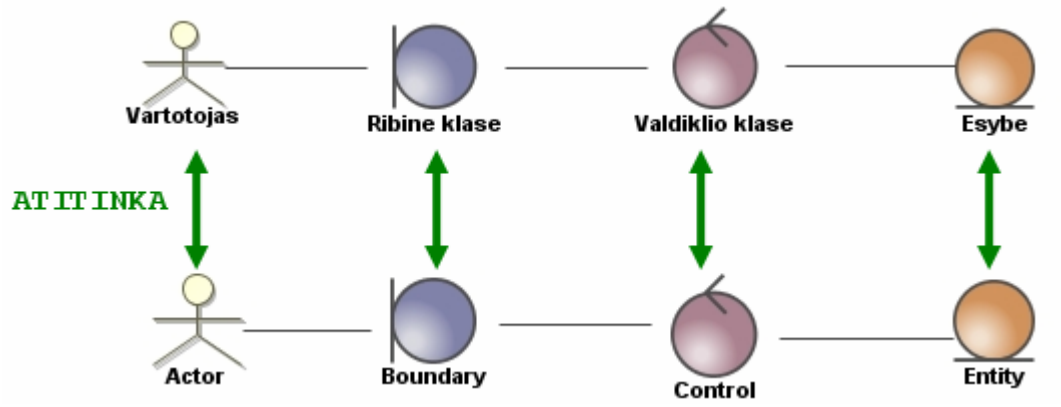
Esminis MVC procesas (proceso diagrama vaizduojama 12 pav.):

1. vartotojas atlieka tam tikrą veiksmą – įvestį;
2. Valdiklis priima ir apdoroja šį veiksmą pagal nurodytą įvykių apdorojimo procedūrą (angl. *event handler*) [12];
3. Valdiklis iškviečia Modelį, kuris atlieka veiksmus su duomenimis;
4. atitinkamas Vaizdas naudojasi Modeliu, kad sugeneruotų vartotojo sąsają (atvaizduotų naują informaciją);
5. vartotojo sąsaja laukia tolimesnių veiksmų iš vartotojo.



12 pav. MVC proceso diagrama [12]

RUP procese MVC principai matomi išbaigties (angl. *robustness*) diagramoje, kur vyksta minėtų sluoksnių atskyrimas. Diagramoje vaizduojami sistemos objektai ir jų tarpusavio ryšiai. Sistemoje dalyvauja trijų tipų objektai, kurie vaizduojami trimis skirtingomis ikonomis (13 pav.).



13 pav. Išbaigties diagramos objektų žymėjimas

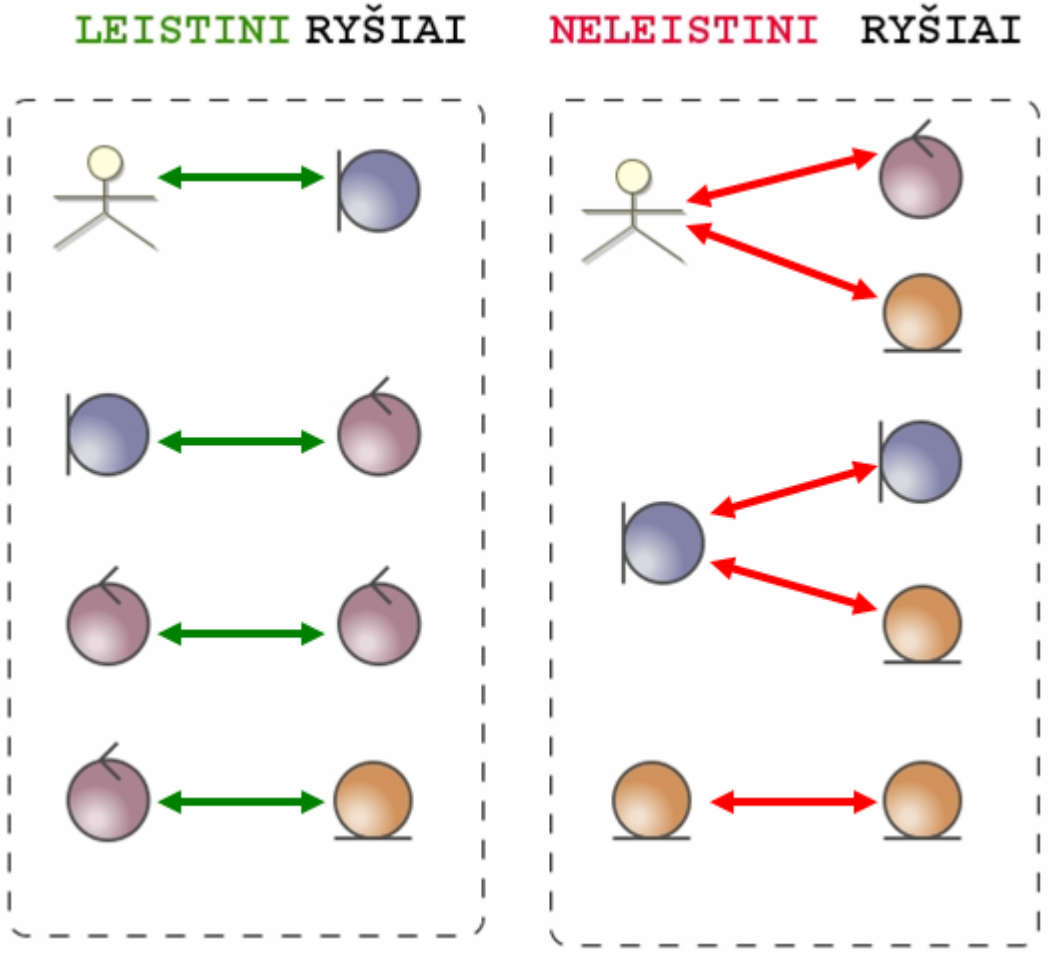
Ribiniai (angl. *Boundary*) objektai naudojami aktorių bendravimui su sistema.

Esybių (angl. *Entity*) objektai skirti atvaizduoti sistemos viduje nuolat vykstantiems duomenų/informacijos srautams.

Valdiklio (angl. *Control*) objektai – per juos vyksta bendravimas tarp ribinių ir esybių objektų [19].

Egzistuoja 4 taisyklės (vaizduojamos 14 pav.), kuriomis privaloma vadovautis IS kūrimo metu naudojant išbaigtumo diagramą:

1) sistemos aktorius gali bendrauti tik su ribiniais objektais;
2) ribiniai objektai gali bendrauti tik su aktoriais ir valdiklio objektais;
3) esybių objektai gali bendrauti tik su valdiklio objektais;
4) valdiklio objektai gali bendrauti tik su kitais valdiklio objektais, ribiniais ir esybių objektais, bet negali bendrauti su aktoriais [23].



14 pav. Išbaigtumo diagramos elementų tarpusavio bendravimo taisyklės

Išbaigtumo diagramos ir MVC šablono kombinaciją galima pritaikyti realizuojant perėjimą iš reikalavimų modelio į projektą.

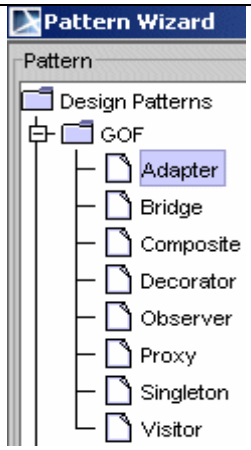
2.7. CASE įrankių automatizavimo galimybių analizė

Didesniuose projektuose specializuotų UML įrankių naudojimas leidžia dirbti daug efektyviau. Galima modeliuoti sistemą skirtingais abstrakcijos lygiais. Pavyzdžiui, kuriamos sistemos esybes ir jų ryšius vaizduojanti klasių diagrama gali būti naudojama reikalavimų

analizės metu, o vėliau pagal ją gali būti sukuriama detali realizacijos klasių diagrama, kurioje nurodomi specifiniai realizacijos kalbos duomenų tipai, atliekamos reikalingos ryšių transformacijos, pridedamos tik realizacijai reikalingos savybės (tokios kaip identifikaciniai kodai). Modeliavimas skirtingais abstrakcijos lygiais leidžia glaudžiau susieti programinės įrangos architektūros projektavimą su reikalavimų analizės veikla. CASE įrankių analizei pasirinkti *MagicDraw* bei *MS Visio* įrankiai; jų kriterijų palyginimas pateiktas 2 lentelėje.

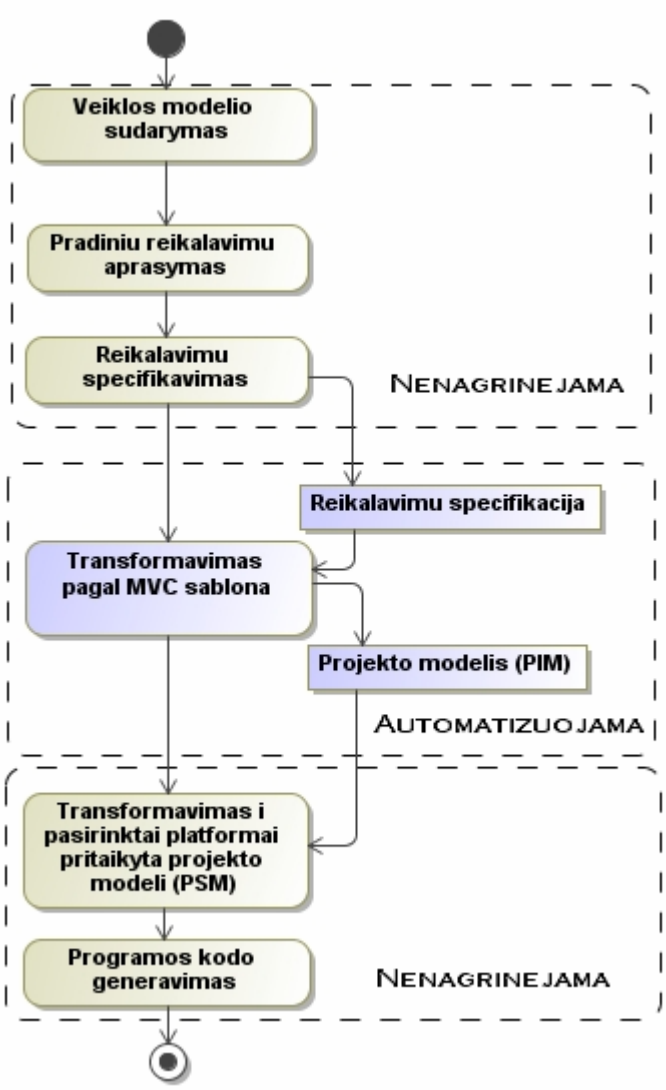
2 lentelė. MagicDraw ir MS Visio įrankių palyginimas

Palyginimo kriterijai	**MagicDraw (15.5)**	**MS Visio (2003)**
Palaikomų diagramų tipai	Klasių diagrama; panaudojimo atvejų diagrama; bendradarbiavimo diagrama; sekos diagrama; būsenų diagrama; veiklos diagrama; paketų diagrama; realizavimo diagrama; išbaigtumo diagrama; laiko diagrama; veiklos procesų diagrama; (Composit structure diagrama; Content diagrama; Web diagrama; Corba IDL diagrama; Generic DDL diagrama; Networking diagrama; WSDL diagrama; XML Schema diagrama; Struts diagrama; Oracle DDL diagrama; Content diagrama).	Klasių diagrama; panaudojimo atvejų diagrama; bendradarbiavimo diagrama sekos diagrama; būsenų diagrama; veiklos diagrama; paketų diagrama; statechart diagrama; komponentų diagrama; deployment diagrama.

Projektavimo šablonų naudojimas (*GoF*)		-
MVC karkasas	Palaikomi MVC stereotipai; yra išbaigtumo (angl. *Robustness*) diagrama.	-
Transformacijų galimybė	Yra: PIM ->PSM PSM -> PIM UML -> Generic DDL DDL -> UML UML -> Oracle DDL UML -> XML schema XML schema -> UML modelis	Yra: Duomenų bazės schemos transformavimas iš esybių ryšių modelio į SQLServerDB.
Diagramų susiejimas tarpusavyje	Tarpusavyje susietos klasių, sekų, bendradarbiavimo, veiklos, būsenų, panaudojimo atvejų diagramos.	Galimas tik kartu naudojant „UML Stencil“ įskiepį.
Profilių palaikymas	RUP; Model transformation; Use Case; Relationships; C#; JAVA; MOF	-
Mokymo medžiaga, pagalbos failai	Neišsami, mažas kiekis pagalbos failų.	Išsami, didelis kiekis pagalbos failų.

2.8. Siekiamas sprendimas

Siekiamas sprendimas - patobulinti reikalavimų transformavimo į projektą metodą, papildant jį transformacija, sudaryta remiantis RUP ir MVC architektūriniu šablonu; sukurti RUP pricipus atitinkantį architektūrinį šabloną ir aprašyti algoritmą, kad, taikant šį šabloną reikalavimų modeliui, atlikti transformaciją į projekto modelį. Siekiamas sprendimas vaizduojamas 15 pav.



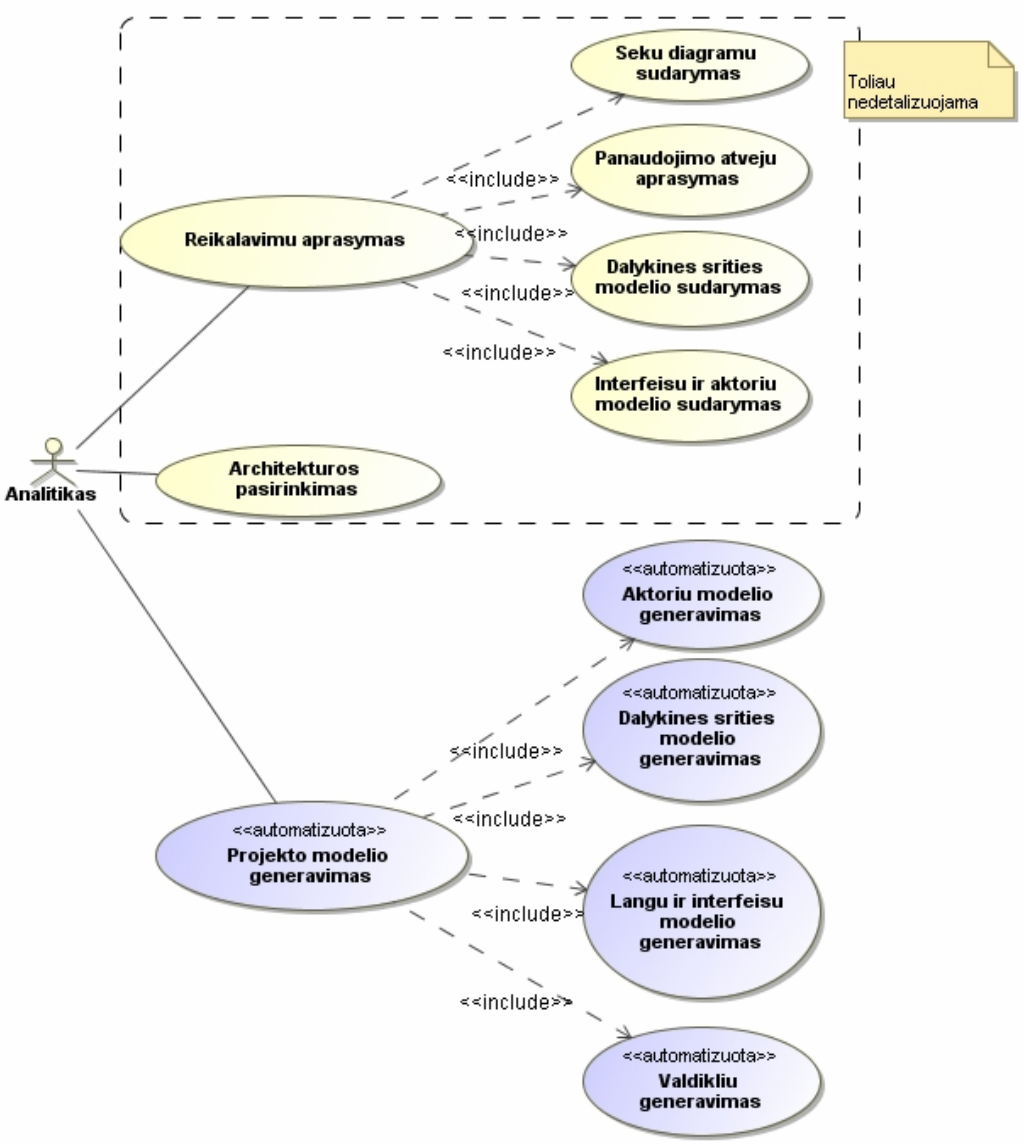
15. pav. Siekiamas sprendimas

2.9. Transformavimo algoritmo specifikacija

Pateikiamoje diagramoje (16 pav.) vaizduojami IS kūrimo metu analitiko atliekami panaudojimo atvejai. Šiame darbe detaliau nagrinėjami ir kompiuterizuojami panaudojimo atvejai:

- *Reikalavimų aprašymas* – atliekamas rankiniu būdu. Šiame darbe aktualus tik reikalavimų aprašymo rezultatas.
- *Architektūros pasirinkimas* – nėra detalizuojamas.

Toliau specifikuojamas (3 - 6 lentelės) ir nagrinėjamas panaudojimo atvejis „*Projekto modelio generavimas“*.



16 pav. Reikalavimų aprašymo ir projekto modelio generavimo panaudojimo atvejų diagrama

3 lentelė. Panaudojimo atvejo „Aktorių modelio generavimas“ specifikacija

PA „**Aktorių modelio generavimas**“		
Prieš sąlyga		Aktorių modelis aprašytas reikalavimų pakete. Analitikas prisijungęs prie sistemos.
Sužadinimo sąlyga		Transformacija *AnyToAny*
Susiję panaudojimo atvejai	**Išplečia PA**	-
	Apima PA	„Aprašyti reikalavimus“
	Specializuoja PA	-
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka transformacijos tipą.		**1.1**. Sistema leidžia pasirinkti aktorius transformacijai.
2. Vartotojas nurodo transformacijos rezultato paketą.		**2.1**. Sistema tikrina, ar toks paketas yra. Jei toks paketas yra, sistema leidžia atlikti transformaciją. Jeigu nėra – vartotojas sukuria naują projekto paketą.
3. Vartotojas patvirtina transformacijos rezultatų paketą.		**3.1.** Sistema atlieka transformaciją.
4. Vartotojas peržiūri aktorių sąrašą.		**4.1.** Sistema atvaizduoja aktorius projekto modelyje. Jeigu vartotojas nori įvesti naujus aktorius – sistema pereina į 5 įvykį. Jeigu ne - į 6.
5. Vartotojas sukuria naujus aktorius, įvesdamas jų vardus tam skirtame laukelyje.		**5.1.** Naujai sukurtus aktorius sistema atvaizduoja projekto pakete kartu su anksčiau sukurtaisiais.
6. Vartotojas pereina į kitą vedlio žingsnį, patvirtindamas aktorius “Next” mygtuko paspaudimu.		**6.1.** Sistema pereina į kitą algoritmo žingsnį.
Po sąlyga:		Projekto modelyje sugeneruotas ir išsaugotas aktorių modelis.

4 lentelė. Panaudojimo atvejo „Dalykinės srities modelio generavimas“ specifikacija

PA „Dalykinės srities modelio generavimas“		
Prieš sąlyga		Dalykinės srities modelis aprašytas reikalavimų pakete. Analitikas prisijungęs prie sistemos.
Sužadinimo sąlyga		Transformacija *AnyToAny*
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„Aprašyti reikalavimus“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka transformacijos tipą.		**1.1**. Sistema leidžia pasirinkti dalykinės srities objektus transformacijai.
2. Vartotojas nurodo transformacijos rezultato paketą.		**2.1**. Sistema tikrina, ar toks paketas yra. Jei toks paketas yra, sistema leidžia atlikti transformaciją. Jeigu nėra – vartotojas sukuria naują projekto paketą.
3. Vartotojas patvirtina transformacijos rezultatų paketą.		**3.1.** Sistema atlieka transformaciją.

4. Vartotojas peržiūri dalykinės srities objektų sąrašą.	**4.1.** Sistema atvaizduoja dalykinės srities objektus projekto modelyje. Jeigu vartotojas nori įvesti naujus objektus – sistema pereina į 5 įvykį. Jeigu ne - į 7.
5. Vartotojas sukuria naujus dalykinės srities objektus, jų vardus įvesdamas tam skirtame laukelyje.	**5.1.** Naujai sukurtus dalykinės srities objektus sistema atvaizduoja projekto pakete kartu su anksčiau sukurtaisiais.
6. Vartotojas kiekvienai esybei sukuria sukuria po 4 operacijas, įvesdamas jų pavadinimus esybių specifikacijų skiltyse.	**6.1.** Sistema kiekvienoje esybėje patalpina 4 sukurtasias operacijas.
7. Vartotojas pereina į kitą algoritmo žingsnį, patvirtindamas aktorius.	**7.1.** Sistema pereina į kitą algoritmo žingsnį.
Po sąlyga:	Projekto modelyje sugeneruotas ir išsaugotas dalykinės srities objektų modelis.

5 lentelė. Panaudojimo atvejo „Langų ir interfeisų modelio generavimas“ specifikacija

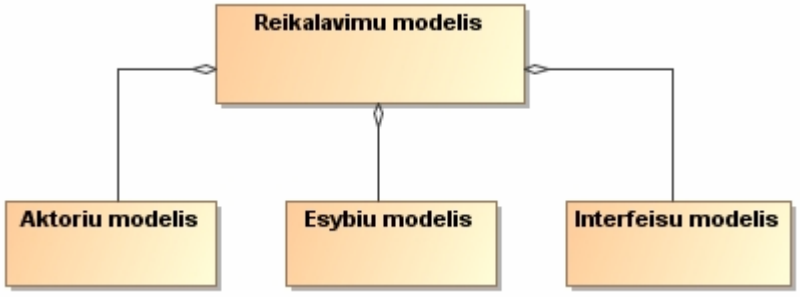
PA „**Langų ir interfeisų modelio generavimas**“		
Prieš sąlyga		Analitikas prisijungęs prie sistemos.
Sužadinimo sąlyga		Aktyvuojamas algoritmą realizuojantis įrankis.
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„Generuoti projektą“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka projekto modelio paketą.		**1.1**. Sistema leidžia pasirinkti paketą, kuriame bus talpinami rezultatai.
2. Vartotojas pasirenka peržiūrėti interfeisų sąrašą.		**2.1**. Sistema atvaizduoja interfeisų sąrašą.
3. Vartotojas įveda ribinių klasių pavadinimus.		**3.1.** Sistema sukuria ribines klases.
4. Vartotojas nurodo ryšius tarp ribinių objektų ir interfeisų		**4.1.** Sistema atvaizduoja nurodytus ryšius.
5. Vartotojas nurodo ryšius tarp ribinių objektų ir valdiklių.		**5.1.** Sistema atvaizduoja nurodytus ryšius.
6. Vartotojas nurodo ryšius tarp ribinių objektų ir aktorių.		**6.1.** Sistema atvaizduoja nurodytus ryšius.
Po sąlyga:		Projekto modelyje sukurtos ir išsaugotos ribinės klasės.

6 lentelė. Panaudojimo atvejo „Valdiklių modelio generavimas“ specifikacija

PA „**Valdiklių modelio generavimas**“		
Prieš sąlyga		Analitikas prisijungęs prie sistemos.
Sužadinimo sąlyga		Aktyvuojamas algoritmą realizuojantis įrankis.
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„Generuoti projektą“
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka projekto modelio paketą.		**1.1**. Sistema leidžia pasirinkti paketą, kuriame bus talpinami rezultatai.
2. Vartotojas pasirenka peržiūrėti interfeisų sąrašą.		**2.1**. Sistema vedlio lange atvaizduoja interfeisų sąrašą.
3. Vartotojas įveda valdiklio klasių pavadinimus.		**3.1.** Sistema sukuria valdiklio klases.
4. Vartotojas kiekvienai valdiklio klasei sukuria operaciją.		**4.1.** Sistema atvaizduoja sukurtąją operaciją valdiklio klasėje.
5. Vartotojas nurodo ryšius tarp valdiklio klasių ir ribinių objektų.		**5.1.** Sistema atvaizduoja nurodytus ryšius.
Po sąlyga:		Projekto modelyje sukurtos ir išsaugotos valdiklio klasės.

2.10. Reikalavimai duomenims

Reikalavimų modelį sudaro trys pagrindinės dalys (17 pav.): aktorių, esybių ir interfeisų modeliai. Šios dalys naudojamos transformacijai.



17 pav. Reikalavimų modelio sudėtis

Minimalūs reikalavimai transformavimo algoritmo duomenims (reikalavimų modeliui) yra:

- ✓ bent vienas sistemos aktorius, turintis pavadinimą;
- ✓ bent vienas interfeisas, turintis bent vieną operaciją;
- ✓ bent vienas ryšys tarp aktoriaus ir interfeiso;
- ✓ bent viena esybė.

2.11. Analizės išvados

1) Išanalizavus EMDA kūrimo metodą, pastebėta, kad jame siūloma transformacija iš reikalavimų į projekto modelį yra grindžiama tik vienu pasiūlytu architektūriniu šablonu, o būtų naudinga sukurti ir daugiau transformacijų variantų, suteikiančių projektuotojui galimybę rinktis pageidaujamą architektūrinį šabloną.

2) Atlikus IS kūrimo metodų analizę pastebėta, jog RUP metode aiškiai atskirti reikalavimų analizės ir projektavimo etapai, metodas pagrįstas užduotimis, per kurias išreiškiami reikalavimai; procese akcentuojama architektūra, todėl šis metodas pasirinktas taikymui transformacijoje.

