KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Linas Kozlovskis
Raimondas Mačionis
Žydrūnas Alšauskas

Komponentinio IS modelio transformavimo sistema

Magistro darbas

Darbo vadovas:
prof. S. Gudas

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Linas Kozlovskis
Raimondas Mačionis
Žydrūnas Alšauskas

Komponentinio IS modelio transformavimo sistema

Magistro darbas

Recenzentas
doc. dr. V. Sekliuckis
2008-01-14

Vadovas
prof. S. Gudas
2008-01-14

Atliko
IFM–2/4 gr. stud.
Linas Kozlovskis
Raimondas Mačionis
Žydrūnas Alšauskas
2008-01-14

Kaunas, 2008

COMPONENT – BASED MODEL TRANSFORMATION SYSTEM

SUMMARY

Presented work covers an approach to applications development based on the principles of the model-driven architecture and using the component-based system model (CBSM). The CBSM helps to refine main components and interfaces of the application at the design stage. The information system's architecture is structured considering a business system as a set of different domains (Business, Data, Information process) with definite types of components, and with interfaces between the components of different types.

Presented work is topical, when creates the same information system's components. These components can be used and they can be modified or changed.

Component - based model transformation system is created and tested with special project.

TURINYS

LENTELĖS

PAVEIKSLAI

1. ĮVADAS

Informacinės technologijos – viena iš sparčiausiai pastaruosius keletą dešimtmečių besivystančių mokslo šakų. Nors ji yra labai pažengusi ir taikoma daugelyje sričių, tačiau pasitaiko problemų, kurias spręsti ne taip lengva. Projektavimui palengvinti naudojamas komponentinis modeliavimas.

Šio darbo tikslas:

- Integruoti veiklos modelį ir detalų IS projekto modelį, panaudojant komponentinį sistemos projektavimo metodą.
- Panaudojant MagicDraw UML 12.5 paketo galimybes sukurti metodiką, profailus bei panaudojant MS Visual Studio 2005 sukurti programinę įrangą, kuri įgalina:
    - Transformuoti komponentinio IS modelio dalį, kuri aprašo vartotojo sąsają, į detalų vartotojo sąsajos objektinį modelį (**Linas Kozlovskis**).
    - Transformuoti komponentinio IS modelio dalį, kuri aprašo duomenis, į sistemos duomenų modelį (**Raimondas Mačionis**).
    - Transformuoti komponentinio IS modelio dalį, kuri aprašo procedūrų logiką, į sistemos procedūrų logikos modelį (**Žydrūnas Alšauskas**).

Darbas aktualus tuo, kad kuriant informacines sistemas, tenka kurti tuos pačius sistemos komponentus, o sistemą realizuojant komponentinio modeliavimo principu, galima panaudoti jau sukurtus komponentus juos papildant, susiaurinant bei pagal poreikius koreguojant.

Komponentinio modelio transformavimo uždavinio analizės dalyje įsigilinama į komponentinio modelio sudarymo subtilybes, bandoma išsiaiškinti, kaip komponentinis modelis yra susijęs su UML modeliais, iš kokių dalių jis yra sudarytas.

2. KOMPONENTINIO MODELIO TRANSFORMAVIMO UŽDAVINIO ANALIZĖ

2.1. Tyrimo sritis, objektas ir problema

Darbo sritis: komponentinis modeliavimas, objektinis programavimas;

Darbo objektas: komponentinis IS modelis;

Problema: komponentinio IS modelio atitinkamų lygmenų transformavimas į:

- detalų vartotojo sąsajos objektinį modelį,
- sistemos duomenų modelį,
- procedūrų logikos modelį.

2.2. Analizės tikslas

Norint geriau suprasti realizuojamos sistemos prasmę, reikalavimus bei kontekstą, įsigilinama į komponentinio modelio sudarymo subtilybes, bandoma išsiaiškinti, kaip komponentinis modelis yra susijęs su UML modeliais, iš kokių dalių jis yra sudarytas. Pateikiama kelių panašių sistemų analizė, tam tikrų modelių transformavimo į objektinius modelius metodika.

2.3. Problemos sprendimo metodų literatūros šaltiniuose analizė

Pagal [1] literatūros šaltinį, modeliais pagrįsta architektūra (MDA) (Model Driven Architecture) – tai OMG (Object Management Group) iniciatyva, kuri pateikia efektyvaus programinės įrangos modelių kūrimo ir panaudojimo strategijas. MDA apibrėžia tokį sistemų specifikavimo būdą, kuris atskiria sistemos funkcionalumo specifikaciją nuo sistemos realizavimo specifikacijos tam tikrai technologinei platformai. MDA nėra nauja architektūra – tai nauja programinės įrangos modelių kūrimo strategija. MDA tikslas – ne vieno uniforminio standarto įdiegimas, o aiškus skirtingų abstrakcijos lygių atskyrimas. MDA architektūros svarbiausios dalys yra skirtingų abstrakcijų lygių modeliai ir transformacijos tarp jų. MDA naudoja modeliavimo kalbas kaip programavimo kalbas [1].