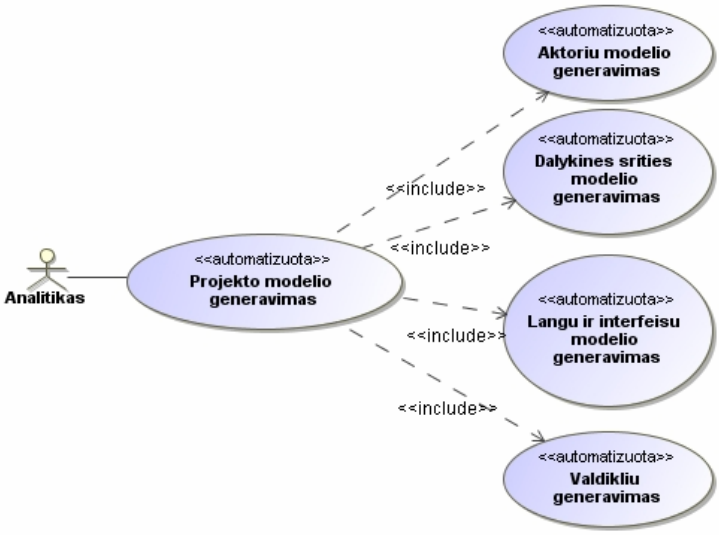
3) Išanalizavus *GoF* architektūrinius projektavimo šablonus pastebėta, jog RUP procese dažnai rekomenduojama taikyti MVC principus, todėl iš *GoF* šablonų detaliau analizuotas ir taikymui transformacijoje pasirinktas Stebėtojo šablonas bei konkretesnis jo taikymo pavyzdys – MVC šablonas, kuris atskiria vaizdavimo, duomenų ir veiklos logikos sluoksnius.

4) Atlikus UML CASE įrankių analizę pastebėta, jog MagicDraw įrankis turi daugiau galimybių ne tik UML diagramų modeliavimo, bet ir transformacijų srityje, todėl šis įrankis pasirinktas transformavimo iš reikalavimų į projektą algoritmo realizavimui.

3. Transformavimo iš reikalavimų modelio į projektą pagal MVC šabloną algoritmas

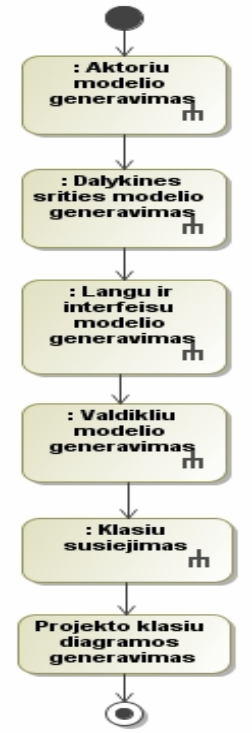
3.1. Transformacijos algoritmo panaudojimo atvejų specifikacijos

Transformavimo iš reikalavimų modelio į projektą algoritmas skirtas projekto modelio generavimui, kuris susideda iš: aktorių modelio generavimo; dalykinės srities modelio generavimo; langų ir interfeisų modelio generavimo; valdiklių generavimo (18 pav.).



18 pav. Transformacijos algoritmo panaudojimo atvejai

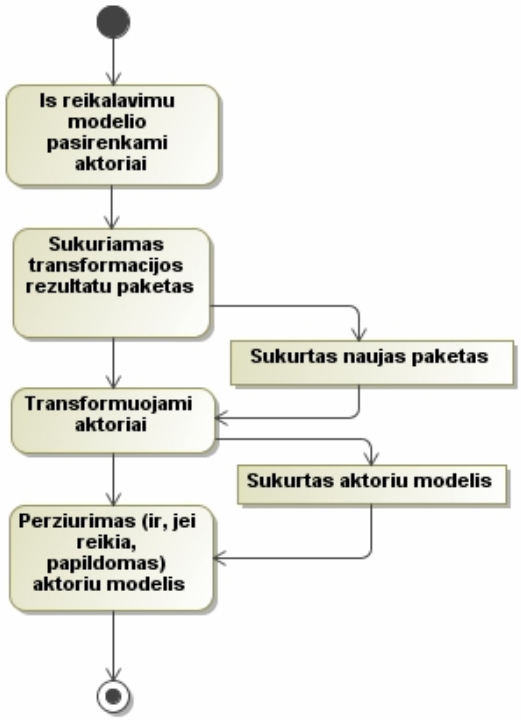
Panaudojimo atvejo **„Projekto generavimas“** veiklos diagrama vaizduojama 19 pav.



19 pav. Panaudojimo atvejo – „Projekto generavimas“ specifikacija

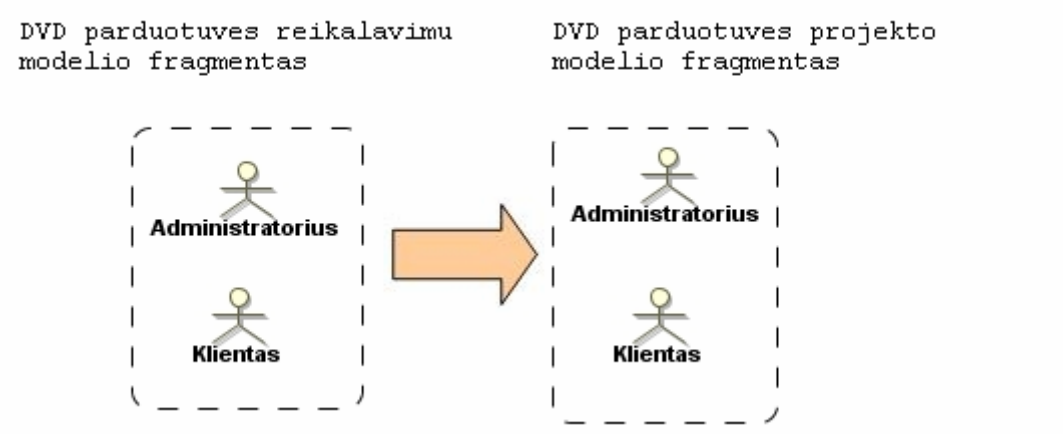
Transformavimo algoritmo taikymui naudojami fragmentai iš kuriamos sistemos – „DVD parduotuvė“.

Panaudojimo atvejis – **„Aktorių modelio generavimas“** (20 pav.): iš reikalavimų modelio pasirenkami aktoriai ir sukuriamas transformacijos rezultatų paketas (angl. *package*). Atliekama aktorių transformacija, kurios rezultatas – sukurtas projekto aktorių modelis. (Projekto aktorių modelis gali būti papildomas naujais aktoriais arba tik peržiūrimas).



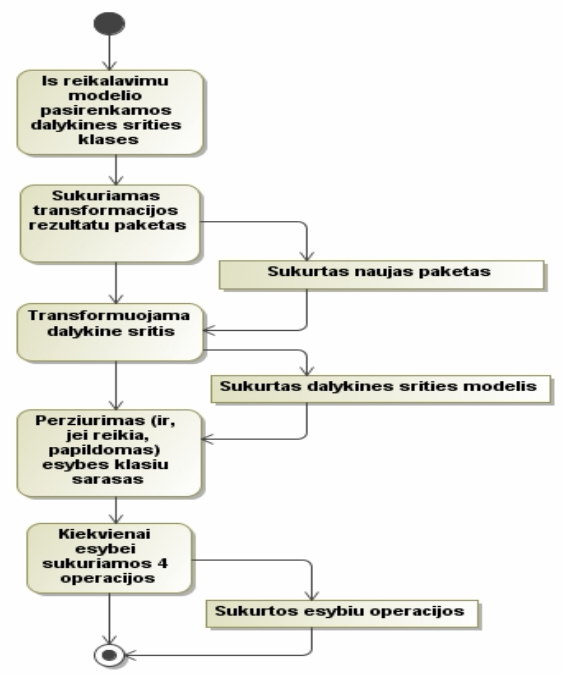
20 pav. Panaudojimo atvejo „Aktorių modelio generavimas“ specifikacija

„DVD parduotuvės“ reikalavimų modelio aktoriai transformuojami į projekto modelį ir išlieka nepakitę (21 pav.):



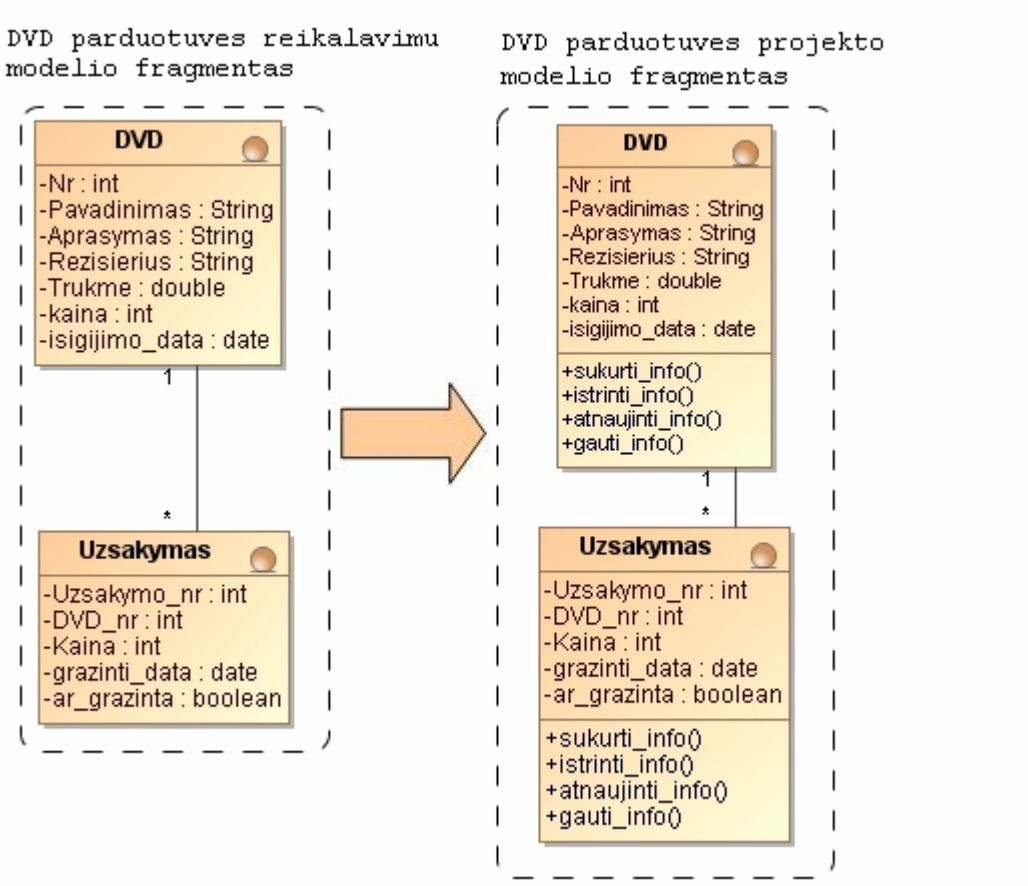
21 pav. Reikalavimų modelio aktorių transformavimo į projekto modelį pavyzdys

Panaudojimo atvejis – **„Dalykinės srities modelio generavimas“** (22 pav.): iš reikalavimų modelio pasirenkamos dalykinės srities klasės, sukuriamas transformacijos rezultatų paketas, reikalavimų modelio dalykinė sritis transformuojama į projektą, kiekvienai esybei kuriamos 4 operacijos. (Projekto dalykinės srities modelis gali būti papildomas naujomis esybėmis arba tik peržiūrimas).



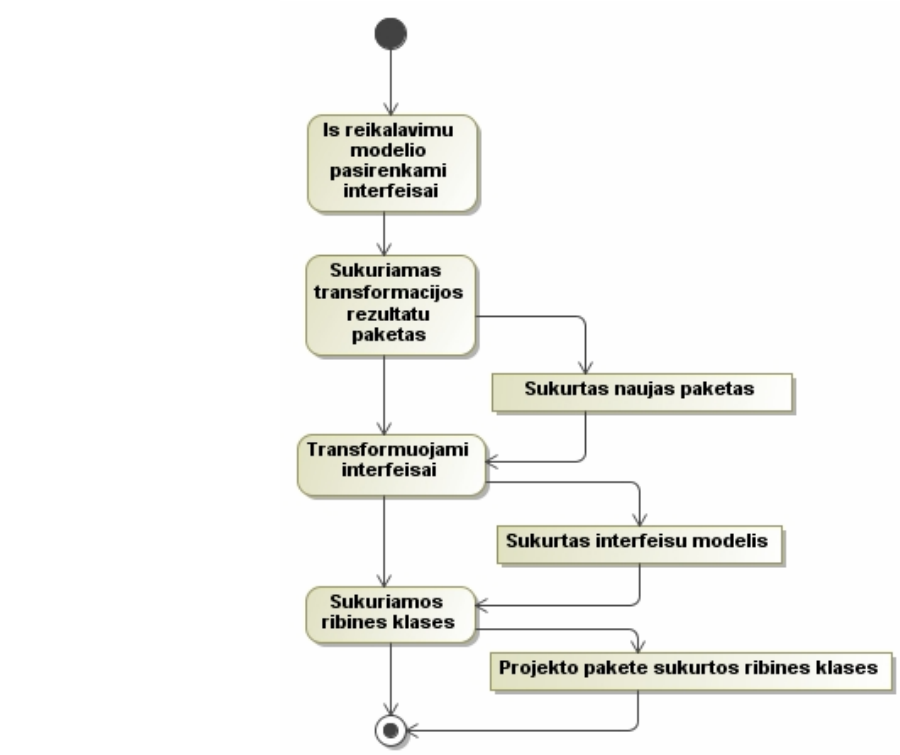
22 pav. Panaudojimo atvejo „Dalykinės srities modelio generavimas“ specifikacija

„DVD parduotuvės“ reikalavimų modelio dalykinė sritis transformuojama į projekto modelį, kiekvienai esybei sukuriant po 4 operacijas (23 pav.).



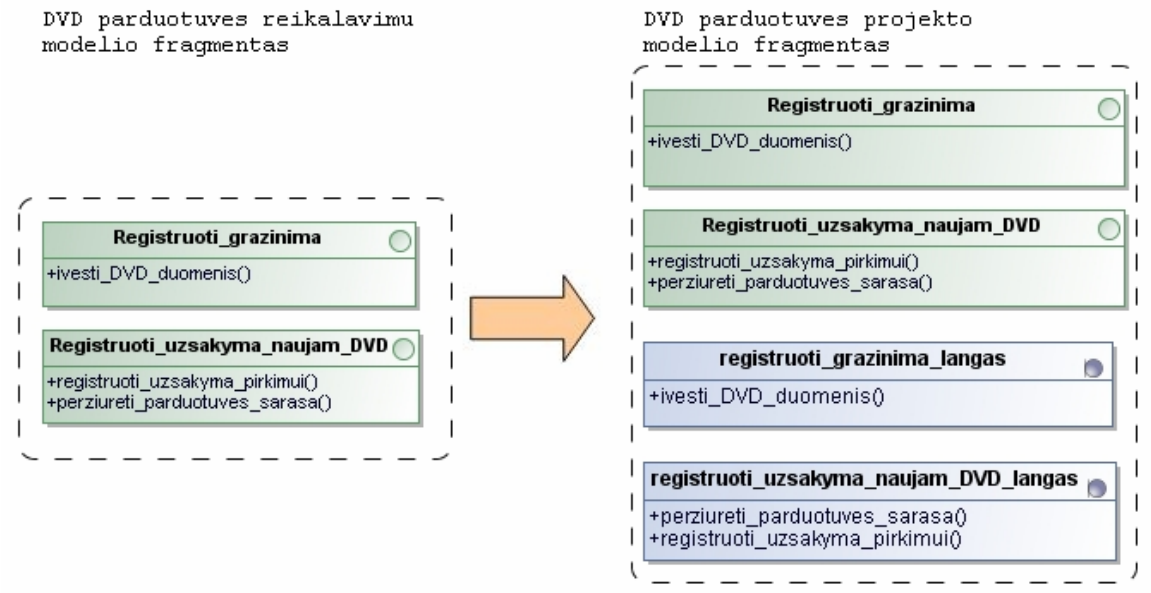
23 pav. Reikalavimų modelio dalykinės srities transformavimo į projekto modelį pavyzdys

Panaudojimo atvejis – **„Langų ir interfeisų modelio generavimas“** (24 pav.): iš reikalavimų modelio pasirenkami interfeisai ir sukuriamas transformacijos rezultatų paketas. Interfeisai transformuojami į projekto paketą, kiekvienam jų sukuriama ribinė klasė, kuri realizuoja konkretų interfeisą.



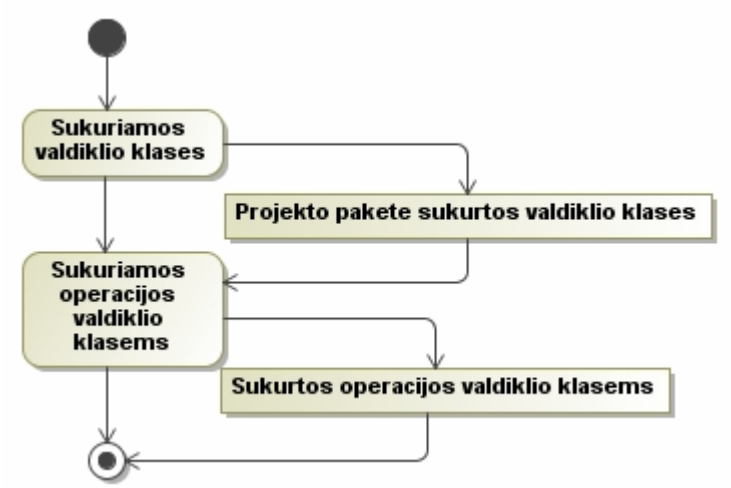
24 pav. Panaudojimo atvejo „Langų ir interfeisų modelio generavimas“ specifikacija

„DVD parduotuvės“ reikalavimų modelio interfeisai transformuojami į projekto modelį (25 pav.), kiekvienam interfeisui sukuriama ribinė klasė, realizuojanti interfeisą**.**



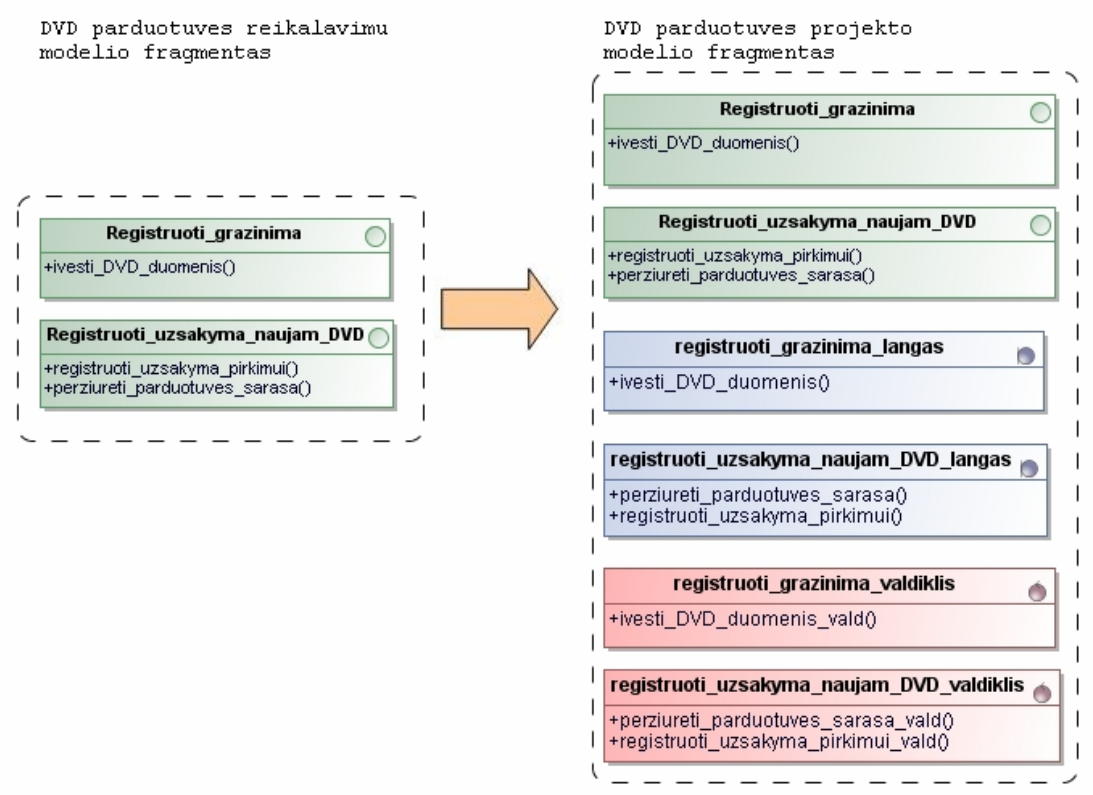
25 pav. Reikalavimų modelio interfeisų transformavimo į projekto modelį pavyzdys

Panaudojimo atvejis – **„Valdiklių modelio generavimas“** (26 pav.): kuriamos projekto modelio valdiklio klasės. Kiekvienai klasei, realizuojančiai konkretų interfeisą, kuriamos operacijos, kurių pavadinimai atitinka tokią struktūrą: interfeiso_operacijos_pavadinimas_*vald*.



26. pav. Panaudojimo atvejo „Valdiklių modelio generavimas“ specifikacija

Reikalavimų modelio interfeisus projekto modelyje realizuoja ribinės klasės ir valdikliai (27 pav.).



27 pav. Valdiklio klasių ir jų operacijų sukūrimo projekto modelyje pavyzdys

Algoritmo žingsnis **„Projekto klasių diagramos generavimas“:** automatiškai generuojama klasių diagrama, kurioje vaizduojami sistemos projekto modelio aktoriai, interfeisai, klasės ir ryšiai tarp jų. Sugeneruota klasių diagrama atvaizduojama vartotojui ir išsaugoma projekto pakete.

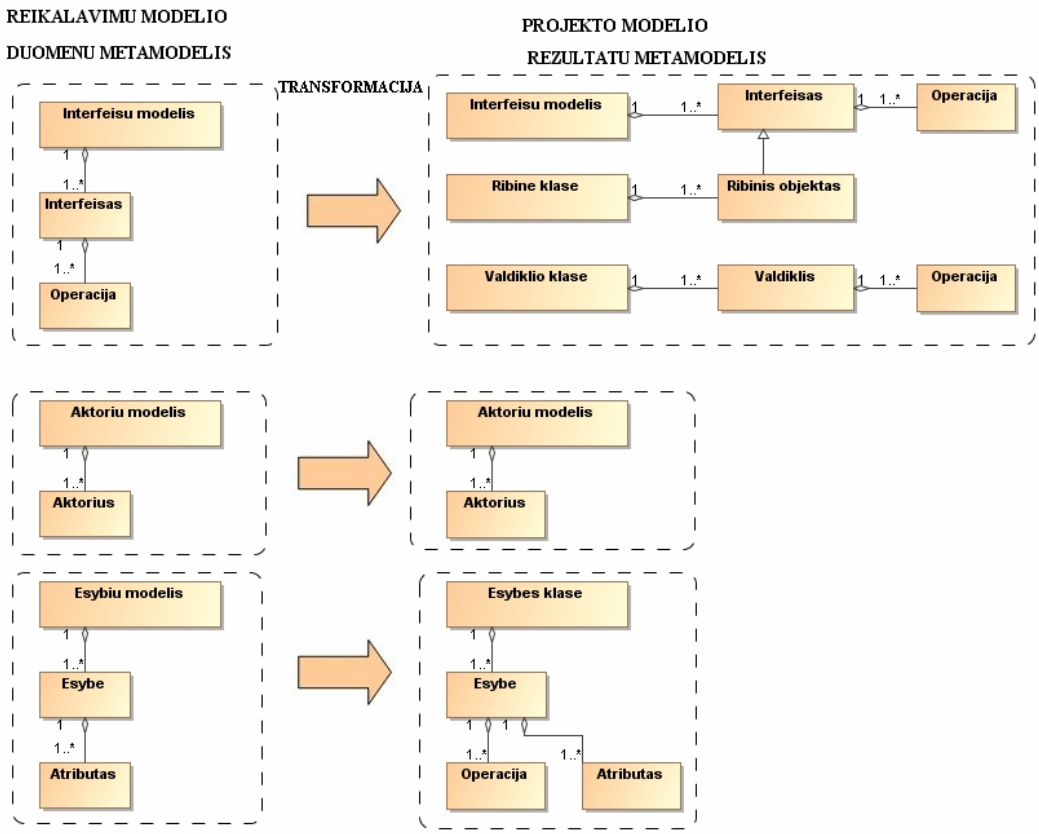
3.2. Transformacijos algoritmo duomenų ir rezultatų metamodelis

Reikalavimų modelį (28 pav.) sudaro:

- interfeisų modelis, jame esantis bent vienas interfeisas, turintis bent vieną operaciją;
- aktorių modelis, kuriame yra bent vienas aktorius;
- esybių modelis, kuriame yra bent viena esybė, turinti bent vieną atributą.

Projekto modelį (28 pav.) sudaro:

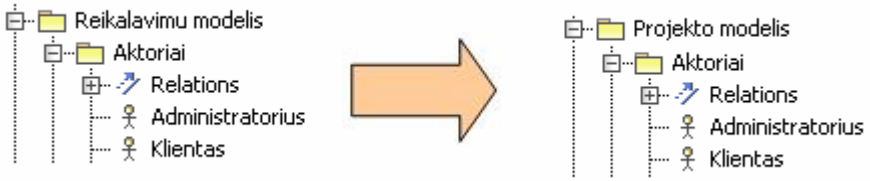
- valdymo ir ribinių klasių modelis, kurį sudaro:
    - interfeisai;
    - ribinės klasės;
    - valdiklio klasės;
- aktorių modelis;
- esybių modelis.



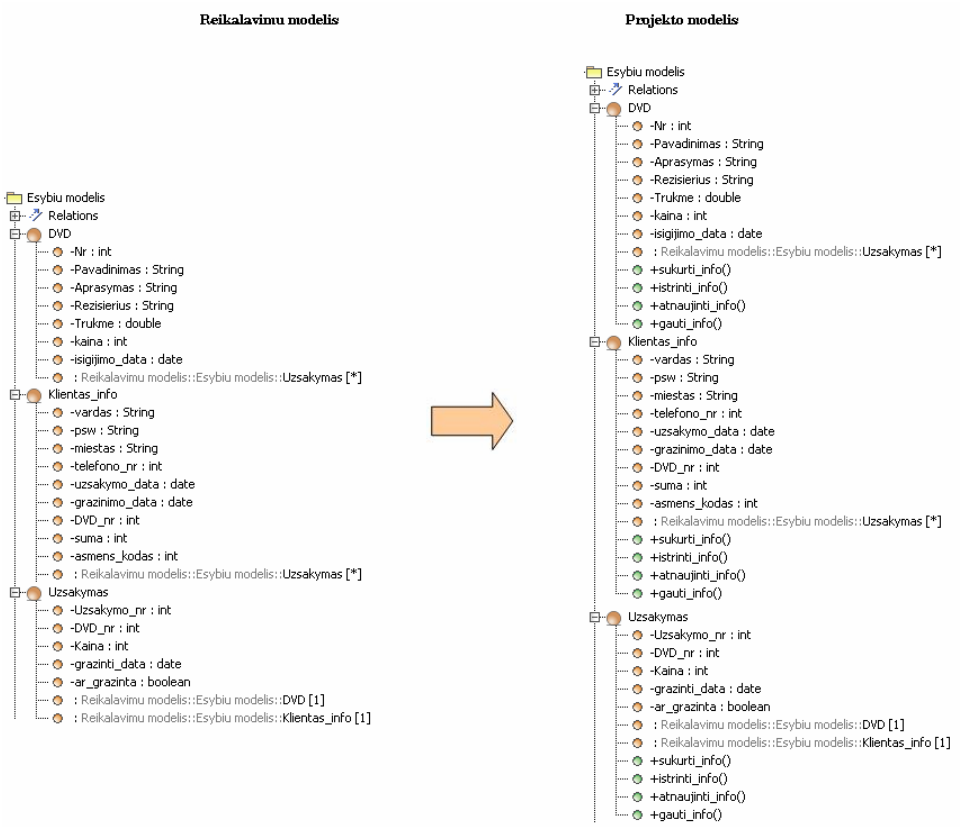
28 pav. Algoritmo duomenų ir rezultatų metamodelis

Po transformacijos interfeisų, valdiklių, langų, aktorių ir esybių modeliai (vaizduojami 29 – 31 pav.) projekto etape atitinkamai pasipildo algoritmo metu sukurtais naujais elementais.

29 pav. Interfeisų, valdiklių ir langų modeliai projekto etape (DVD parduotuvės pavyzdys)



30 pav. Aktoriu modeliai reikalavimų ir projekto etapuose (DVD parduotuvės pavyzdys)



31 pav. Dalykinės srities modelis reikalavimų ir projekto etapuose (DVD parduotuvės pavyzdys)

4. Realizacijos modelis

4.1. MagicDraw priemonės

Kuriant transformavimo iš informacinės sistemos reikalavimų modelio į projekto modelį, naudota MagicDraw siūloma priemonė – klasių diagramos vedlys (angl. *Class Diagram Wizard*). Šio vedlio pagrindu sukurtas naujas vedlys „*Transformation from DIM to PIM*“, realizuojantis transformavimo iš IS reikalavimų modelio į projekto modelį algoritmą. Vedlys sukuria naują klasių diagramą, kuomet visos klasės ir ryšiai tarp jų yra jau sukurti ir nurodyti (angl. *speficied*). Vedlio pagalba galima pasirinkti kurios klasės, paketai ir tarpusavio ryšiai bus įtraukti į naują klasių diagramą. Ši priemonė leidžia išvengti klasių diagramos kūrimo rankiniu būdu, taip automatizuojant diagramos (ir jos elementų) kūrimą. Vedlys per keletą žingsnių surenka informaciją iš sistemos projektuotojo ir automatiškai sugeneruoja klasių diagramą su projektuotojo nurodytais elementais ir ryšiais tarp jų.

Taip pat panaudota modelių transformavimo priemonę „*AnytoAny*“, kuri transformavimo metu kopijuoja sistemos elementus: kopija yra identiška originalui, o transformuoti elementai virsta savarankiškais.

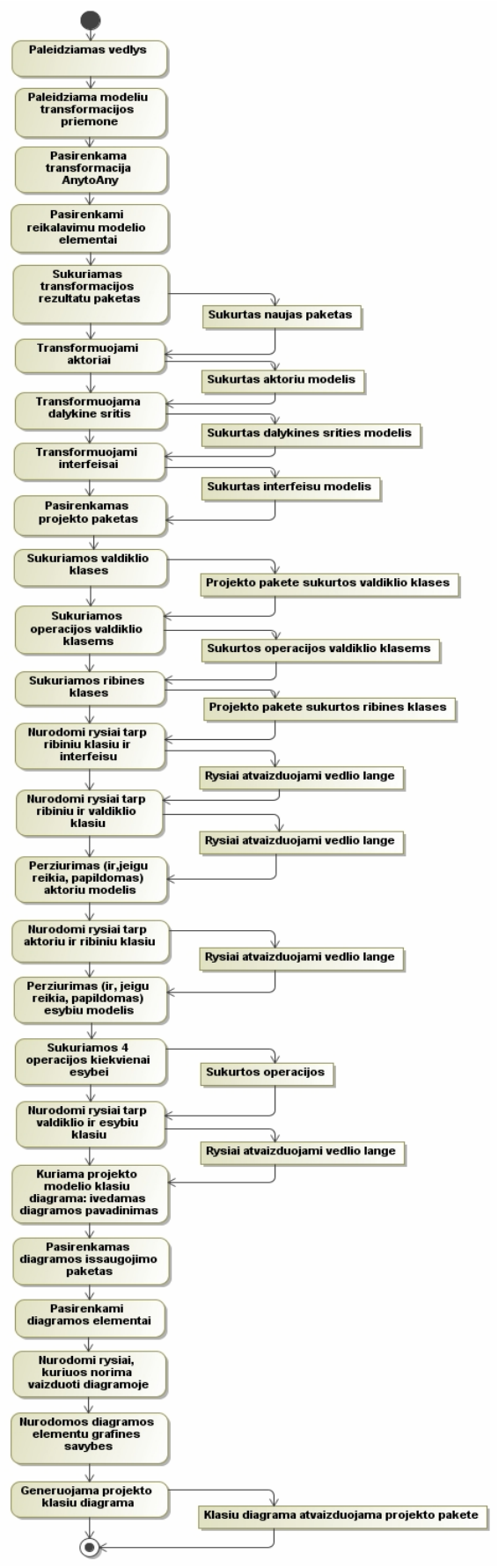
4.2. Transformacija, naudojant MagicDraw priemones

Atliekant transformaciją iš informacinės sistemos reikalavimų modelio į projekto modelį, atliekama ši veiksmų seka (32 pav.):

1) paleidžiamas vedlys (angl. *Wizard*);
2) paleidžiama modelių transformacijos (angl. *Model Transformations*) priemonė;
3) pasirenkama transformacija *AnytoAny*;
4) iš reikalavimų modelio pasirenkami transformuojamieji elementai (aktorių, dalykinės srities ir interfeisų modeliai);
5) sukuriamas transformacijos rezultatų paketas;
6) reikalavimų modelio aktoriai transformuojami į projektą;
7) reikalavimų modelio dalykinė sritis transformuojama į projektą;
8) reikalavimų modelio interfeisai transformuojami į projektą;
9) projekto pakete sukuriamos valdiklio klasės;
10) kiekvienai valdiklio klasei sukuriamos operacijos;
11) projekto pakete sukuriamos ribinės klasės;
12) nurodomi ryšiai tarp ribinių klasių ir interfeisų;
13) nurodomi ryšiai tarp ribinių ir valdiklio klasių;
14) peržiūrimas (ir, jeigu reikia, papildomas) aktorių modelis;
15) nurodomi ryšiai tarp aktorių ir ribinių klasių;
16) peržiūrimas (ir, jeigu reikia, papildomas) esybių modelis;
17) kiekvienai esybės klasei sukuriamos 4 operacijos;
18) nurodomi ryšiai tarp valdiklio ir esybių klasių;

Kuriama projekto modelio klasių diagrama:

1) įvedamas klasių diagramos pavadinimas;
2) pasirenkamas diagramos išsaugojimo paketas;
3) pasirenkami diagramos elementai;
4) nurodomi ryšiai, kuriuos norima vaizduoti diagramoje;
5) nurodomos diagramos elementų grafinės savybės;
6) automatiškai generuojama projekto klasių diagrama.

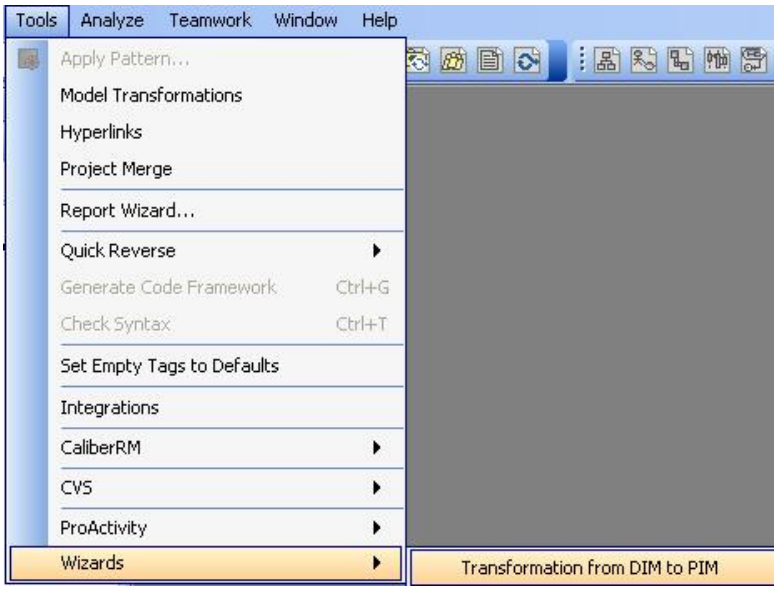


32 pav. Transformavimo algoritmo, realizuoto naudojant MagicDraw vedlį, veikimą aprašanti veiklos diagrama

5. Eksperimentinis transformavimo algoritmo tyrimas

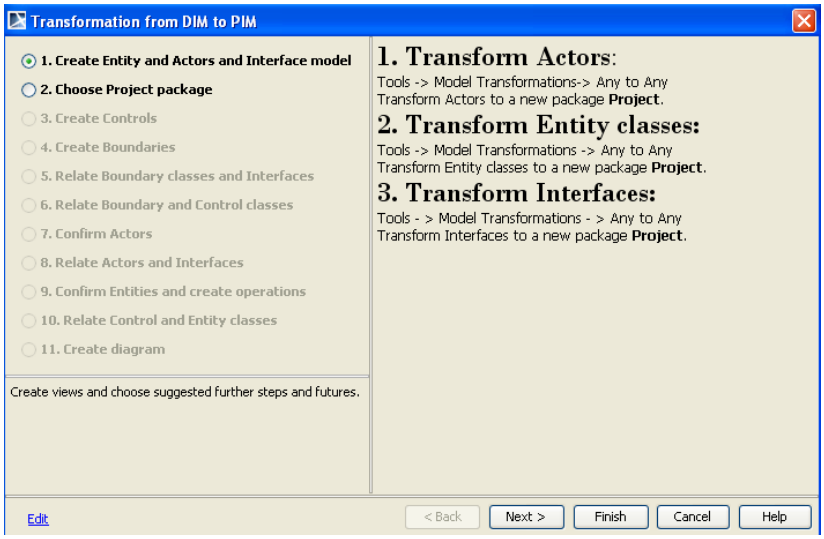
5.1. Sistemos naudojimo instrukcija

Vedlys „*Transformation from DIM to PIM*“ paleidžiamas MagicDraw programos kontekstiniame meniu pasirinkus *Tools/Wizards/Transformation from DIM to PIM* (33 pav.).



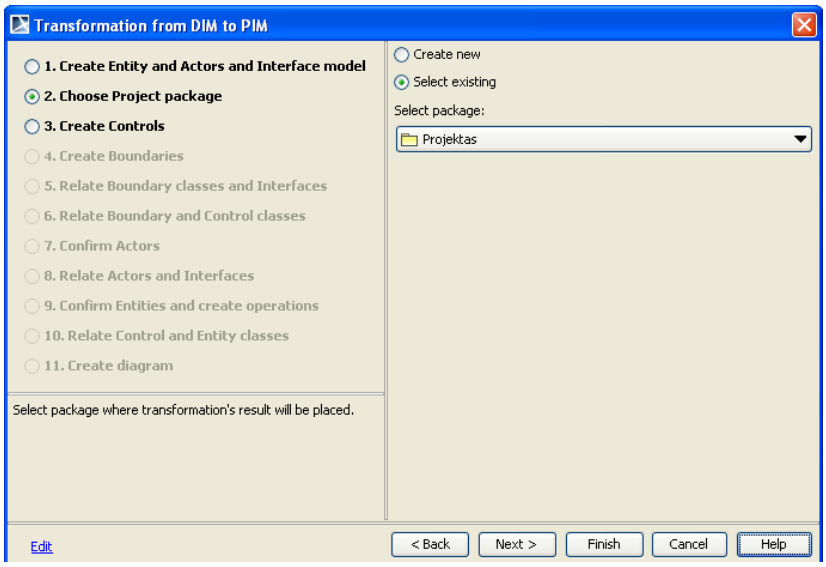
33 pav. Vedlio paleidimo langas

Pereinant nuo reikalavimų prie projektavimo etapo, reikalavimų modelio *aktoriai*, *dalykinė sritis* ir *interfeisai* perkeliami į projekto paketą (34 pav.). Tam naudojamas modelių transformavimas (kopijuojant elementus, kopija yra identiška originalui, o transformuotas elementas virsta savarankišku).



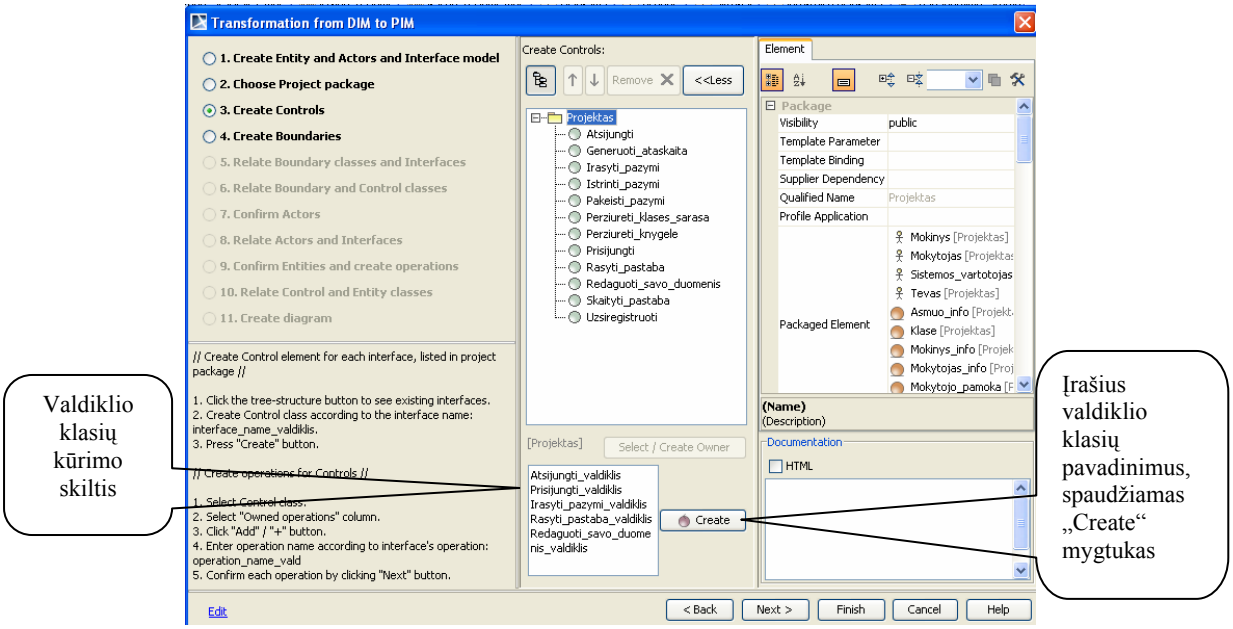
34 pav. Aktorių, dalykinės srities ir interfeisų modelių transformacijos langas

Pasirenkamas projekto paketas (35 pav.):



35. pav. Projekto paketo pasirinkimo langas

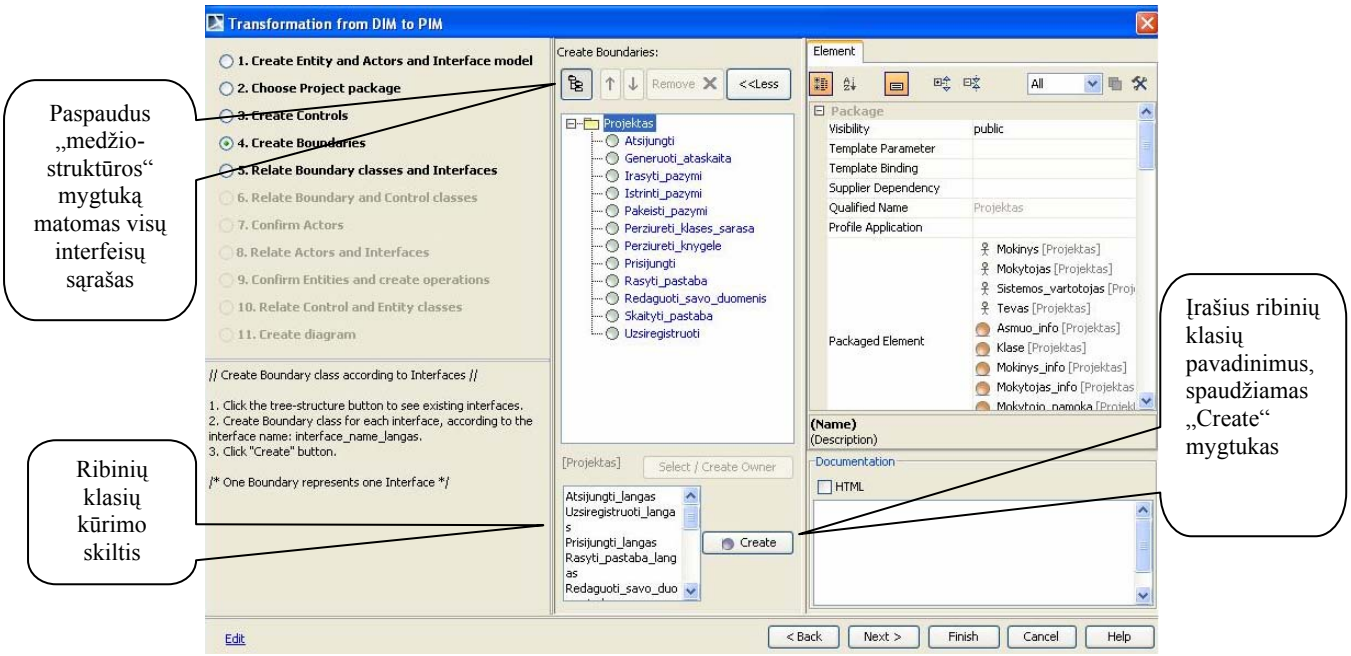
Trečiasis vedlio žingsnis – valdiklio (angl. *Control*) klasių sukūrimas. Kiekvienam projekto interfeisui kuriama po vieną valdiklio klasę. Kuriamos valdiklio klasės pavadinimas turi sutapti su interfeiso pavadinimu, prie jo pridedant galūnę „valdiklis“ (36 pav.). Valdiklio klasių pavadinimai rašomi žemiau interfeisų sąrašo esančiame lange, atskiriant juos „enter“ paspaudimu. Surašius valdiklių pavadinimus, spaudžiamas „Create“ mygtukas.



36 pav. Valdiklio klasių kūrimo langas

Ketvirtajame vedlio žingsnyje kuriamos ribinės (angl. *Boundary*) klasės - kiekvienam projekto interfeisui sukuriama po vieną ribinę klasę (37 pav.). Kuriamos ribinės klasės pavadinimas turi sutapti su interfeiso pavadinimu, prie jo pridedant galūnę „langas“.

Valdiklio klasių pavadinimai rašomi žemiau interfeisų sąrašo esančiame lange, atskiriant juos „enter“ paspaudimu. Surašius valdiklių pavadinimus, spaudžiamas „Create“ mygtukas (37 pav.).

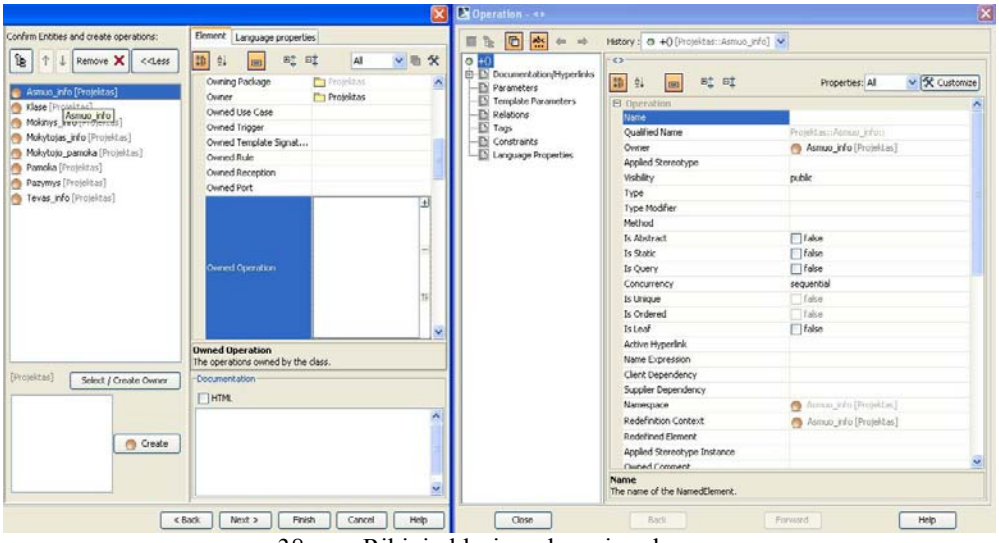


37 pav. Ribinių klasių kūrimo langas

Kiekvienai esybės klasei sukuriama po 4 operacijas:

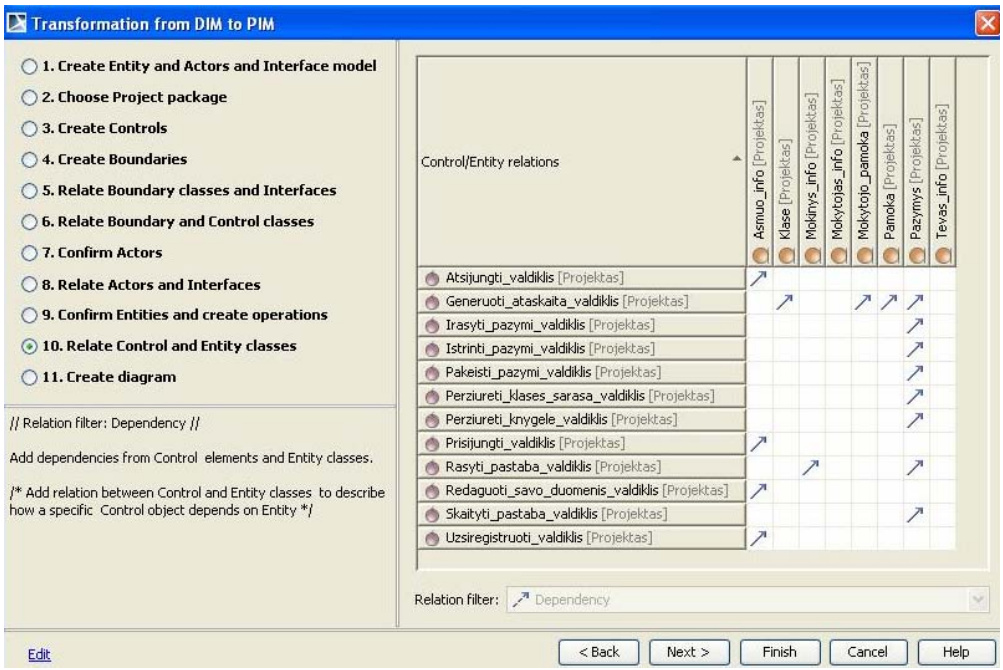
1) sukurti_info();
2) gauti_info();
3) ištrinti_info();
4) atnaujinti_info()

Tai atliekama klasių redagavimo meniu skiltyje „Owned operations“ įvedant operacijos pavadinimą (38 pav.).



38 pav. Ribinių klasių redagavimo langas

Nurodomi ryšiai tarp klasių, interfeisų ir aktorių (39 pav.)



39 pav. Ryšių tarp valdiklio klasių ir esybių nurodymo langas

Kuriama projekto modelio elementų klasių diagrama.

Pilna vartotojo instrukcija pateikiama priede

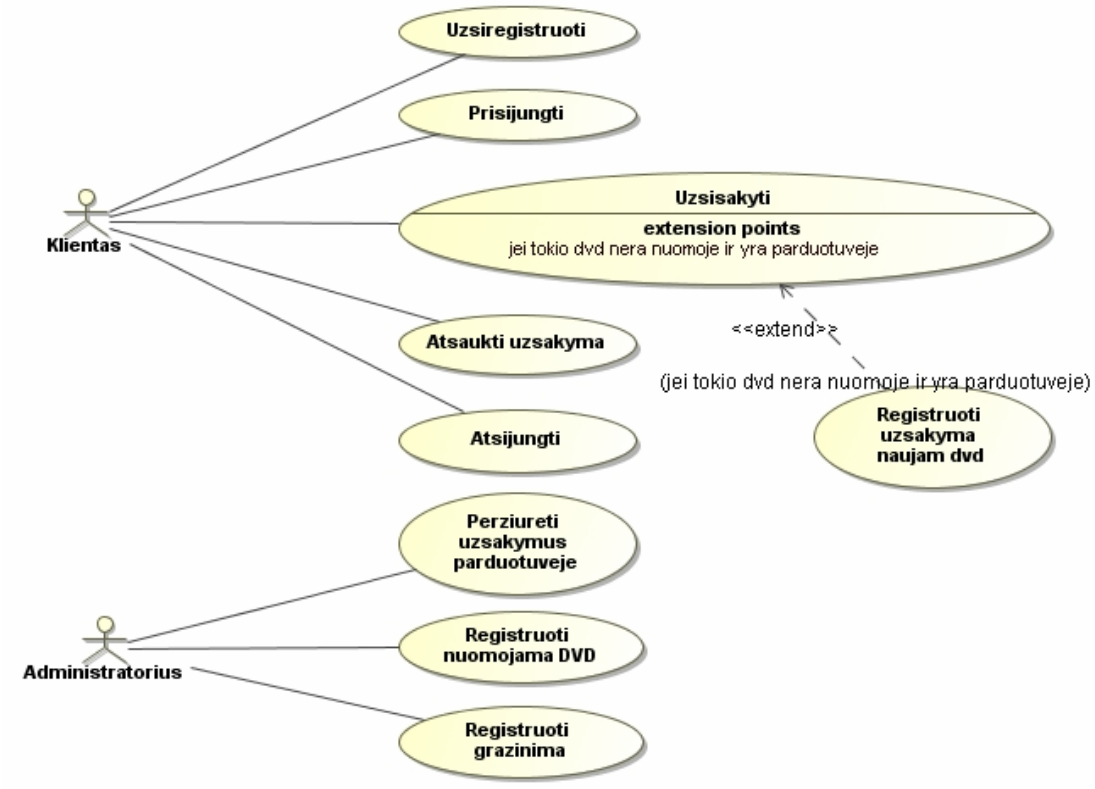
5.2. Transformavimo algoritmo taikymas DVD parduotuvės projekto modelio sudaryme

Pirmiausia sukurtas algoritmas išbandytas rankiniu būdu, projektuojant skaitmeninių video diskų (angl. *DVD*) nuomos sistemą. Sistemoje dalyvaujantys aktoriai:

- *Klientas* – sistemos vartotojas, galintis užsiregistruoti sistemoje, prisijungti, peržiūrėti nuomojamų DVD sąrašą, užsisakyti diską, atšaukti jo užsakymą, atsijungti nuo sistemos.
- *Administratorius* – sistemos dalyvis, galintis peržiūrėti klientų užsakymus sistemoje, registruoti naują nuomojamą DVD arba jo grąžinimą (40 pav.).