2.4. Architektūros analizė

2.4.1. Architektūra grindžiamas IS projektavimas

Viena iš pažangiausių veiklos procesų ir taikomųjų programų integravimo metodologijų vadinama „architektūriniu modeliavimu“ ar „architektūra grindžiamas IS projektavimas“ (architecture-driven).

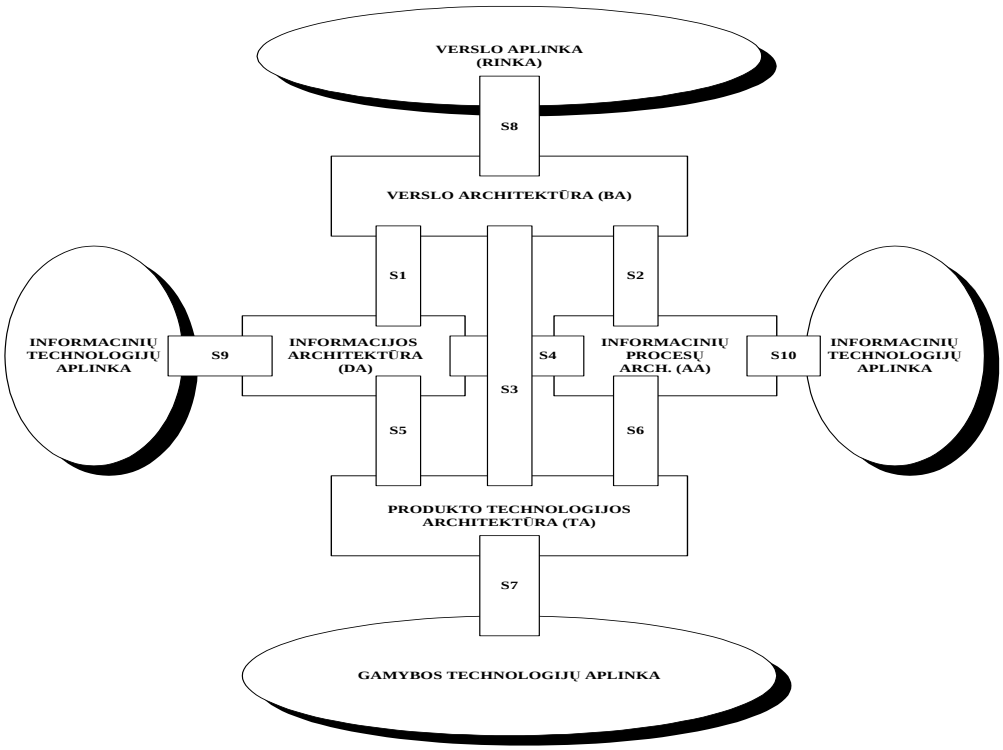
Veiklos informacinė architektūra apima bendros sistemos struktūros, sistemos komponenčių, loginių jų ryšių ir išoriškai matomų savybių modeliavimą (projektavimą).

2.4.2. Organizacijos veiklos informacinės architektūros modelis

Organizacijos veiklos informacinės architektūros (VIA) modeliavimas skirtas informacijos sistemų, atitinkančių realius veiklos poreikius, projektavimo ir realizavimo metodams plėtoti [2]. Organizacijos veikla gali būti nagrinėjama iš skirtingų pozicijų, išskiriant skirtingo pobūdžio veiklos dalykines sritis, vadinamas veiklos domenais. Veiklos domenai nurodo organizacijos dalis, kuriose vyksta skirtingos prigimties procesai.

Organizacijos veiklos informacinės architektūros (VIA) modelis, sudarytas iš keturių domenų, pateiktas 1 paveiksle.

IS projektavimo eigoje sudaromas kiekvieno domeno architektūros modelis.



1 pav. Organizacijos veiklos informacinės architektūros (VIA) modelis

Pagrindiniai organizacijos veiklos domenai, kurių visuma ir sąveikos užtikrina organizacijos funkcionavimą, aprašyti 1 lentelėje.

1 lentelė. Pagrindiniai organizacijos veiklos domenai

Veiklos domenas	Žymėjimas	Domeno paskirtis
1. Verslo procesų domenas	BD	Tai ekonominę ir gamybinę veiklą vykdančios organizacijos dalies (valdymo funkcijos, ekonominė veikla) informaciniai poreikiai ir reikalavimai IS;
2. Informacijos domenas	DD	Tai duomenys, žinios ir tikslai, jų saugojimo ir perdavimo organizacijos padaliniams procesai;
3. Informacinių procesų domenas	IPD	Organizacijoje atliekami skaičiavimai, sprendimo priėmimo procesai, galima vadinti taikomųjų uždavinių domenu;
4. Technologinių procesų domenas	TPD	Tai organizacijos dalies, atliekančios produkto gamybą (ar formavimą) - produkto gamybos procesų informaciniai poreikiai ir reikalavimai IS
5. Darbo vietų domenas	DVD	Darbo vietų visuma, informaciniai reikalavimai darbo vietose atliekamoms funkcijoms. Darbo vietų domenas yra pasiskirstęs, t.y. darbo vietos išsidėstę kituose domenuose ir sąsajose

Pirmieji keturi domenai yra pagrindiniai. Literatūroje yra minimas dar vienas (išvestinis) organizacijos veiklos domenas – darbo vietų (KDV) domenas. Darbo vietų domenas - tai organizacijos suvokimas kaip darbo (veiklos) vietų visumos, aprašant reikalavimus darbo vietose vykdomoms funkcijoms.