5.2.1. Reikalavimų (duomenų) modelis

Sistemos panaudojimo atvejai



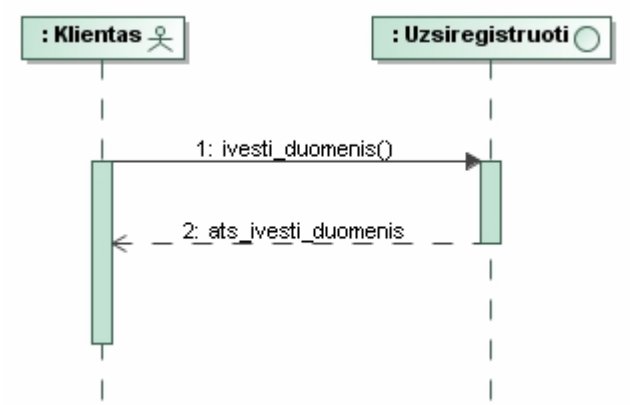
40 pav. Sistemos panaudojimo atvejų diagrama

Reikalavimų etape kiekvienam panaudojimo atvejui sukurta sekos diagrama, vaizduojanti ryšį tarp sistemos aktoriaus ir kiekvieno panaudojimo atvejo.

Reikalavimų etapo sekos diagramos

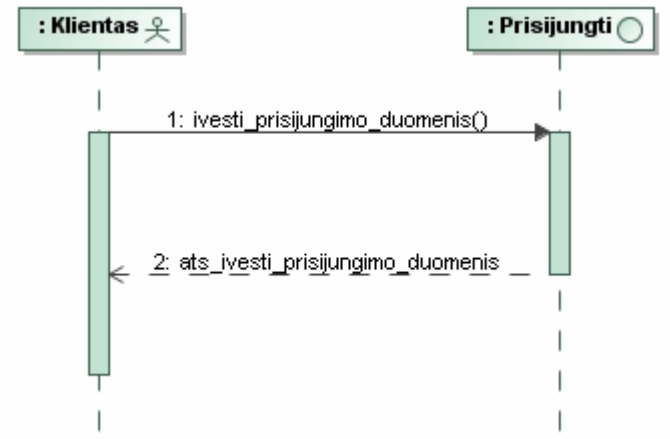
Žemiau pateiktose diagramose vaizduojami aktoriaus „Klientas“ kuriami pranešimai kiekvienam panaudojimo atvejui bei panaudojimo atvejo kuriamas atsakymo pranešimas aktoriui.

Klientas, norėdamas užsiregistruoti sistemoje, įveda savo duomenis. Interfeisas kuria atsakymo pranešimą klientui apie įvestus duomenis (41 pav.).



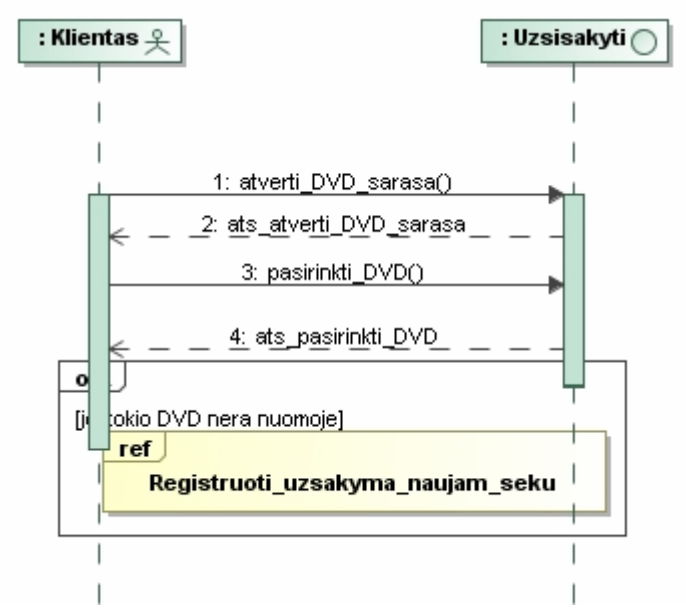
41 pav. Sekos diagrama panaudojimo atvejui „Užsiregistruoti“

Prisijungimo prie sistemos metu klientas įveda savo prisijungimo duomenis. Interfeisas kuria atsakymo pranešimą klientui apie įvestus duomenis (42 pav.).

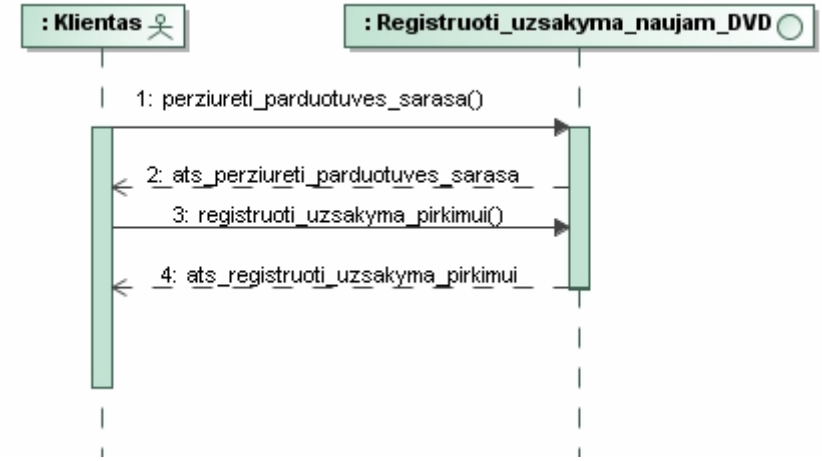


42 pav. Sekos diagrama panaudojimo atvejui „Prisijungti“

Klientas, norėdamas užsisakyti pageidaujamą DVD, atveria jų sąrašą. Interfeisas klientui pateikia nuomojamų DVD sąrašą. Klientas pasirenką konkretų DVD, interfeisas kuria atsakymo pranešimą klientui apie pasirinktą prekę. (43 pav.) Jei tokio DVD nuomoje nėra – klientas registruoja užsakymą naujai (pageidaujamai) prekei (44 pav.).

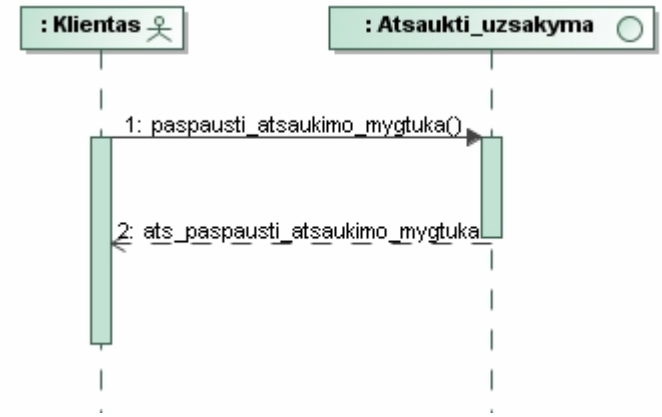


43 pav. Sekos diagrama panaudojimo atvejui „Užsisakyti“



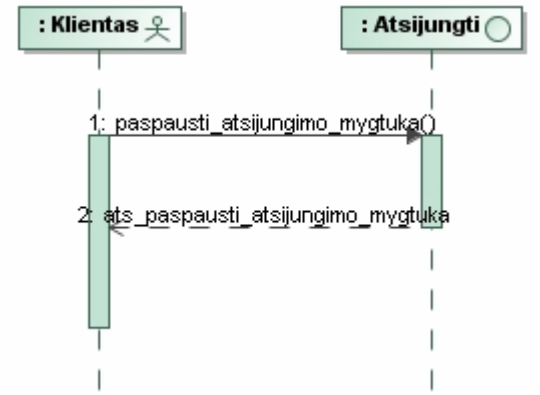
44 pav. Sekos diagrama panaudojimo atvejui „Registruoti užsakymą naujam DVD“

Klientas, norėdamas atšaukti savo užsakymą, paspaudžia atšaukimo mygtuką. Interfeisas grąžina atsakymo pranešimą (45 pav.).



45 pav. Sekos diagrama panaudojimo atvejui „Atšaukti užsakymą“

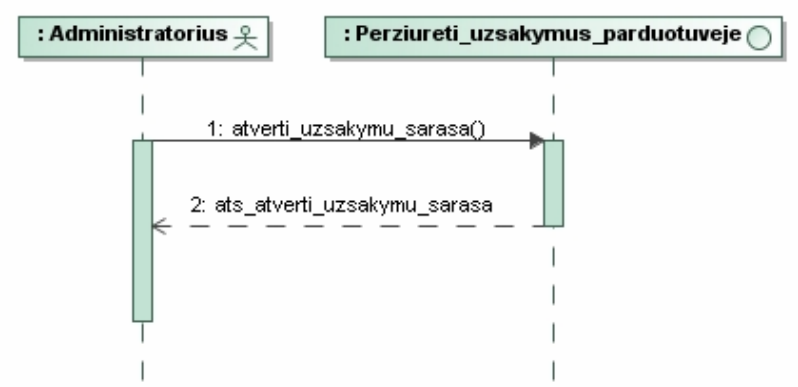
Baigęs darbą su sistema, klientas atsijungia – paspaudžia atsijungimo mygtuką (46 pav.).



46 pav. Sekos diagrama panaudojimo atvejui „Atsijungti“

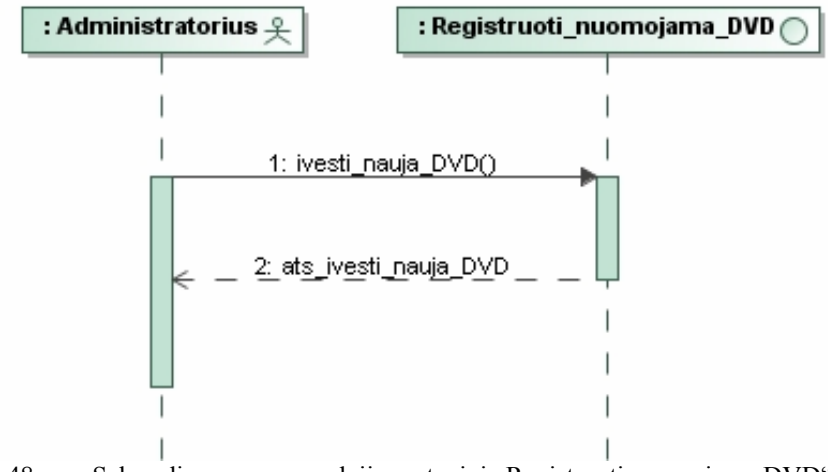
Kitas sistemos vartotojas – „Administratorius“. Sekos diagramose vaizduojami aktoriaus kuriami pranešimai kiekvienam panaudojimo atvejui bei panaudojimo atvejo kuriamas atsakymo pranešimai aktoriui.

Administratorius, norėdamas peržiūrėti klientų užsakymus DVD parduotuvėje, atveria užsakymų sąrašą (47 pav.).



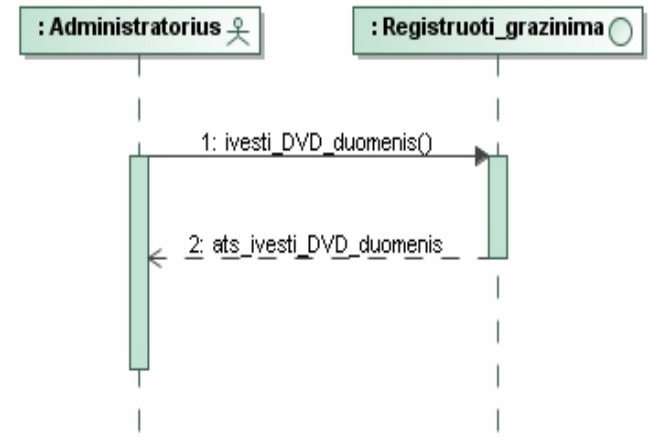
47 pav. Sekos diagrama panaudojimo atvejui „Peržiūrėti užsakymus parduotuvėje“

Norėdamas užregistruoti nuomojamą DVD, administratorius sistemoje įveda prekės duomenis (48 pav.).



48 pav. Sekos diagrama panaudojimo atvejui „Registruoti nuomojamą DVD“

Klientui grąžinus prekę, administratorius sistemoje užregistruoja grąžinimą, įvesdamas DVD duomenis.



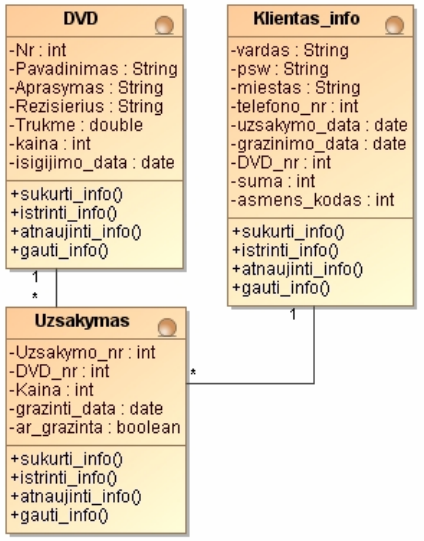
49 pav. Sekos diagrama panaudojimo atvejui „Registruoti grąžinimą“

Esybių modelis

Sistemoje sukurtos trys esybės – *DV*D, *Klientas_info* ir *Užsakymas* (50 pav.).

Kiekvienai esybei sukurtos 4 operacijos:

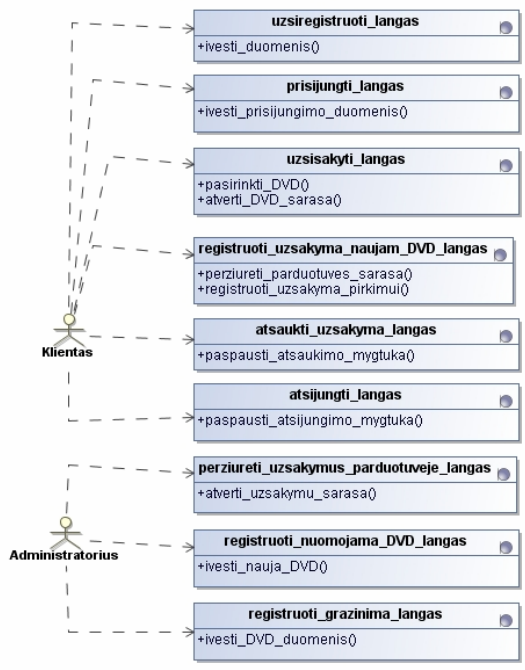
1) sukurti_info;
2) ištrinti_info;
3) atnaujinti_info;
4) gauti_info.



50 pav. Esybių modelio diagrama

Interfeisų modelis

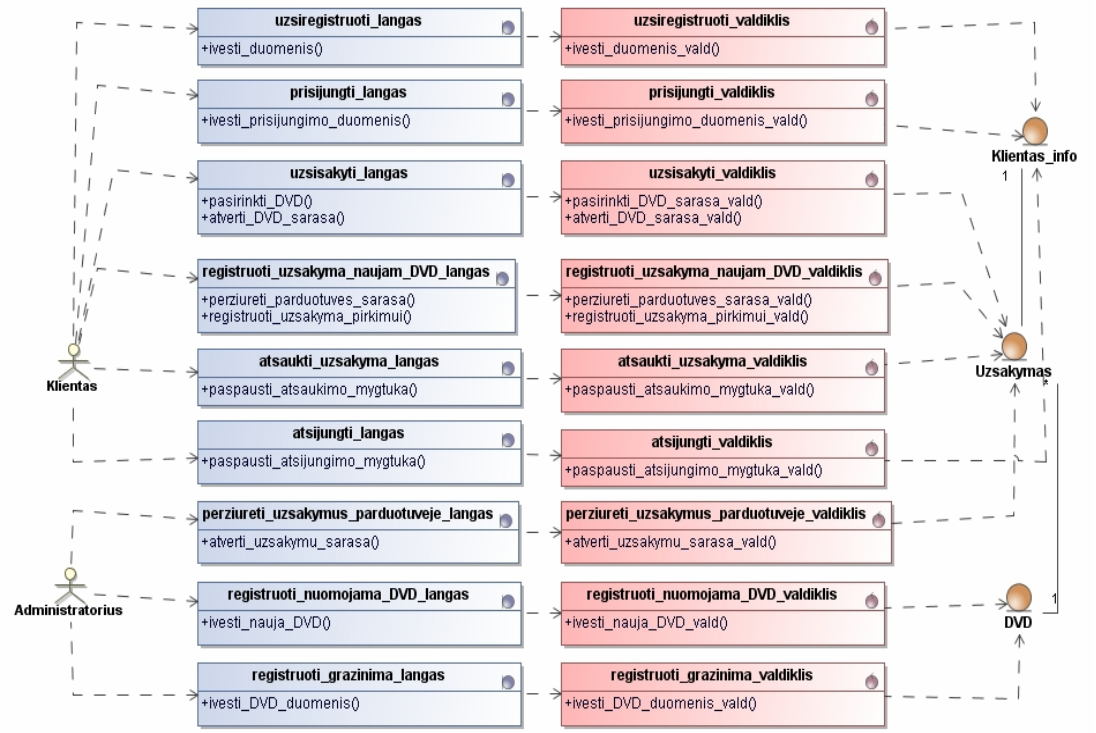
Sistemoje sukurti kiekvieną panaudojimo atvejį realizuojantys interfeisai ir jų operacijos (51 pav.).



51 pav. Interfeisų modelis

5.2.2. Projekto (transformavimo rezultatų) modelis

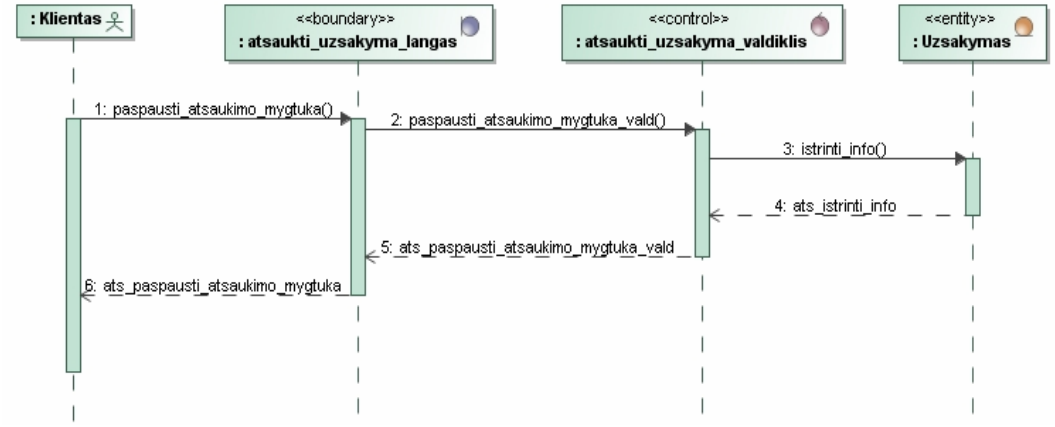
Projekto modelyje suprojektuojama klasių diagrama (52 pav.), kurioje vaizduojami aktoriai, ribinės klasės, valdikliai, esybės ir ryšiai tarp jų.



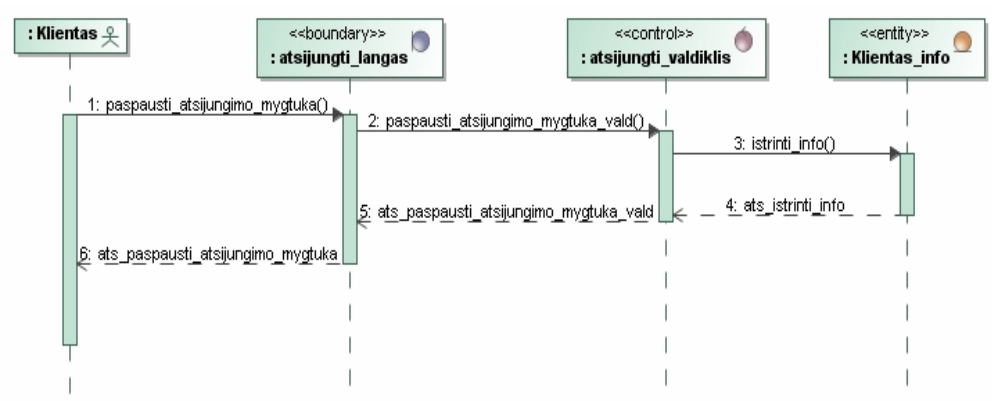
52. pav. Projekto modelyje suprojektuota klasių diagrama

Projekto modelyje kiekvienai reikalavimų etapo diagramai suprojektuojama po vieną sekos diagramą MVC šablono principu.

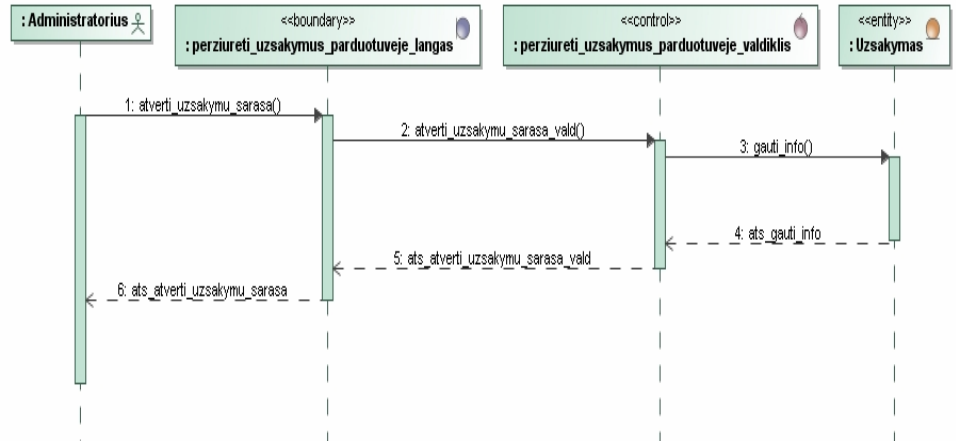
Gautos sekų diagramos vaizduojamos žemiau pateiktuose paveiksluose:



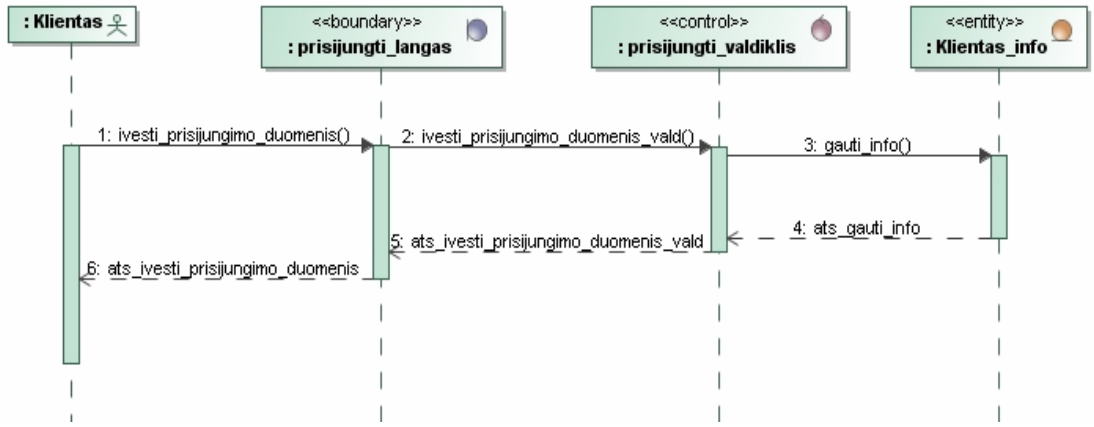
53 pav. „Atšaukti užsakymą“ sekos diagrama, suprojektuota MVC šablono principu



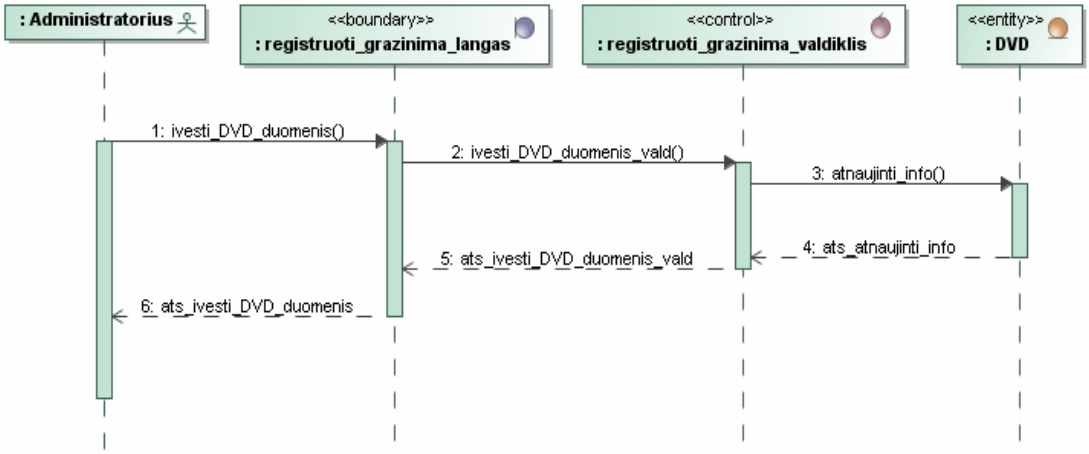
54 pav. „Atsijungti“ sekos diagrama, suprojektuota MVC šablono principu



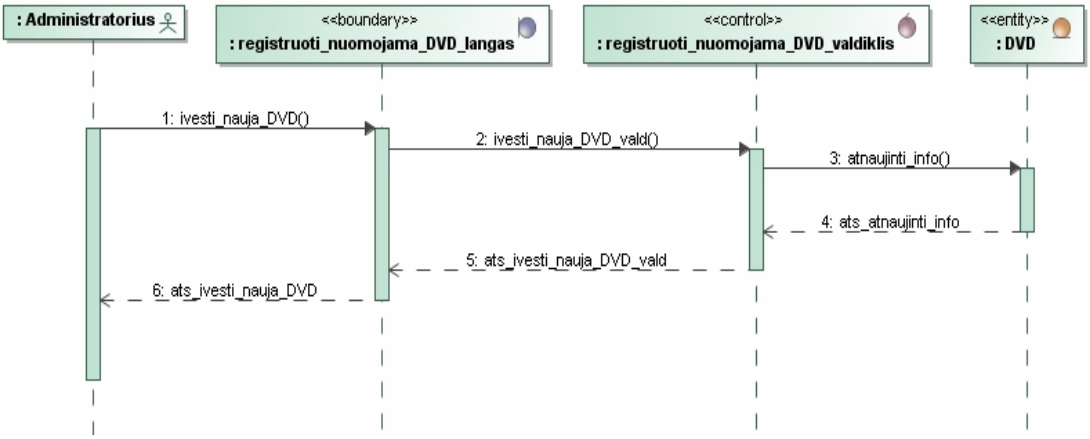
55 pav. „Peržiūrėti užsakymus parduotuvėje“ sekos diagrama, suprojektuota MVC šablono principu



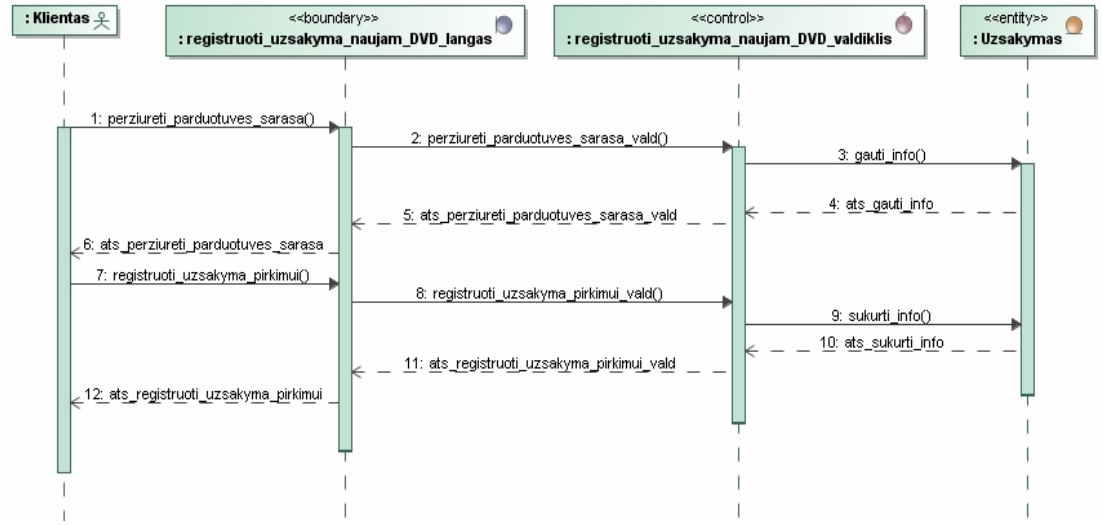
56 pav. „Prisijungti“ sekos diagrama, suprojektuota MVC šablono principu



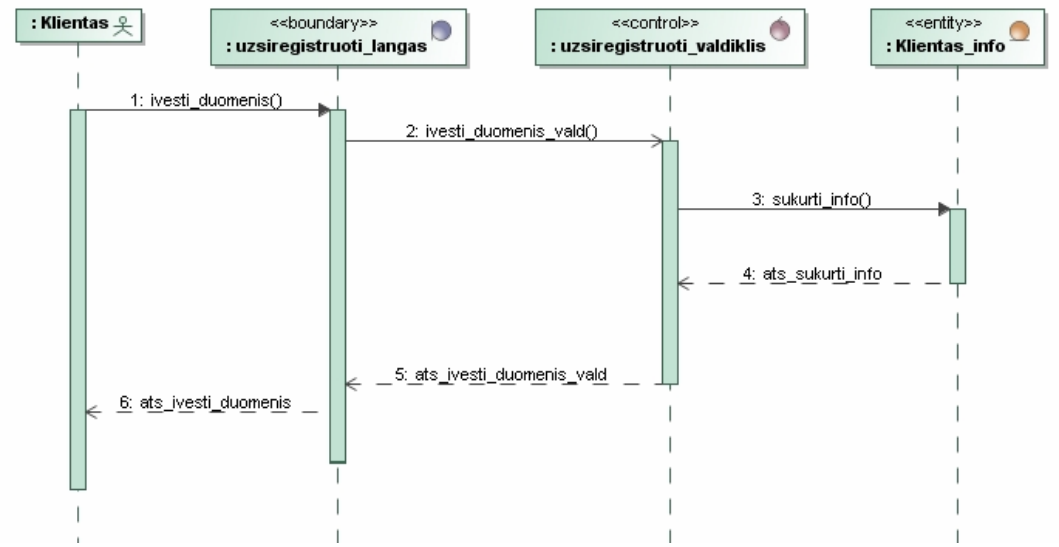
57 pav. „Registruoti grąžinimą“ sekos diagrama, suprojektuota MVC šablono principu



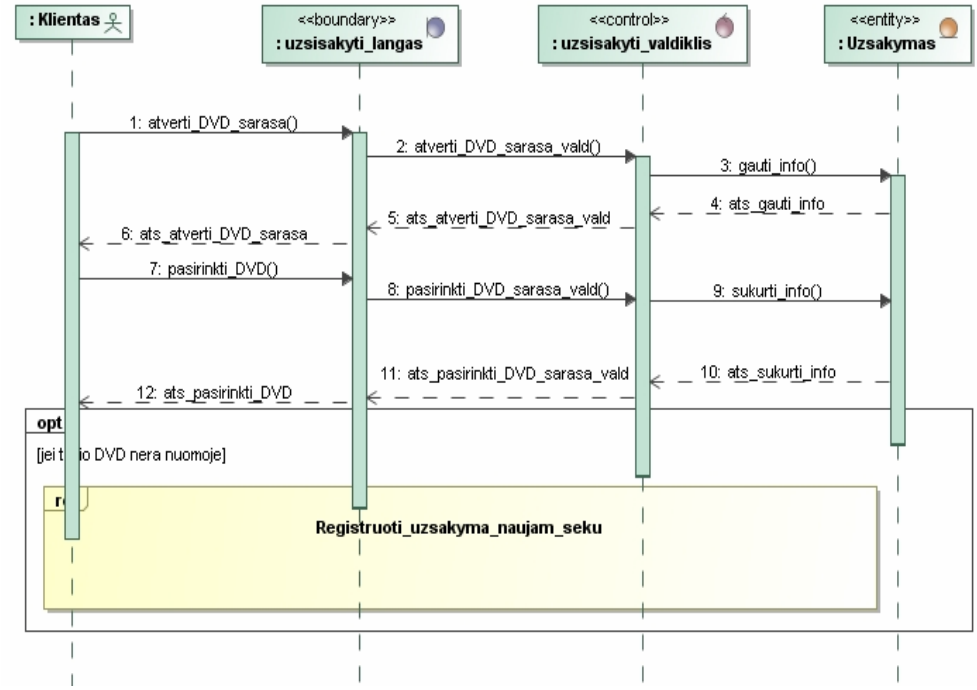
58 pav. „Registruoti nuomojamą DVD“ sekos diagrama, suprojektuota MVC šablono principu



59 pav. „Registruoti užsakymą naujam DVD“ sekos diagrama, suprojektuota MVC šablono principu



60 pav. „Užsiregistruoti“ sekos diagrama, suprojektuota MVC šablono principu



61. pav. „Užsisakyti“ sekos diagrama, suprojektuota MVC šablono principu

Iš interfeisų diagramos gaunama valdymo ir ribinių klasių diagrama (vaizduojama 62 pav.)

62 pav. Valdymo ir ribinių klasių diagrama

5.3. Transformavimo algoritmo taikymas elektroninio dienyno projekto modelio sudaryme

Realizuoto algoritmo išbandymui naudotas elektroninio mokyklos dienyno pavyzdys. Sistemoje egzistuoja 4 vartotojai, turintys skirtingas teises matyti sistemos duomenis bei juos redaguoti (63 pav.).

Sistemos_vartotojas – aktorius, galintis užsiregistruoti sistemoje, prisijungti prie jos, redaguoti savo duomenis ir atsijungti nuo sistemos.

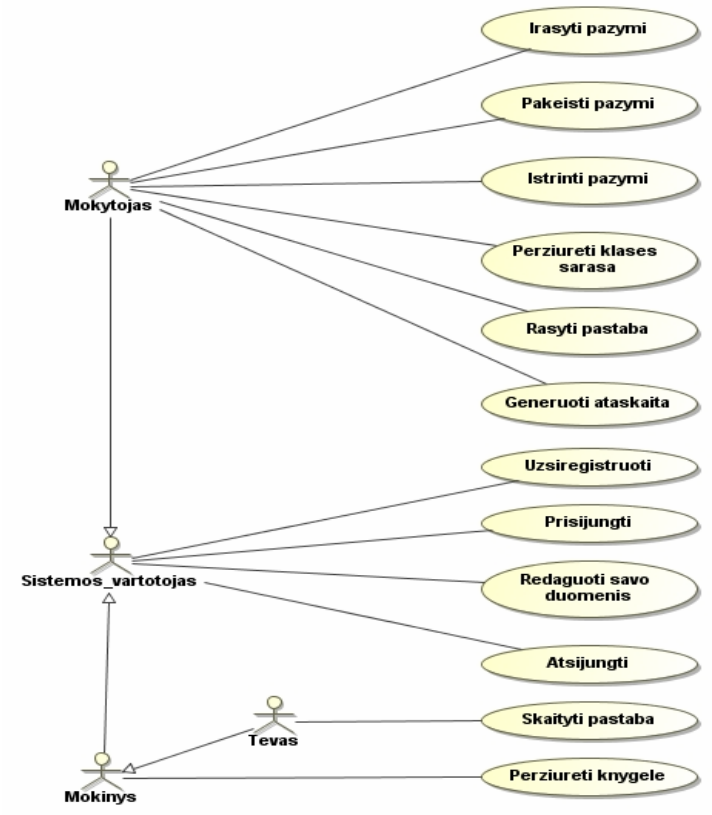
Mokytojas – sistemos vartotojas, galintis įrašyti pažymį, jį pakeisti kitu, ištrinti, peržiūrėti klasės sąrąšą, įrašyti pastabą ir generuoti ataskaitą. (Aktorius *Mokytojas* paveldi visus veiksmus iš aktoriaus *Sistemos_vartotojas*).

Mokinys – sistemos vartotojas, galintis peržiūrėti savo pažymių knygelę ir atlikti veiksmus, kuriuos atlieka aktorius *Sistemos_vartotojas*.

Tėvas – sistemos vartotojas (mokinio tėvas), galintis skaityti pastabą ir atlikti visus veiksmus, kuriuos atlieka aktoriai *Sistemos_vartotojas* ir *Mokinys*.

5.3.1. Reikalavimų (duomenų) modelis

Sistemos panaudojimo atvejai



63 pav. Sistemos panaudojimo atvejų diagrama

Reikalavimų etape kiekvienam panaudojimo atvejui sukurta sekos diagrama, vaizduojanti ryšį tarp sistemos aktoriaus ir kiekvieno panaudojimo atvejo.

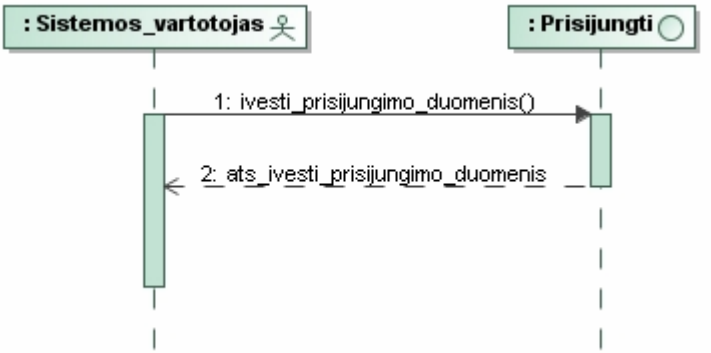
Reikalavimų modelio sekos diagramos

Žemiau pateiktose diagramose vaizduojami aktoriaus „Sistemos vartotojas“ kuriami pranešimai kiekvienam panaudojimo atvejui bei panaudojimo atvejo kuriamas atsakymo pranešimas šiam aktoriui.

Sistemos vartotojas, norėdamas užsiregistruoti sistemoje, įveda savo duomenis. Interfeisas kuria sistemos vartotojui atsakymo pranešimą apie įvestus duomenis (64 pav.).

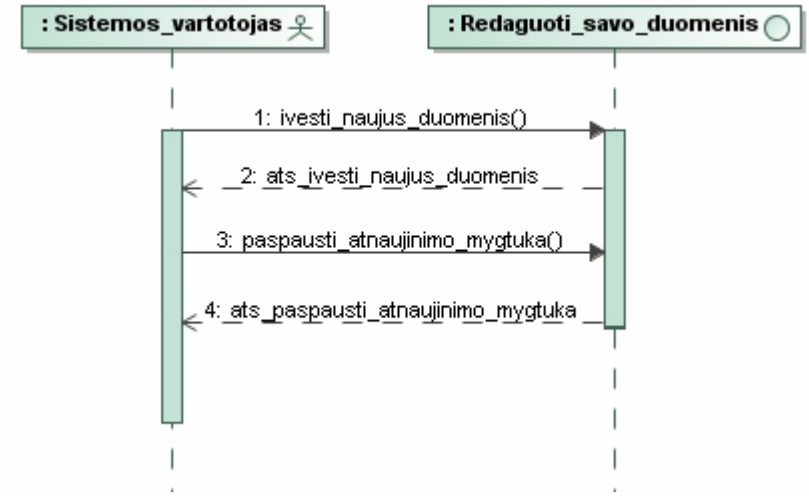
64 pav. Sekos diagrama panaudojimo atvejui „Užsiregistruoti“

Prisijungimo metu sistemos vartotojas įveda savo prisijungimo duomenis. Interfeisas kuria atsakymo pranešimą klientui apie įvestus prisijungimo duomenis (65 pav.).



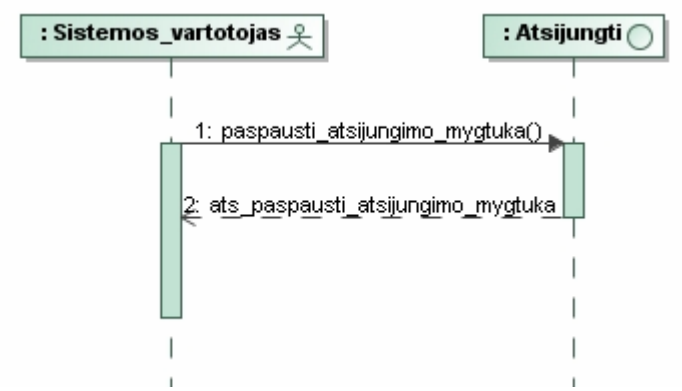
65 pav. Sekos diagrama panaudojimo atvejui „Prisijungti“

Sistemos vartotojas, norėdamas redaguoti informaciją apie save, sistemoje įveda naujus duomenis ir paspaudžia duomenų atnaujinimo mygtuką. Interfeisas šiam klientui kuria atsakymo pranešimus apie įvestus naujus duomenis (66 pav.).



66 pav. Sekos diagrama panaudojimo atvejui „Redaguoti savo duomenis“

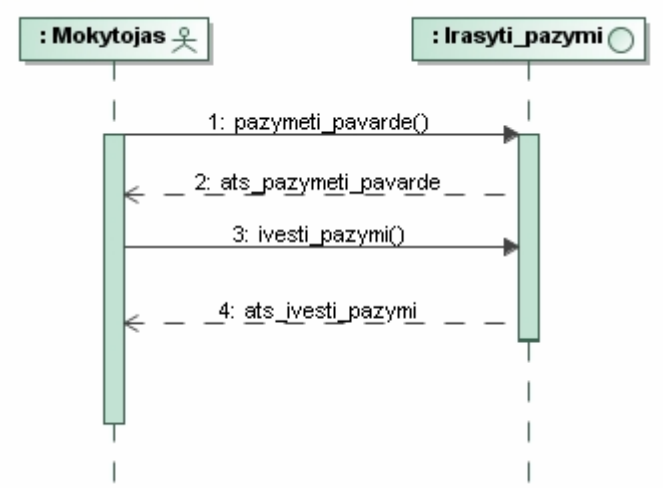
Baigęs darbą, sistemos vartotojas paspaudžia atsijungimo mygtuką, sistema jam praneša apie atsijungimą nuo elektroninio mokyklos dienyno sistemos (67 pav.).



67 pav. Sekos diagrama panaudojimo atvejui „Atsijungti“

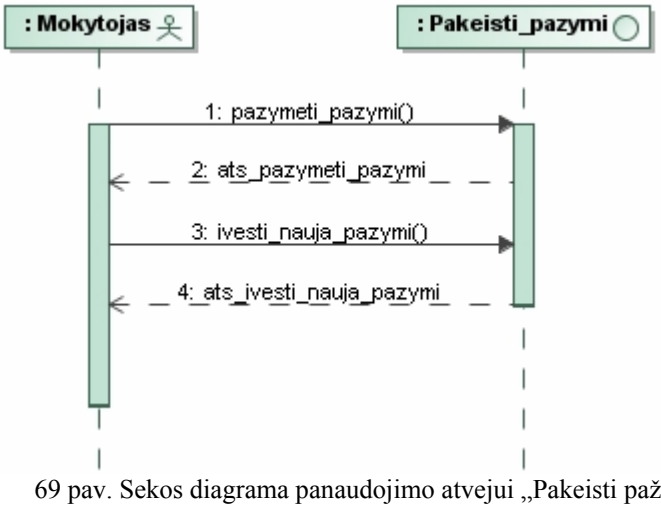
Kitas sistemos vartotojas – „Mokytojas“. Sekos diagramose vaizduojami šio aktoriaus kuriami pranešimas kiekvienam panaudojimo atvejui bei panaudojimo atvejo kuriamas atsakymo pranešimai aktoriui.

Mokytojas, norėdamas įrašyti mokinio pažymį į elektroninį dienyną, pirmiausia nurodo konkretaus mokinio pavardę. Mokytojui pažymėtoji pavardė atvaizduojama sistemoje. Mokytojas įveda pažymį, kuris įrašomas į elektroninį dienyną (68 pav.).



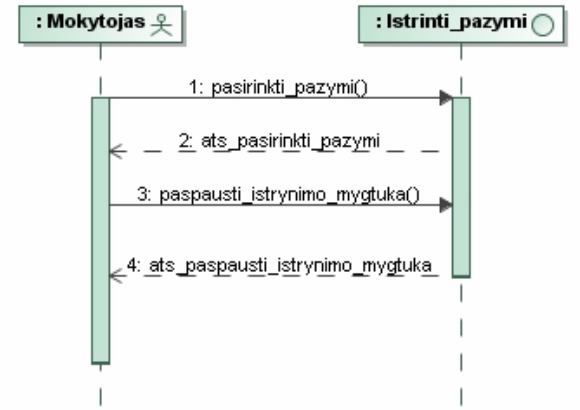
68 pav. Sekos diagrama panaudojimo atvejui „Įrašyti pažymį“

Norėdamas pakeisti pažymį, mokytojas jį pažymi ir įveda naują. Kuriamas atsakymo pranešimas mokytojui – atvaizduojamas įvestas naujas pažymys (69 pav.).



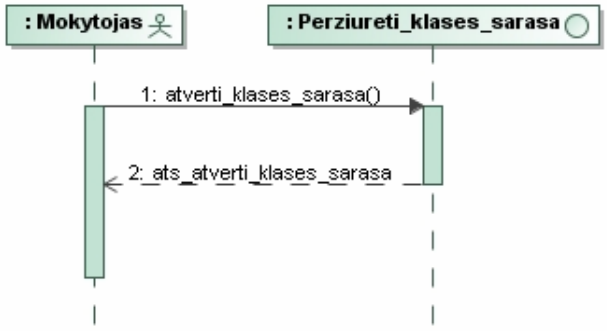
69 pav. Sekos diagrama panaudojimo atvejui „Pakeisti pažymį“

Mokytojas, norėdamas ištrinti pažymį, jį pasirenka ir paspaudžia ištrynimo mygtuką. Sistema mokytojui atvaizduoja, jog pažymys ištrintas (70 pav.).



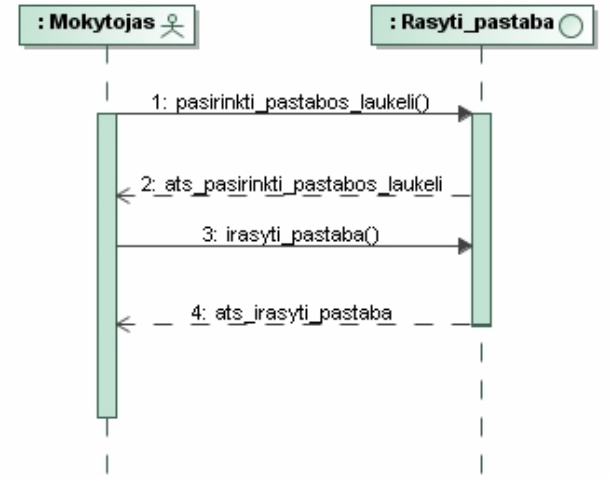
70 pav. Sekos diagrama panaudojimo atvejui „Ištrinti pažymį“

Norėdamas peržiūrėti klasės sąrašą, mokytojas paspaudžia sąrašo atidarymo mygtuką. Kuriamas atsakymo pranešimas mokytojui – pateikiamas klasės sąrašas (71 pav.).



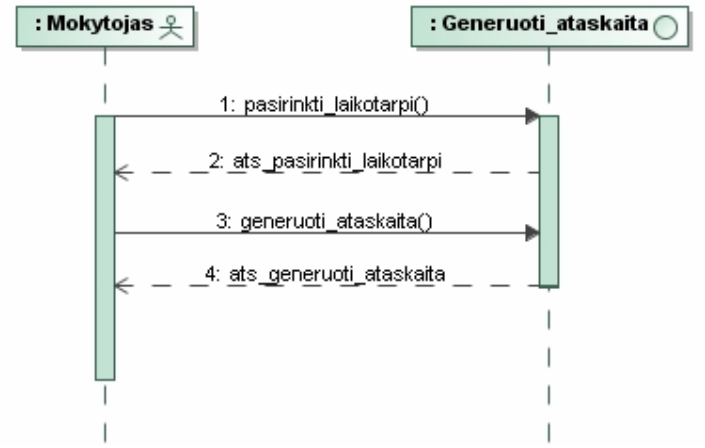
71 pav. Sekos diagrama panaudojimo atvejui „Peržiūrėti klasės sąrašą“

Šis sistemos aktorius gali rašyti pastabas mokiniams. Norėdamas tai atlikti, mokytojas pasirenka tam skirtą laukelį. Sistema jį pateikia mokytojui. Tuomet mokytojas įveda pastabą (tekstą), kuri sistemoje įrašoma konkrečiam mokiniui (72 pav.).



72 pav. Sekos diagrama panaudojimo atvejui „Rašyti pastabą“

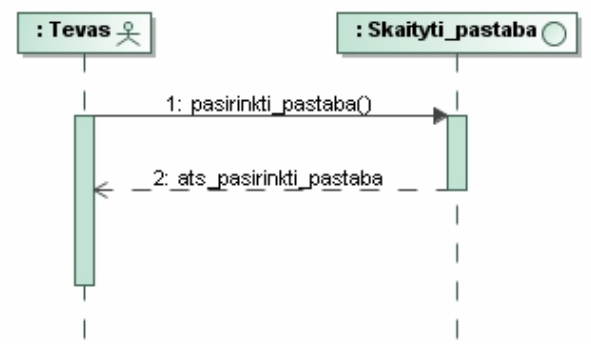
Mokytojas, norėdamas generuoti ataskaitą, pasirenka laikotarpį, kurio duomenis pageidauja pateikti ataskaitoje; sistema fiksuoja šį mokytojo nurodytą laikotarpį. Mokytojas paspaudžia ataskaitos generavimo mygtuką ir sistema sugeneruoja šią ataskaitą (73 pav.)



73 pav. Sekos diagrama panaudojimo atvejui „Generuoti ataskaitą“

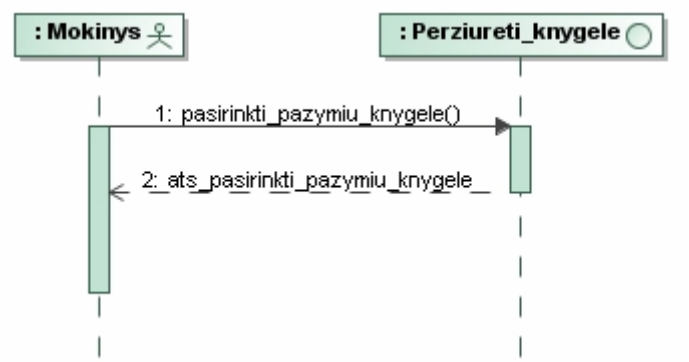
Kiti du sistemoje egzistuojantys vartotojai yra Tėvas ir Mokinys.

Aktorius *Tėvas*, prisijungęs sistemoje (prisijungimo atvejis aprašytas 65 pav.) pasirenka pastabos laukelį. Sistema kuria atsakymo pranešimą šiam aktoriui – atvaizduoja pastabos tekstą (74 pav.)



74 pav. Sekos diagrama panaudojimo atvejui „Skaityti pastabą“

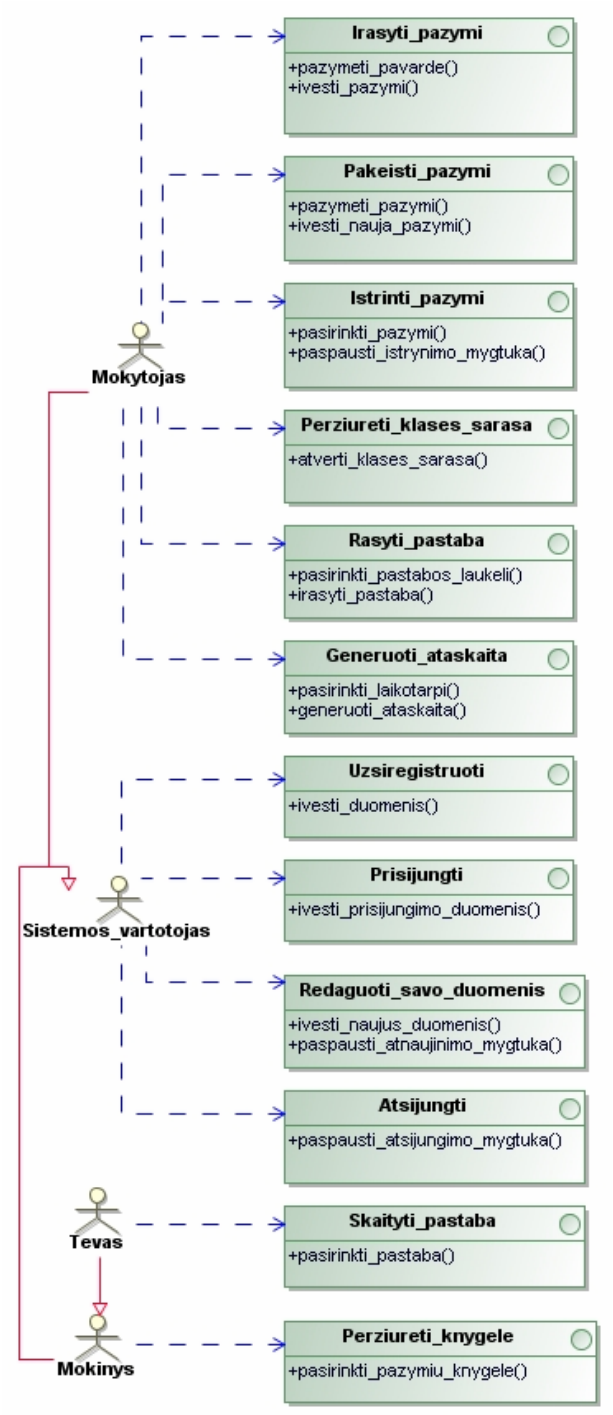
Mokinys gali peržiūrėti savo pažymių knygelę paspausdamas tam skirtą mygtuką (75 pav.). Sistemos lange mokiniui pateikiami jo pažymiai.



75 pav. Sekos diagrama panaudojimo atvejui „Peržiūrėti knygelę“

Interfeisų modelis

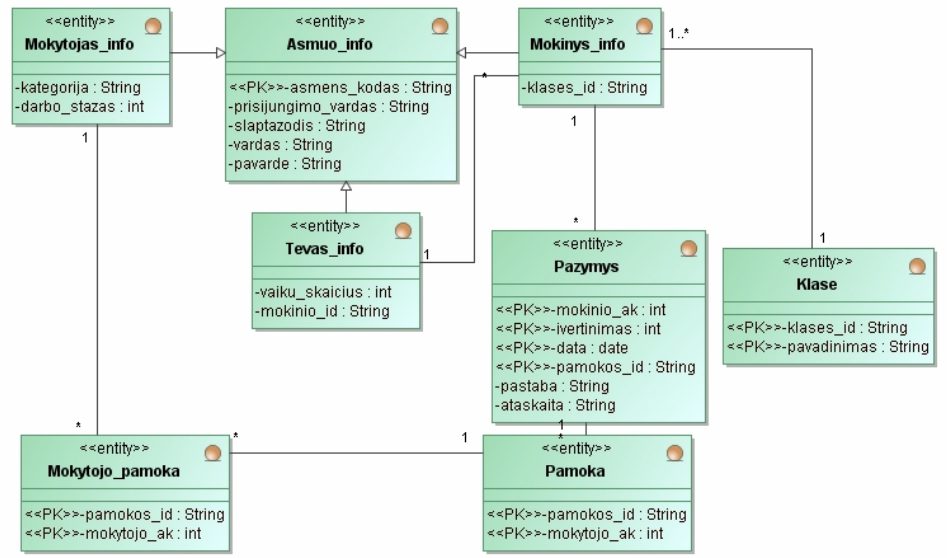
Reikalavimų modelio interfeisai transformuojami į projektą ir išlieka nepakitę (76 pav.).



76. pav. Interfeisų modelis

Esybių modelis

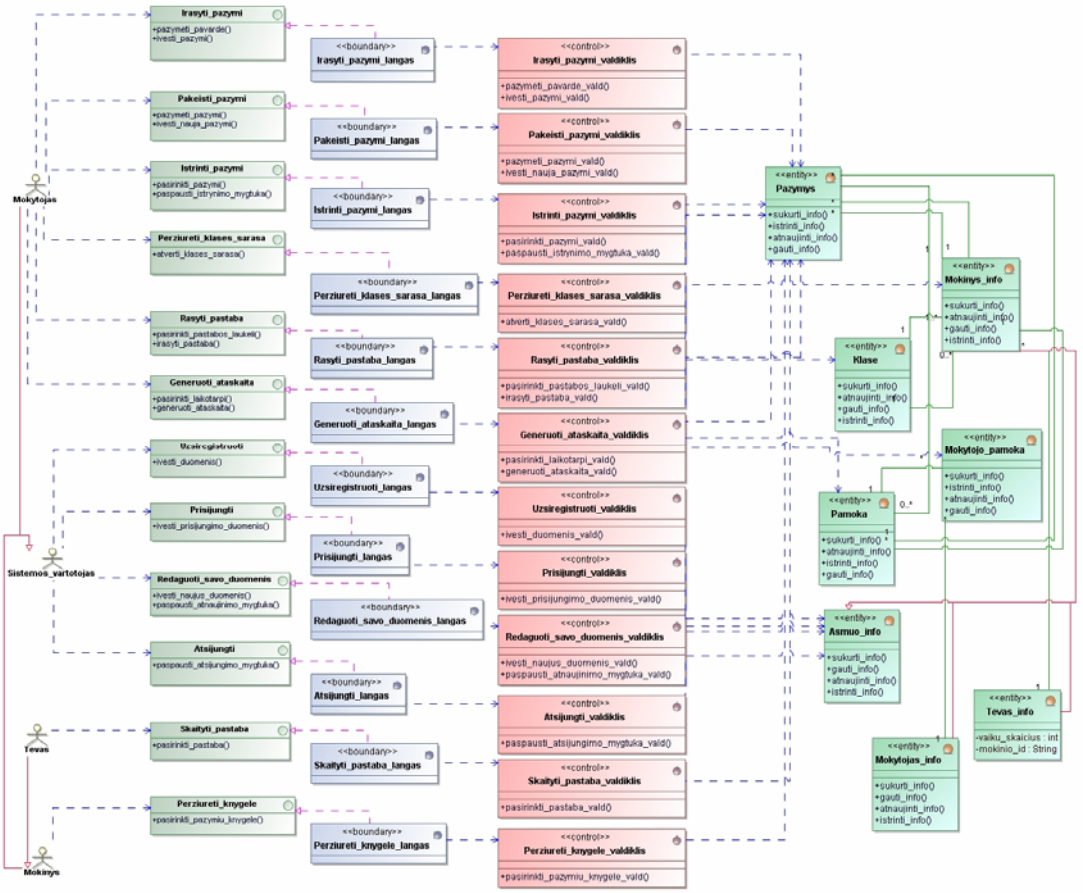
Transformavus reikalavimų modelio esybių klases, vedlio sistemos eigoje kiekvienai esybės klasei sukurtos 4 operacijos (77 pav.)



77. pav. Esybių modelio schema

5.3.2. Projekto (rezultatų) modelis

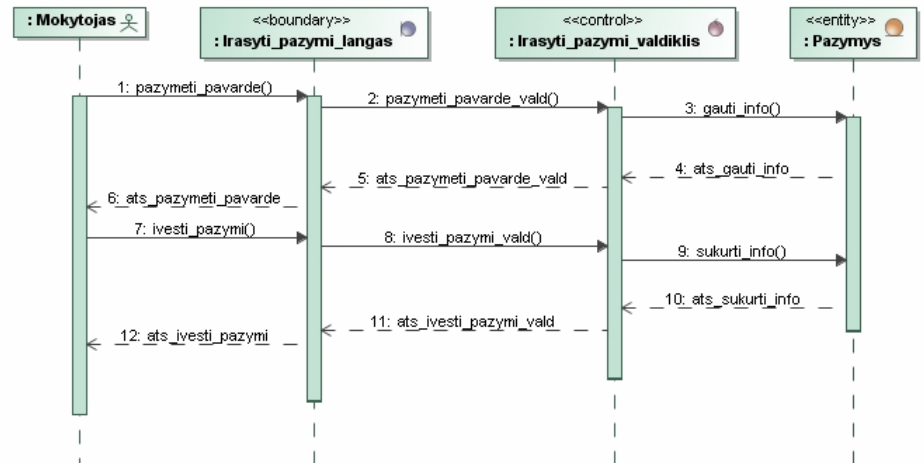
Projekto modelyje gaunama sugeneruota klasių diagrama (78 pav.), kurioje vaizduojami aktoriai, interfeisai, ribinės klasės, valdikliai, esybės ir ryšiai tarp jų.



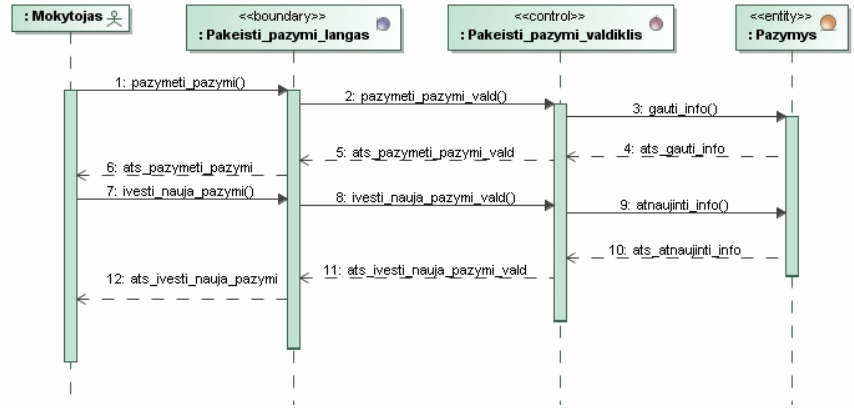
78. pav. Projekto modelyje sugeneruota klasių diagrama

Sugeneravus projekto modelio klasių diagramą, kiekvienai reikalavimų etapo diagramai suprojektuojama po vieną sekos diagramą MVC šablono principu: projekto aktoriai ir klasės įkeliamos į projektuojamą diagramą, operacijos pasirenkamos iš operacijų sąrašo.

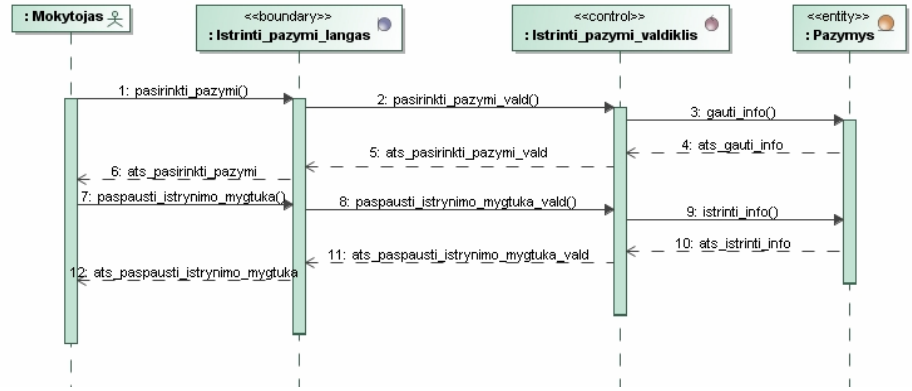
Gautos sekų diagramos vaizduojamos žemiau pateiktuose paveiksluose:



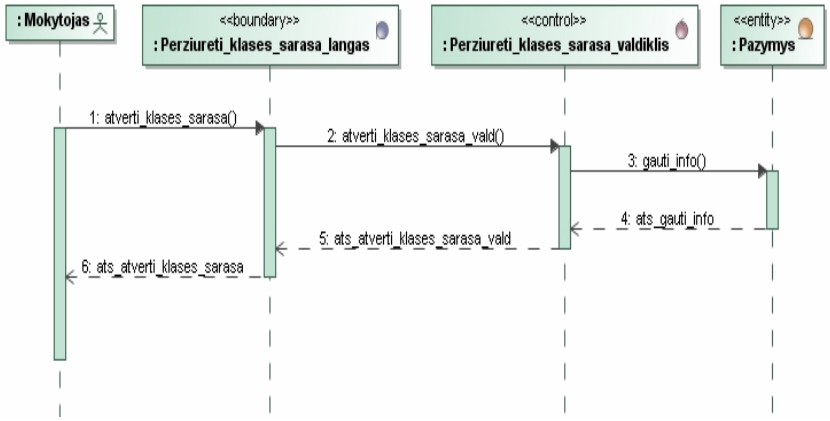
79 pav. „Įrašyti pažymį“ sekos diagrama, suprojektuota MVC šablono principu



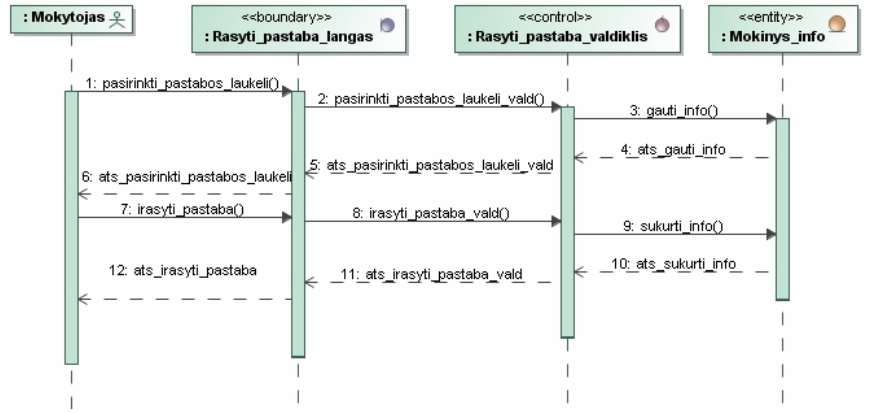
80 pav. „Pakeisti pažymį“ sekos diagrama, suprojektuota MVC šablono principu



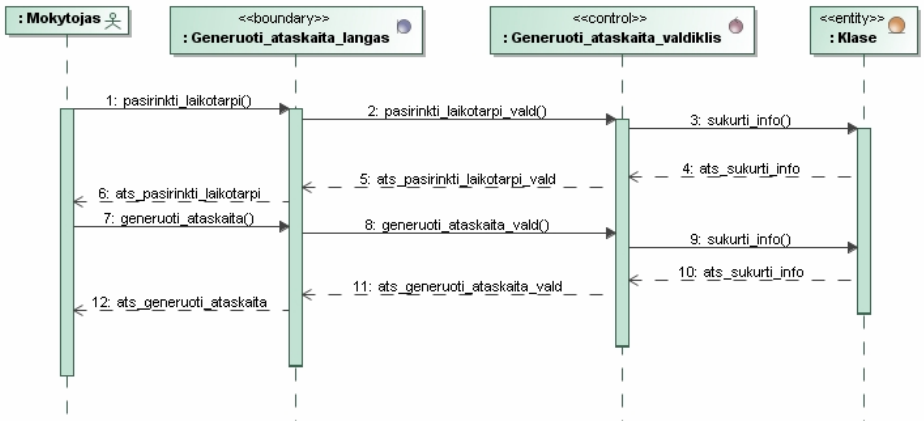
81 pav. „Ištrinti pažymį“ sekos diagrama, suprojektuota MVC šablono principu



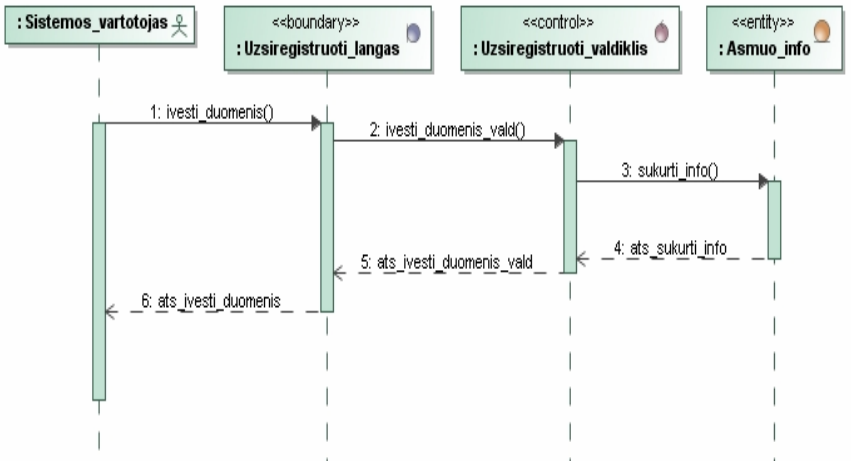
82 pav. „Peržiūrėti klasės sąrašą“ sekos diagrama, suprojektuota MVC šablono principu



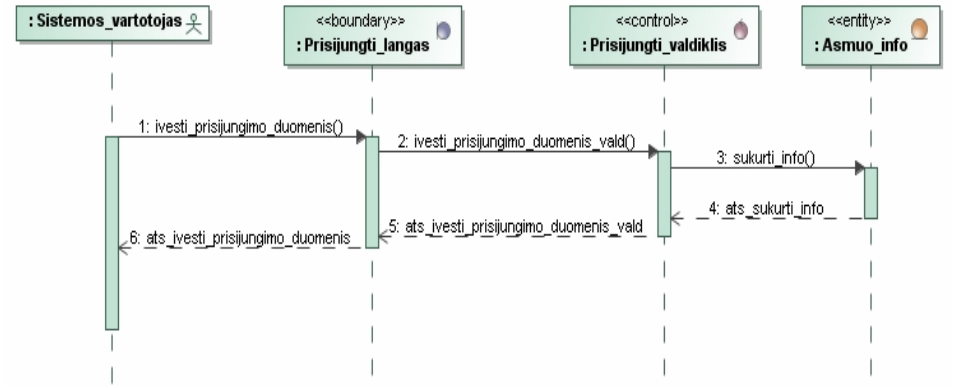
83 pav. „Rašyti pastabą“ sekos diagrama, suprojektuota MVC šablono principu



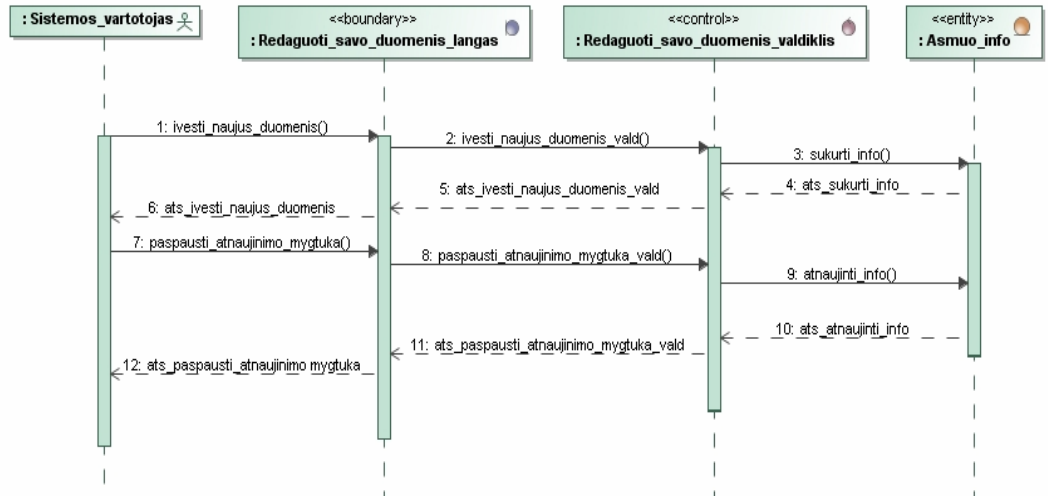
84 pav. „Generuoti ataskaitą“ sekos diagrama, suprojektuota MVC šablono principu



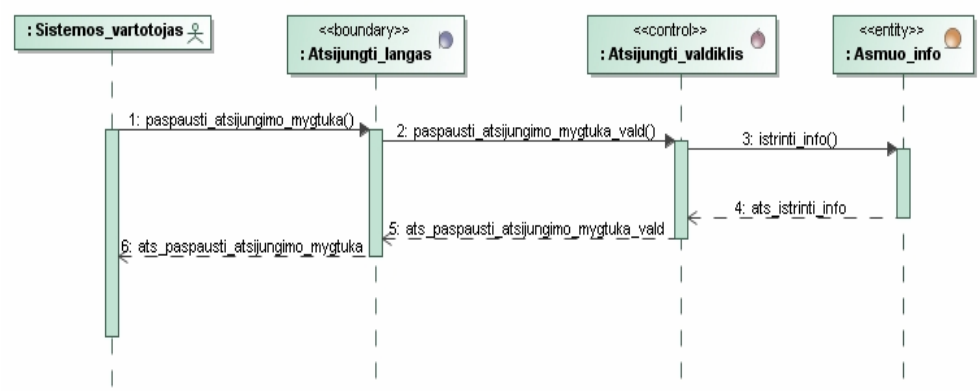
85 pav. „Užsiregistruoti“ sekos diagrama, suprojektuota MVC šablono principu



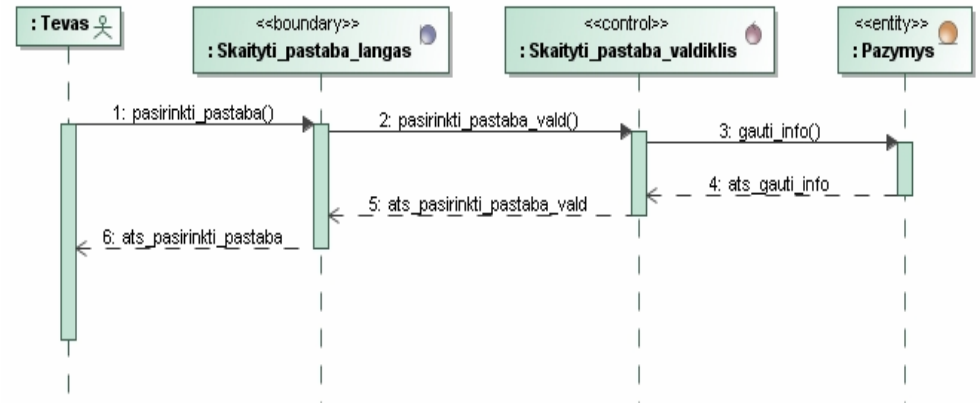
86 pav. „Prisijungti“ sekos diagrama, suprojektuota MVC šablono principu



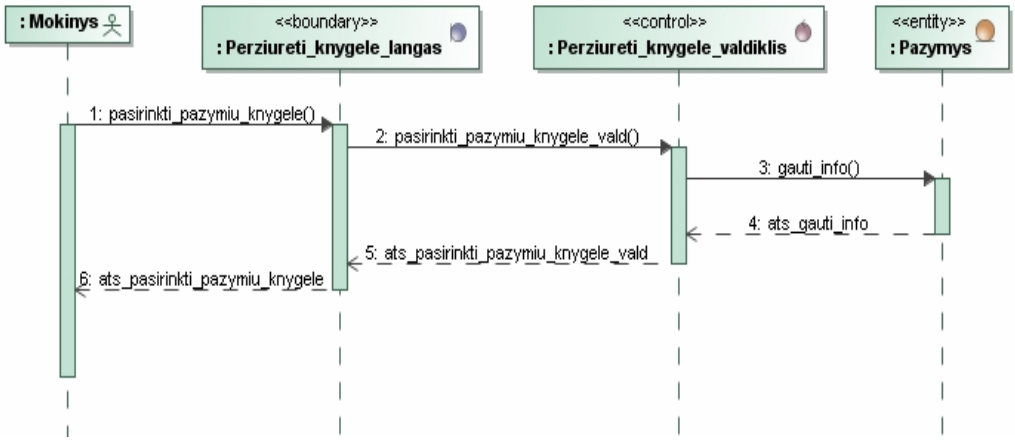
87 pav. „Redaguoti savo duomenis“ sekos diagrama, suprojektuota MVC šablono principu



88 pav. „Atsijungti“ sekos diagrama, suprojektuota MVC šablono principu



89 pav. „Skaityti pastabą“ sekos diagrama, suprojektuota MVC šablono principu



90 pav. „Peržiūrėti knygelę“ sekos diagrama, suprojektuota MVC šablono principu

5.4. Transformavimo algoritmo veikimo ir savybių analizė, taikymo rekomendacijos

Darbe pasiūlytą algoritmą rekomenduojama taikyti, jeigu siekiama palengvinti informacinės sistemos kūrėjo (projektuotojo), naudojančio MVC šabloną, darbą. Projektuotojui užtenka aprašyti reikalavimų modelį, o pradinį projekto modelio variantą, naudojant pasiūlytą algoritmą, sugeneruoja klasių diagramos vedlys. Taip išvengiama diagramos kūrimo rankiniu būdu.

6. Išvados

1) Išanalizavus EMDA kūrimo metodą, pastebėta, kad jame siūloma transformacija iš reikalavimų į projekto modelį yra grindžiama tik vienu pasiūlytu architektūriniu šablonu, o būtų naudinga sukurti ir daugiau transformacijų variantų, suteikiančių projektuotojui galimybę rinktis pageidaujamą architektūrinį šabloną.

2) Atlikus IS kūrimo metodų analizę pastebėta, jog RUP metode aiškiai atskirti reikalavimų analizės ir projektavimo etapai, metodas pagrįstas užduotimis, per kurias išreiškiami reikalavimai; procese akcentuojama architektūra, todėl šis metodas pasirinktas taikymui transformacijoje.

3) Išanalizavus *GoF* architektūrinius projektavimo šablonus pastebėta, jog RUP procese dažnai rekomenduojama taikyti MVC principus, todėl iš *GoF* šablonų detaliau analizuotas ir taikymui transformacijoje pasirinktas Stebėtojo šablonas bei konkretesnis jo taikymo pavyzdys – MVC šablonas, kuris atskiria vaizdavimo, duomenų ir veiklos logikos sluoksnius.

4) Atlikus UML CASE įrankių analizę pastebėta, jog MagicDraw įrankis turi daugiau galimybių ne tik UML diagramų modeliavimo, bet ir transformacijų srityje, todėl šis įrankis pasirinktas transformavimo iš reikalavimų į projektą algoritmo realizavimui.

5) Darbe pasiūlytas algoritmas palengvina projektuotojo, naudojančio MVC šabloną, darbą. Projektuotojui užtenka aprašyti reikalavimų modelį, o pradinį projekto modelio variantą galima sugeneruoti, naudojant pasiūlytą algoritmą.

5) Transformavimo algoritmas realizuotas naudojant sukurtą vedlį „T*ransformationFromDIMtoPIM*“, kuris generuoja projekto modelio klases bei tų klasių diagramą. Ši priemonė leidžia išvengti projekto klasių diagramos kūrimo rankiniu būdu, taip automatizuojant diagramos (ir jos elementų) kūrimą.

6) Ateityje būtų naudinga realizuoti ir kitus GoF projektavimo šablonus, projektuotojui suteikiant galimybę rinktis jam tinkamą variantą iš daugelio siūlomų.

7. Literatūra

1. ČEPONIENĖ, L. *Informacijos sistemų reikalavimų analizės ir transformavimo į projektą metodas*: daktaro disertacija. Technologijos mokslai, informatikos inžinerija (07T). Kauno technologijos universitetas. [Kaunas], 2006.
2. EVANS, G. Getting from use cases to code, Part 1: Use-Case Analysis [interaktyvus]. 2004, liepa [žiūrėta 2007-09-28]. Prieiga per internetą: <http://www.ibm.com/developerworks/rational/library/5383.html>.
3. CANTOR, M. *Rational Unified Process for Systems Engineering. Part 1: Introducing RUP SE Version 2.0.* [interaktyvus]. 2003, rugpjūtis [žiūrėta 2007-10-03]. Prieiga per internetą: <http://www.therationaledge.com/content/aug_03/f_rupse_mc.jsp>.
4. FRANKEL, D. *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley & Sons, 2003.
5. KLEPPE, A.; WARMER, J.; ir BAST, W. *MDA Explained: The Model Driven Architecture™: Practice and Promise.* Addison Wesley. Boston, 2003.
6. JACOBSON, I.; BOOCH, G.; ir RUMBAUGH, J. The Unified Software Development Process. Addison Wesley, 1999.
7. KROLL, P.; ir KRUCHTEN, P. *Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. 2003.
8. GAMMA, Erich, et al. *Design patterns, Elements of reusable object-oriented software*. Second ed. Addison Wesley, 2004.
9. *MSDN Biblioteka* [interaktyvus]. Model-View-Controler. Version 1.0.1. [žiūrėta 2008-01-03] Prieiga per internetą: <http://msdn2.microsoft.com/en-us/library/ms978748.aspx>.
10. MAHEMOFF, M. J.; ir JOHNSTON, L. J. . *Dvipsk. HandlingMultiple Domain Objects with Model-View-Controller.*5.58f , Copyright 1986, 1944 Radical Eye Software. 1-12 p.
11. KRASNER, G. E.; POPE, S. R. *A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80*. JOOP, Rugpjūtis, Rugsėjis 1988. 26-49 p.
12. BUSHMAN, Frank, et al. *Software architecture: a system of patterns*. John Wiley & Sons. Chichester, 1996. 123-139 p.
13. REENSKAUG, T. The Model-View-Controller (MVC) - Its Past and Present. Oslo, Norway, 2003.

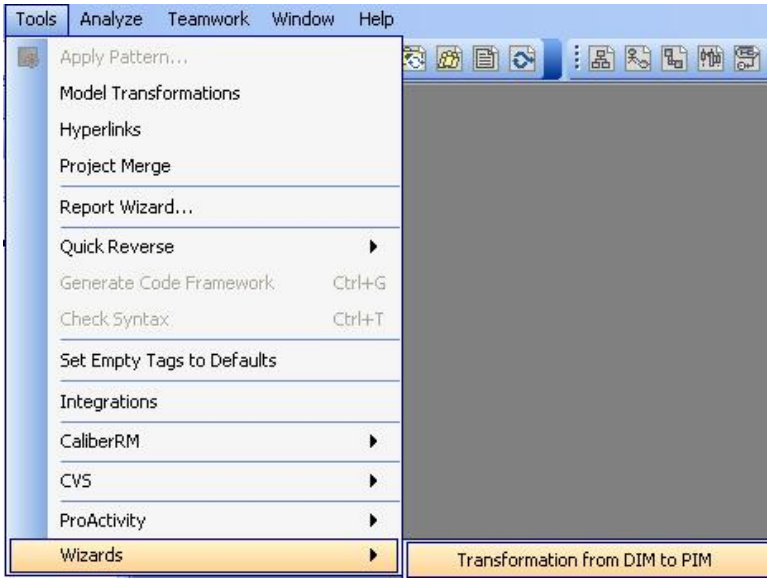
14. FOWLER, M. GUI Architectures [interaktyvus]. [žiūrėta 2008-01-14]. Prieiga per internetą: <http://www.martinflower.com/eaaDevuiArchs.html>.
15. Applying the Model/View/Controller Design Paradigm in VisualAge for Java [interaktyvus]. [žiūrėta 2008-01-06]. Prieiga per internetą: <http://javadude.com/>.
16. BURBECK, S. Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC). [žiūrėta 2008-01-06] Prieiga per internetą: <http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>.
17. *Designing Enterprise Applications with J2EE Platform, Second Edition. Architecture of The Sample Application.* [žiūrėta 2008-01-07] Prieiga per internetą: <http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e>.
18. ROSENBERG, D.; ir SCOTT, K. *Applying Use Case Driven Object Modelling with UML 2*. Addison Wesley. 58-74 p. ISBN: 0-201-73039-1.
19. ROSENBERG, D.; ir SCOTT, K. Driving Design with Use Cases. Software Development, 2000. [žiūrėta 2008-01-06] Prieiga per internetą: <http://www.sdmagazine.com/print/>.
20. MAHMOUD, H. Q.; ir MAAMAR, Z. *Applying the MVC Design Pattern to Multi-Agent Systems*. 1-4244-0038-4, IEEE CCECE/CCGEI. Ottawa, Kovas, 2006. 2420-2423 p.
21. ŠILINGAS, D; ir VITIUTINAS, R. Towards UML-Intensive Framework for Model-Driven Development. Iš Balancing *Agility and Formalism in Software Engineering* [interaktyvus]. Springer Berlin, 2008 [žiūrėta 2008-08-10]. ISBN 978-3-540-85278-0. Prieiga per internetą: <http://www.springerlink.com/content/pn326r3k4pl28756/>.
22. REED, P. R. *Reference Architecture: The Best of Best Practices*. Rational Software, 2002 [žiūrėta 2008-01-06]. Prieiga per internetą: <http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep02/ReferenceArchitectureSep02.pdf>.
23. ROSENBERG, D.; and SCOTT, K. *Successful Robustness Analysis*. Software Development, 2001. [žiūrėta 2008-05-12] Prieiga per internetą: <http://www.sdmagazine.com/articles/2001/0103/0103c/0103c.htm>.
24. NEMURAITĖ, L.; ir KAVALIAUSKAITĖ, L. *Informacinių sistemų projektavimo metodų tobulinimas ir automatizavimas, taikant UML*. Informacinės Technologijos, 2002. Technologija. Kaunas, 2002. p. 408 - 414.

Terminų ir santrumpų žodynėlis

CASE	-	(angl. *Computer Aided Software Engineering*) kompiuterizuota programinės įrangos inžinerija.
CIM	-	(angl. *Computation Independent Model*) nuo skaičiavimų nepriklausomas modelis.
DIM	-	(angl. *Design Independent Model*) nuo projektavimo metodo nepriklausantis reikalavimų modelis.
EMDA	-	(angl. *Extended Model Driven Approach*) IS reikalavimų analizės ir transformavimo į projektą metodas.
GOF	-	(angl.*Gang of Four)* architektūrinių projektavimo šablonų grupė.
ICONIX	-	programinės įrangos kūrimo metodas.
IS	-	Informacijos sistema.
MDA	-	(angl. *Model Driven Architecture)* modeliais grindžiama architektūra.
MVC	-	(angl. *Model-View-Controller)* architektūrinis projektavimo šablonas.
PIM	-	(angl. *Platform Independent Model*) nuo realizavimo platformos nepriklausantis projekto modelis.
PSM	-	(angl. *Platform Specific Model*) konkrečiai realizavimo platformai pritaikytas modelis.
RUP	-	(angl. *Rational Unified Process)* IBM/Rational organizacijos sukurtas programų sistemų kūrimo procesas.
UML	-	(angl. *Unified Modelling Language*) unifikuota modeliavimo kalba.
XP	-	(angl. eXtreme Programming) IS kūrimo metodas.

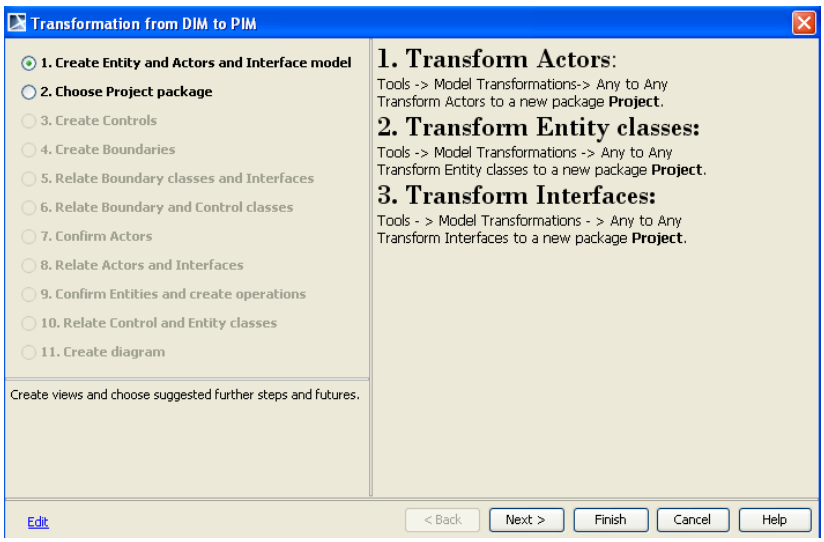
Priedas. Realizuoto transformavimo algoritmo vartotojo instrukcija

Vedlys „*Transformation from DIM to PIM*“ paleidžiamas MagicDraw programos kontekstiniame meniu pasirinkus *Tools/Wizards/ Transformation from DIM to PIM*.



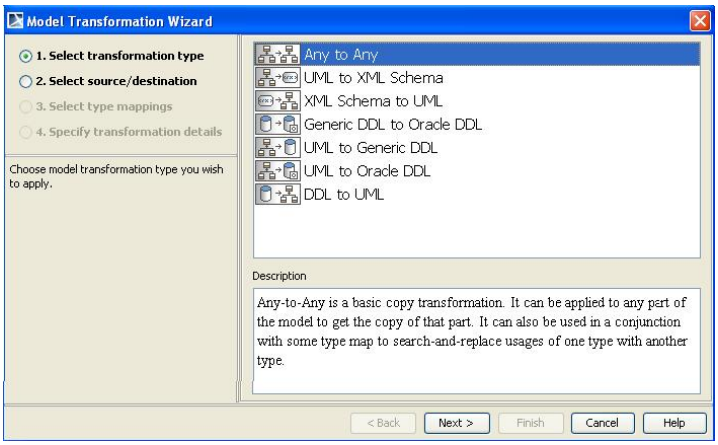
91. pav. Vedlio paleidimo langas

Pereinant nuo reikalavimų prie projektavimo etapo, reikalavimų modelio *aktoriai*, *esybės* ir *interfeisai* perkeliami į projekto paketą. Naudojamas modelių transformavimo priemonė (91 pav.) (Kopijuojant elementus, kopija yra identiška originalui, o transformuotas elementas virsta savarankišku).

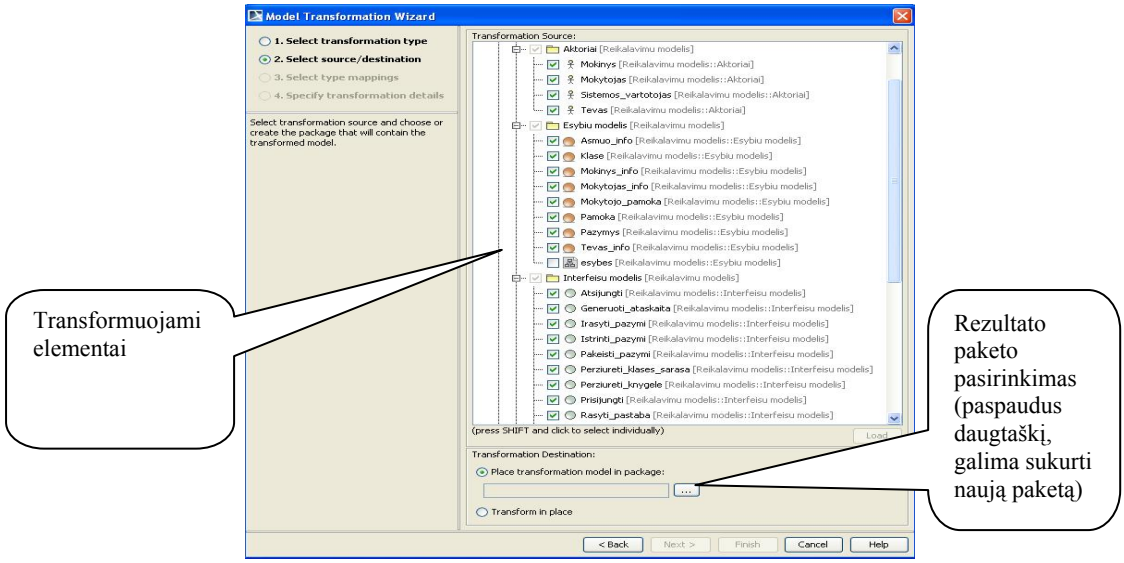


91 pav. Aktorių, dalykinės srities ir interfeisų modelių transformacijos langas

Pasirenkama *Tools/Model Transformations/Any to Any*:



92 pav. Transformavimo įrankio pasirinkimas

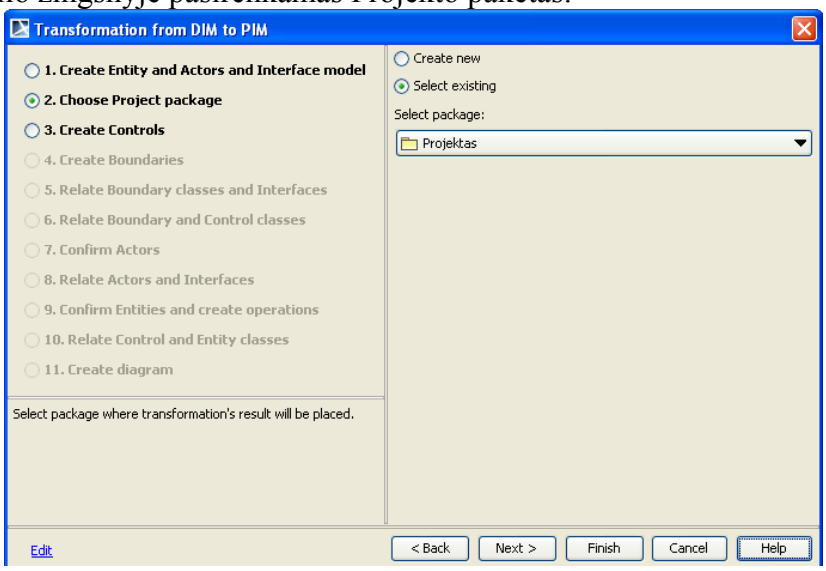


93 pav. Transformacijos šaltinio ir rezultato patalpinimo vietos pasirinkimas

Po transformacijos naujame projekto pakete vaizduojami transformuoti elementai (94 pav.):

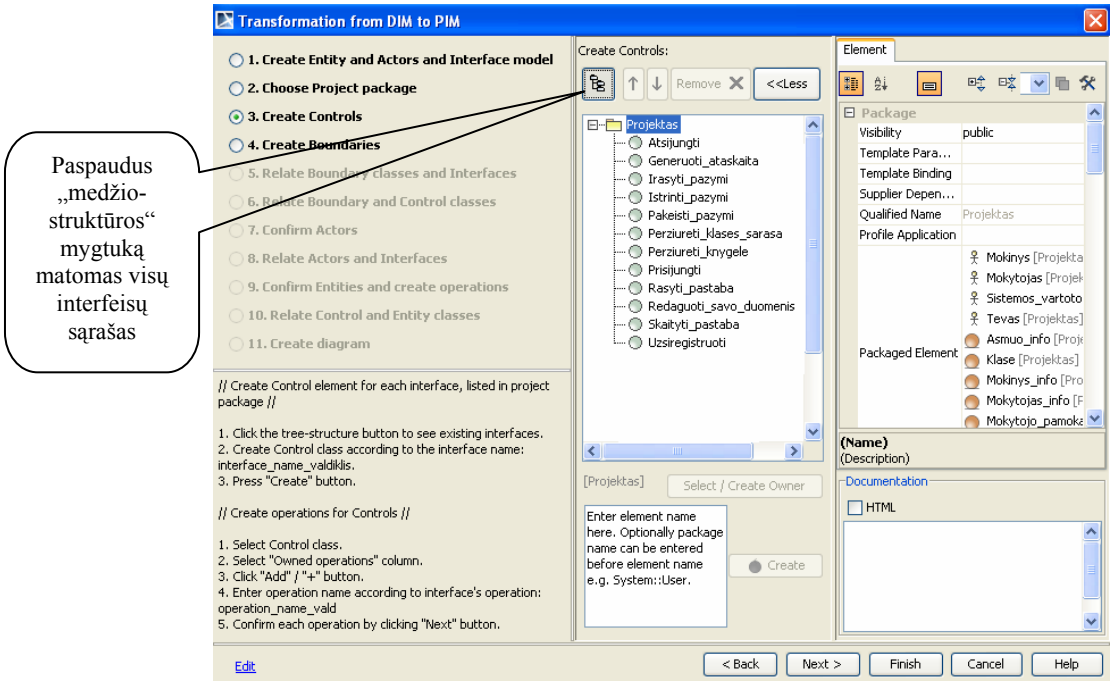
94 pav. Transformuotų elementų sąrašas

Antrajame vedlio žingsnyje pasirenkamas Projekto paketas:



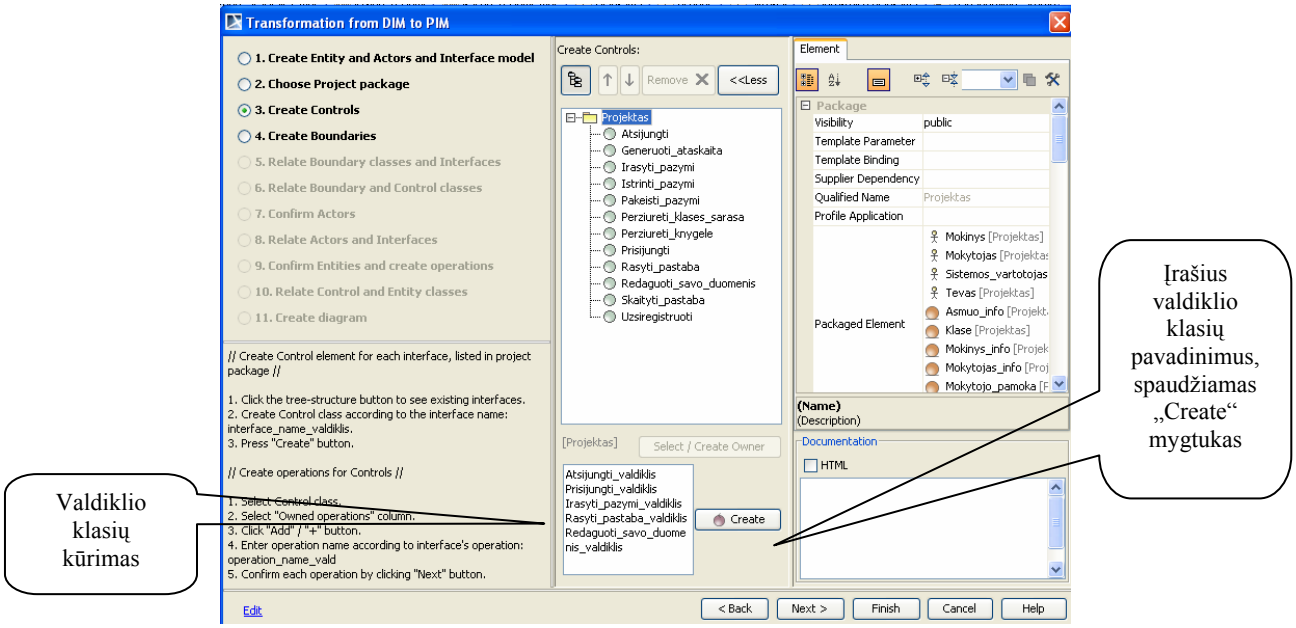
95 pav. Projekto paketo pasirinkimo langas

Trečiasis žingsnis – valdiklio (angl. *Control*) klasių sukūrimas. Kiekvienam projekto interfeisui kuriama po vieną valdiklio klasę.



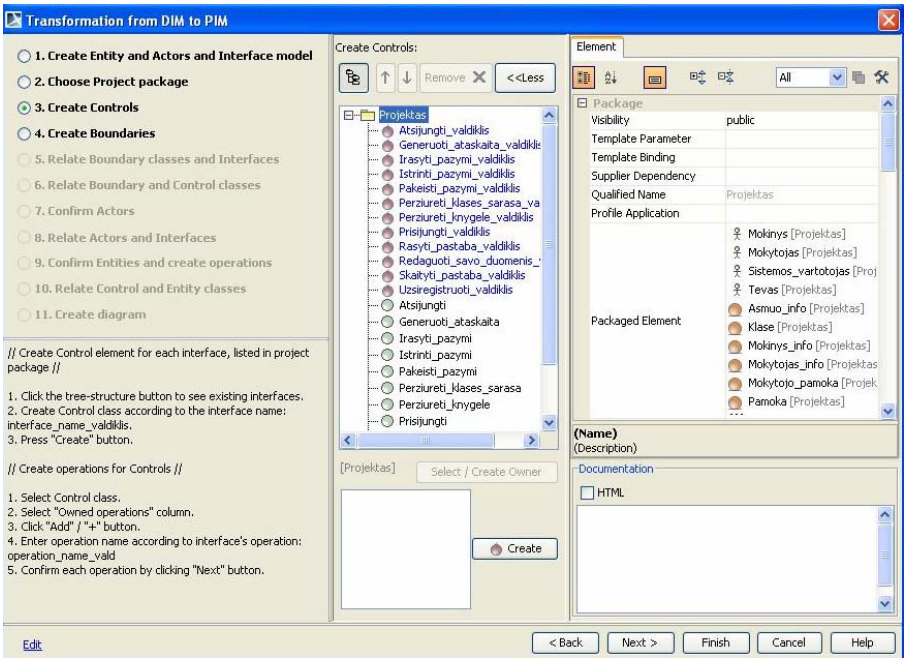
96 pav. Valdiklio klasių kūrimo langas

Kuriamos valdiklio klasės pavadinimas turi sutapti su interfeiso pavadinimu, prie jo pridedant galūnę „valdiklis“. Valdiklio klasių pavadinimai rašomi žemiau interfeisų sąrašo esančiame lange, atskiriant juos „enter“ paspaudimu. Surašius valdiklių pavadinimus, spaudžiamas „Create“ mygtukas (97 pav.).



97 pav. Valdiklio klasių kūrimo langas

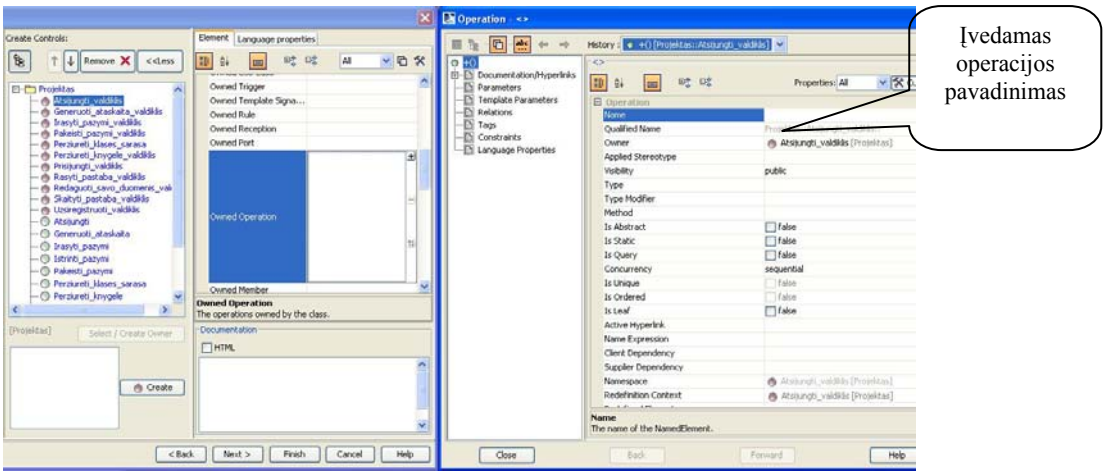
Sukurtos valdiklio klasės atvaizduojamos pakete „Projektas“ (98 pav.):



98 pav. Valdiklio klasių atvaizdavimas projekto pakete

Kiekvienai valdiklio klasei kuriamos operacijos (99 pav.):

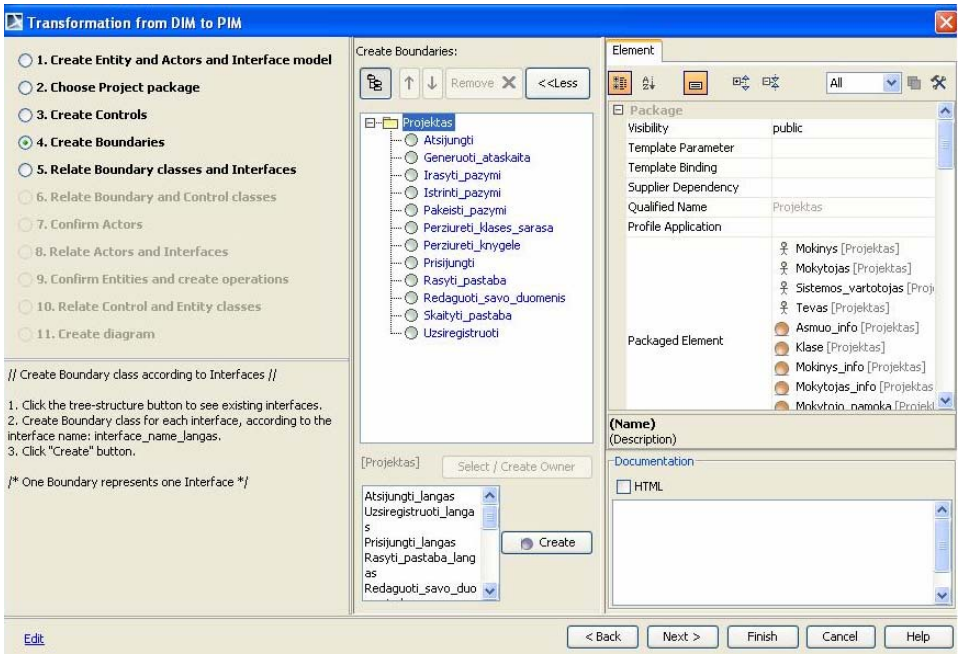
- ✓ pelės mygtuko paspaudimu pažymima valdiklio klasė;
- ✓ specifikacijos lango „Owned operations“ skiltyje spaudžiamas „Add“ arba „+“ mygtukas;
- ✓ ivedamas operacijos pavadinimas (operacijos pavadinimas turi sutapti su interfeiso operacijos pavadinimu, prie jo pridedant galūnę „vald“);
- ✓ kiekviena sukurtoji operacija patvirtinama „Next“ mygtuko paspaudimu.



99 pav. Valdiklio klasių operacijų kūrimas

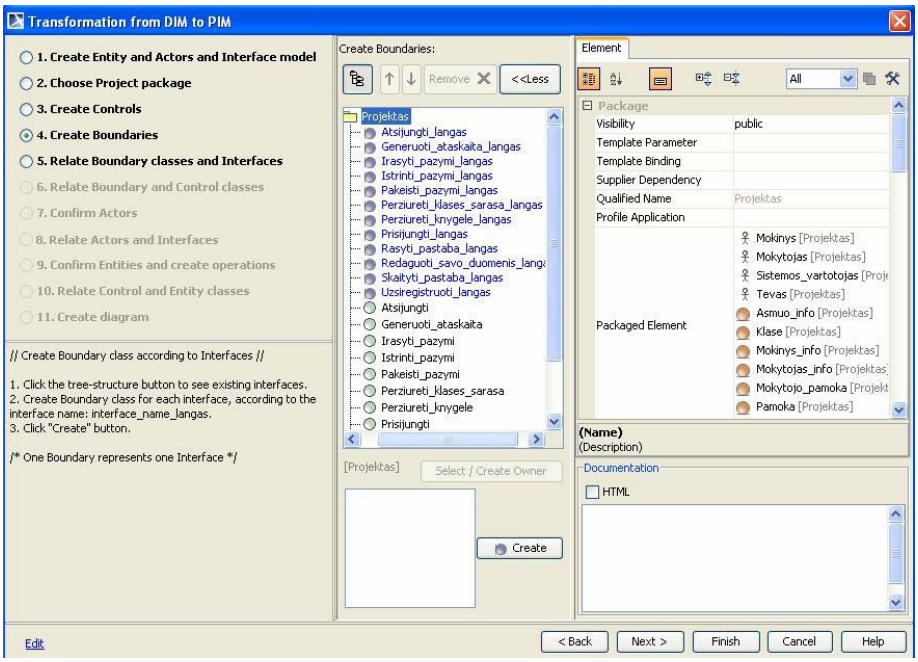
Ketvirtajame vedlio žingsnyje kuriamos ribinės (angl. *Boundary*) klasės - kiekvienam projekto interfeisui sukuriama po vieną ribinę klasę (100 pav.).

Kuriamos ribinės klasės pavadinimas turi sutapti su interfeiso pavadinimu, prie jo pridedant galūnę „langas“. Valdiklio klasių pavadinimai rašomi žemiau interfeisų sąrašo esančiame lange, atskiriant juos „enter“ paspaudimu. Surašius valdiklių pavadinimus, spaudžiamas „Create“ mygtukas.



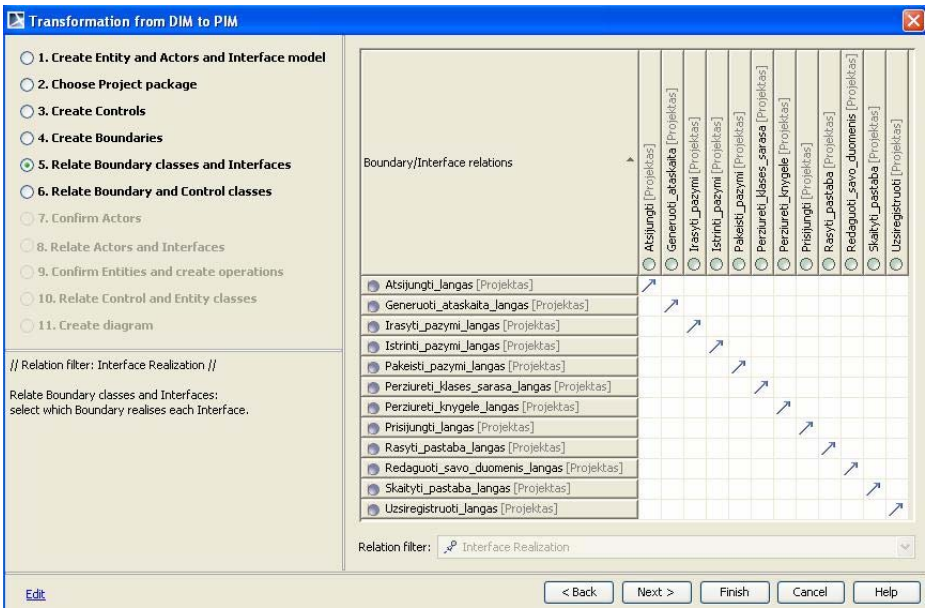
100 pav. Ribinių klasių kūrimo langas

Sukurtos ribinės klasės atvaizduojamos pakete „Projektas“ (101 pav.):

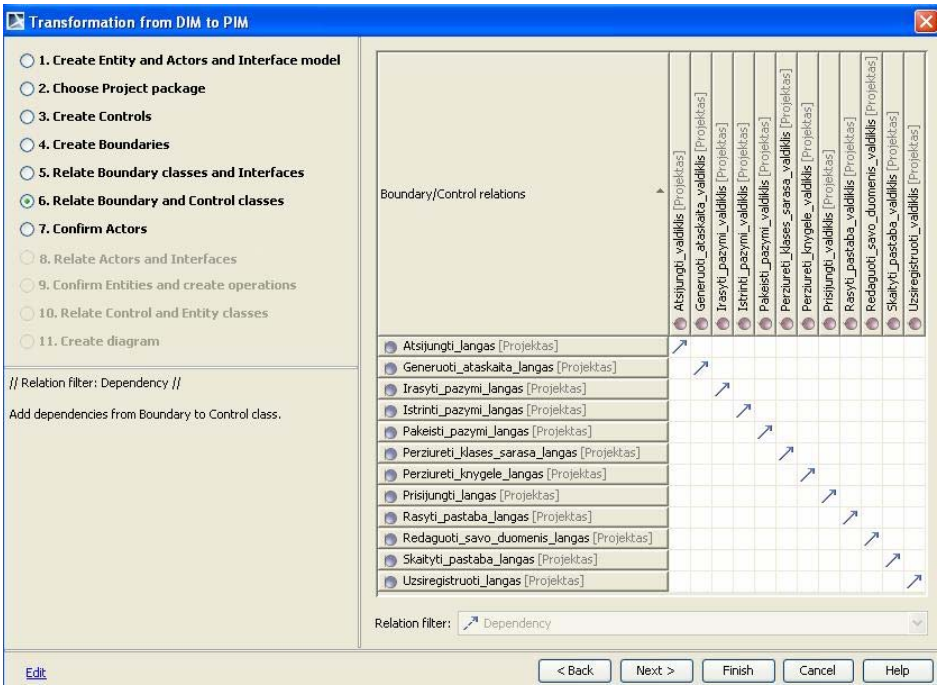


101 pav. Ribinių klasių atvaizdavimas projekto pakete

Žemiau pateiktuose paveiksluose vaizduojami ryšių tarp projekto modelio interfeisų ir klasių nurodymo paveikslai:

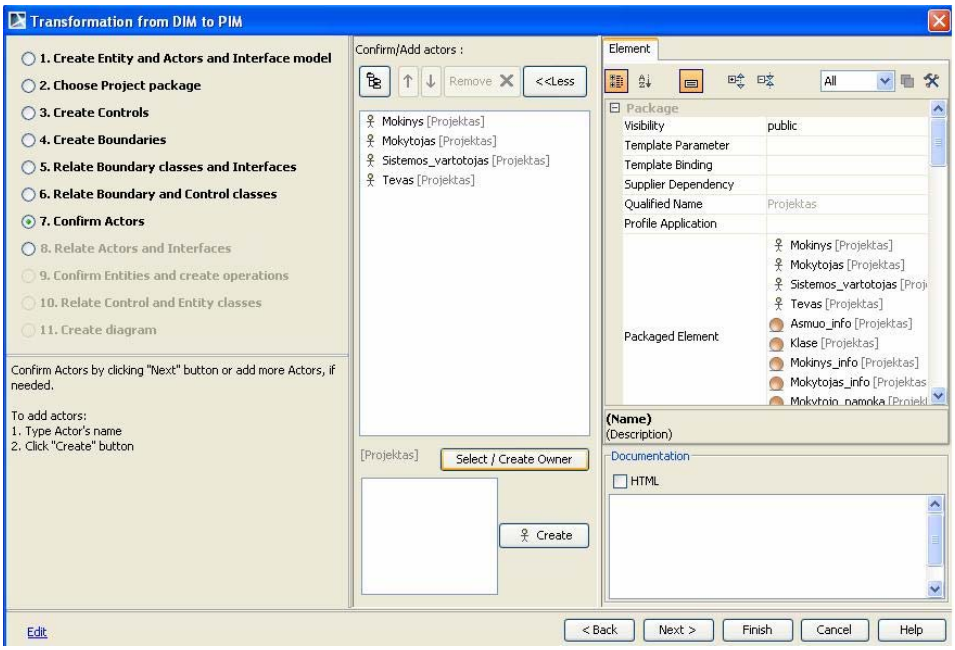


102 pav. Ryšių tarp ribinių klasių ir interfeisų nurodymo langas



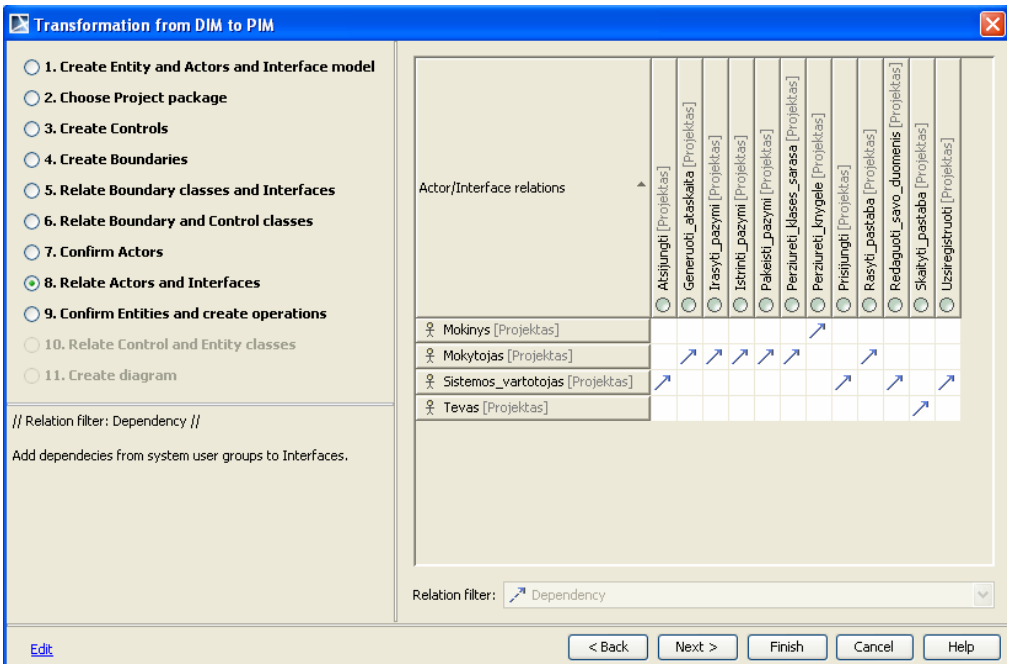
103 pav. Ryšių tarp ribinių ir valdiklio klasių nurodymo langas

Reikalavimų modelio aktoriai po transformacijos talpinami projekto pakete (104 pav.). Aktorių modelis gali būti papildomas įvedant naujus aktorius tam skirtoje vedlio lango dalyje. Įvedus aktorių pavadinimus spaudžiamas „Create“ mygtukas. Nauji aktoriai atvaizduojami projekto pakete kartu su anksčiau sukurtais. Aktorių sąrašas patvirtinamas „Next“ mygtuko paspaudimu.



104 pav. Aktorių patvirtinimo (papildymo) langas

Nurodomi ryšiai tarp projekto modelio aktorių ir interfeisų (105 pav.)



105 pav. Ryšių tarp aktorių ir interfeisų nurodymo langas

Po transformacijos esybių klasės talpinamos projekto pakete. Esybių sąrašas gali būti papildomas įvedant naujas klases tam skirtoje vedlio lango dalyje. Įvedus klasių pavadinimus spaudžiamas „Create“ mygtukas. Naujai sukurtos esybių klasės atvaizduojamos projekto pakete kartu su anksčiau sukurtomis. Klasių sąrašas patvirtinamas „Next“ mygtuko paspaudimu.

Kiekvienai esybės klasei sukuriama po 4 operacijas:

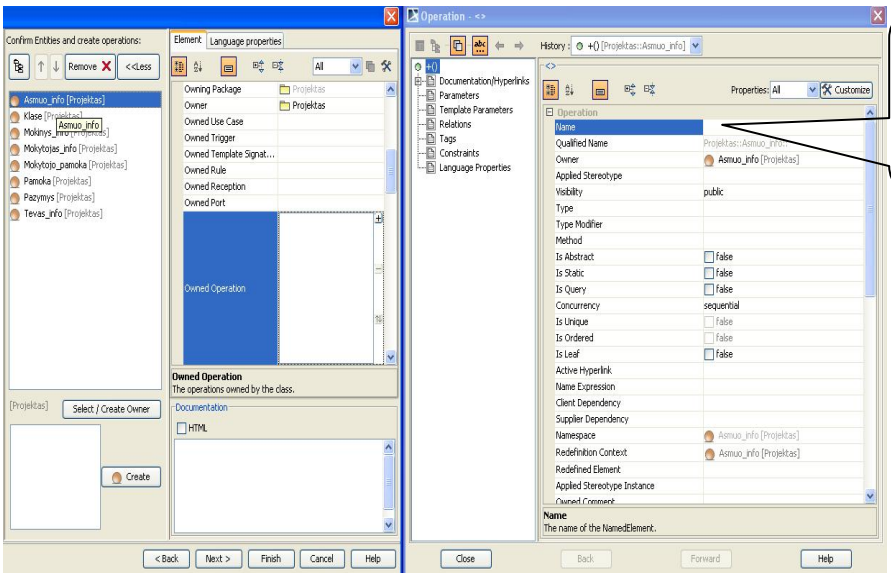
1) sukurti_info();

2) gauti_info();

3) ištrinti_info();

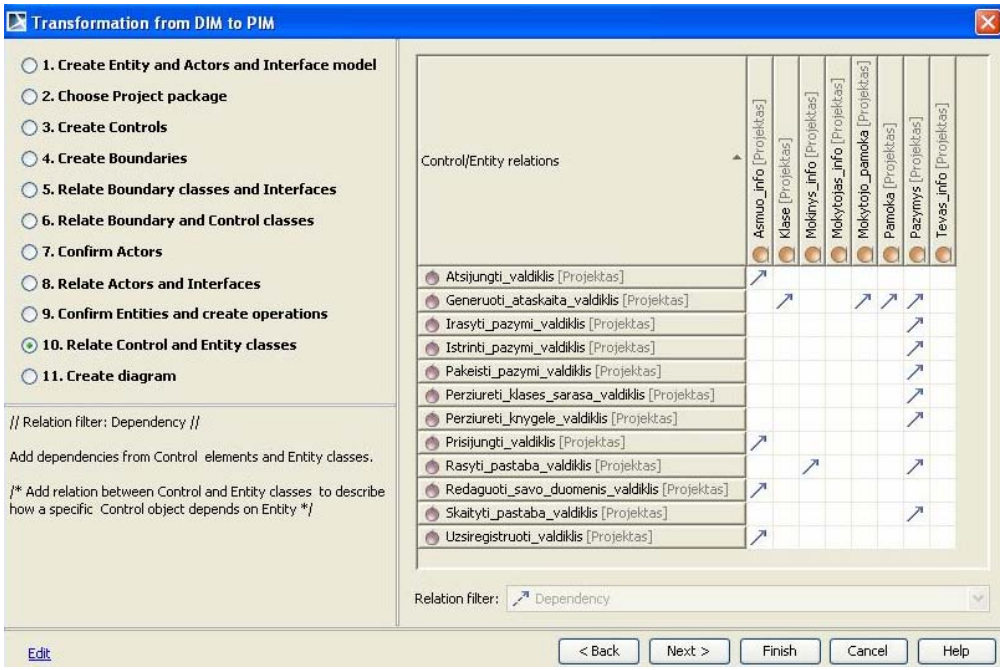
4) atnaujinti_info()

Tai atliekama klasių redagavimo meniu skiltyje „Owned operations“ įvedus operacijos pavadinimą (106 pav).



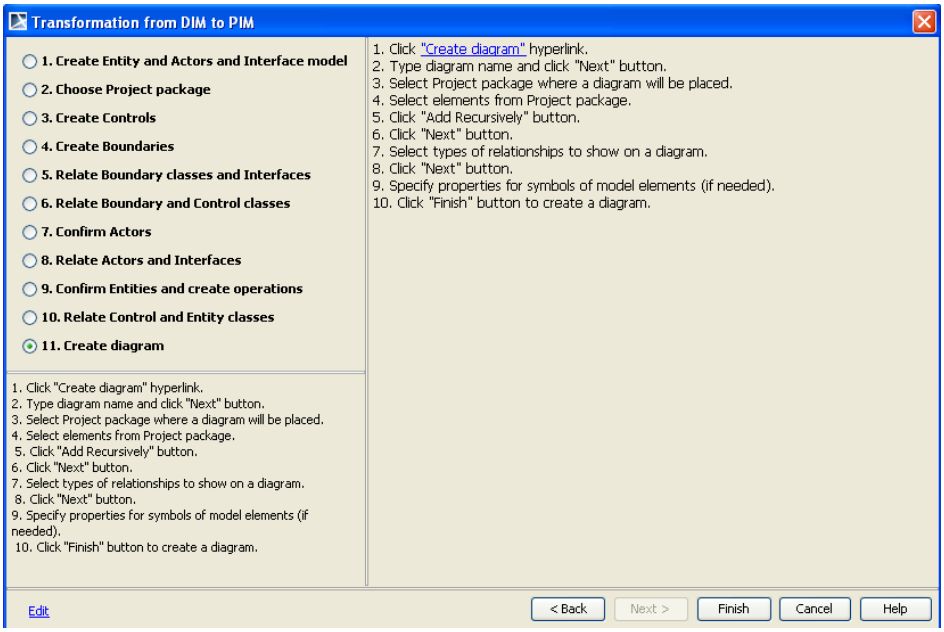
106 pav. Ribinių klasių redagavimo langas

Nurodomi ryšiai tarp valdiklio ir ribinių klasių (107 pav.)



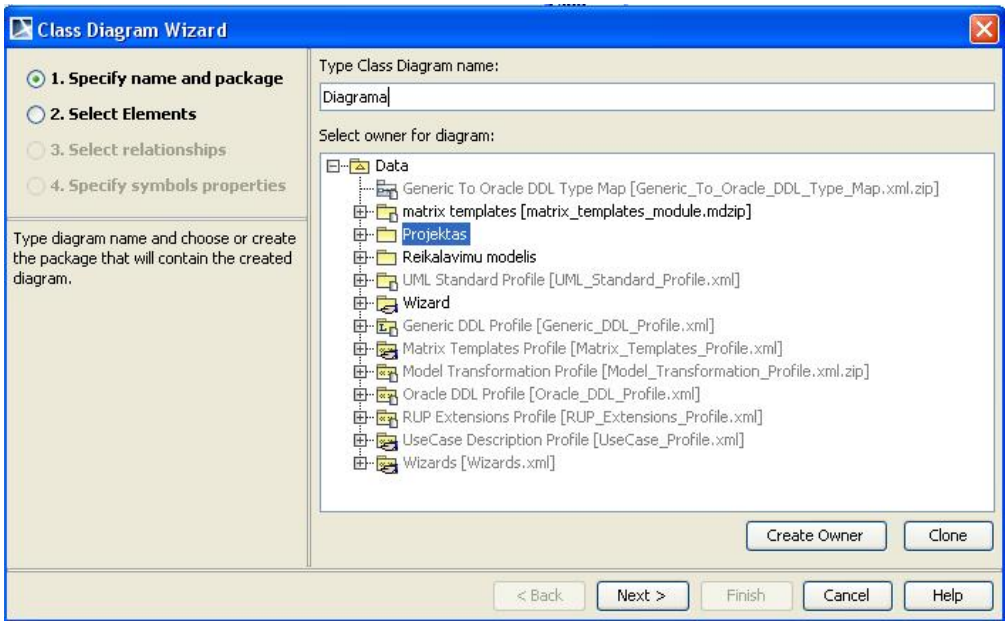
107 pav. Ryšių tarp valdiklio klasių ir esybių nurodymo langas

Kuriama projekto modelio elementų (klasių, aktorių, interfeisų ir ryšių tarp jų) diagrama (108 pav.).



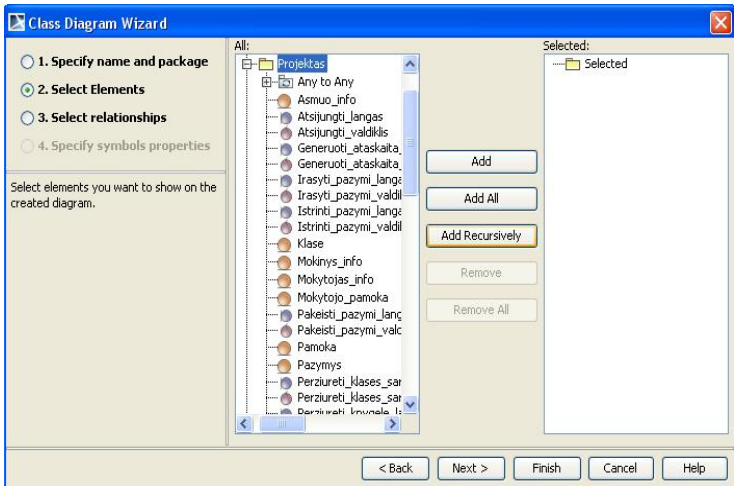
108 pav. Projekto klasių diagramos kūrimo langas

Įvedamas diagramos pavadinimas ir nurodomas projekto paketas, kuriame bus talpinama sugeneruota klasių diagrama (109 pav.):



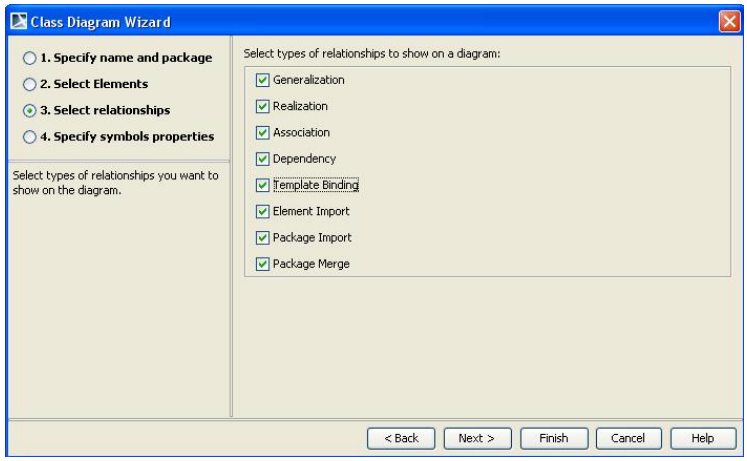
109 pav. Projekto paketo pasirinkimo langas

Vedlio lange pasirenkami elementai, kuriuos norima atvaizduoti projekto klasių diagramoje (110 pav.):



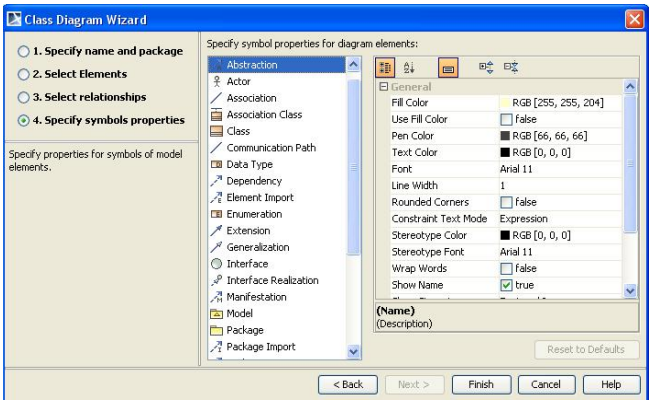
110 pav. Projekto klasių diagramos elementų pasirinkimo langas

Nurodomi ryšių tipai, kuriuos norima atvaizduoti klasių diagramoje (111 pav.):



111 pav. Diagramos ryšių tipų pasirinkimo langas

Diagramos elementų savybes galima redaguoti 112 pav. pateiktame lange, nurodant (pasirenkant) jų spalvą, dydį ir kitas charakteristikas.



112 pav. Diagramos elementų grafinių charakteristikų nustatymo langas