Kiekvienas domenas žymi specifinę organizacijos veiklos sritį (biznio procesus, informaciją, informacijos apdorojimo procesus, produkto gamybos procesus), kuri modeliuojama kaip savarankiškas objektas (komponentė).

2.4.3. Veiklos informacinės architektūros domenų sąsajų tipai

Veiklos domenai sąveikauja tarpusavyje. Domenų sąsajų paskirtis yra integruoti domenų sąveikas, siekiant organizacijos tikslų. Domenų sąsajų architektūros modelis yra gaunamas iš informacijos, surinktos apie domenus, t.y. yra išvedamas iš domenų informacinės architektūros modelio. Domenų sąsajos yra šių domenų informacinės architektūros (IA) komponentų sąsajos (2 lentelė).

Domenų sąsajų tipai aprašyti 2 lentelėje. Sąsajų S1 – S6 paskirtis yra integruoti domenų tarpusavio sąveikas, sąsajų S7 – S10 paskirtis - užtikrinti domenų sąveikas su išorine aplinka.

2 lentelė. Sąsajų tarp veiklos domenų tipai

Sąsajos tipas	Nurodyto tipo sąsajos siejami komponentai
S1	Duomenų domeno (duomenų komponentų) ir verslo domeno (verslo komponentų) sąsaja
S2	Informacinių procesų domeno (funkcinių komponentų) ir verslo domeno (verslo IA komponentų) sąsaja
S3	Technologinių procesų domeno (TP IA komponentų) ir verslo domeno (komponentų) sąsaja
S4	Informacinių procesų domeno (funkcinių komponentų) ir informacijos domeno (komponentų) sąsaja
S5	Technologinių procesų domeno (TP IA komponentų) ir informacijos domeno (duomenų komponentų) sąsaja
S6	Informacinių procesų domeno ir technologinių procesų domeno (TP IA komponentų) sąsaja
S7	Technologinių procesų domeno sąsaja su produkto gamybos technologijų aplinka (aplinkos IA komponentais).
S8	Verslo domeno sąsajos su verslo aplinka
S9, S10	Informacinių procesų domeno ir informacijos domeno sąsajos su informacinių technologijų aplinka (aplinkos IA komponentais)

Kiekviena domenų (t.y. jų atitinkamų komponentų) sąsaja modeliuojama kaip atskiras objektas - atskira IS architektūros komponentė, siejanti du konkrečius VIA domenus.

Skirtingų domenų tarpusavio informacinės sąveikos užtikrinimas, taip pat pagrindinių domenų sąveikos su išorine aplinka realizavimas ir reiškia organizacijos veiklos integravimą.

2.4.4. Sąsajos atskyrimas nuo realizacijos

Sąsajomis grįstas programavimas arba objektų kompozicija leidžia naudoti pakartotinį panaudojimą, be tvirtų sujungimų. Objektų kompozicijos pagrindas yra tai, kad klasės metodų realizacijos detalės nėra parodomos klientui. Klientas žino tik aibę galimų užklausų (kas). Objektai niekada neparodo vidinių atsakymo detalių (kaip). Sąsajomis grįsto programavimo pakartotinio panaudojimo pagrindas – sąsajos atskyrimas nuo realizacijos. Sąsaja yra nepriklausomas duomenų tipas, kuris aprašo pats save.

Bendruoju atveju sąsaja yra aibė viešų metodų aprašų. Ji apibrėžia iškvietimo sintaksę, aibei logiškai susijusių kliento užklausų. Metodų aprašai negali turėti jokios realizacijos, todėl sąsaja gali atskirti klasę nuo kliento, kuris ją naudoja. Sąsaja turi buti realizuota vienos ar daugiau klasių.

Klasei realizavus sąsają, klientas gali sukurti objektą iš klasės ir bendrauti su ja per sąsajos aprašą. Galime naudoti sąsają norėdami sukurti kreipinį į objektą, bet negalime jos naudoti kaip objekto. Objektas turi turėti duomenų savybes ir metodų realizacijas, kurių negali pateikti sąsaja. Sąsaja yra abstraktus duomenų tipas.

Projektavimo požiūriu sąsaja yra kontraktas. Klasė, kuri realizuoja sąsają, garantuoja, kad objektas atitiks tam tikra elgseną. Klasė turi pateikti realizaciją kiekvienam metodui aprašytam sąsajoje. Bendraujant su objektu per sąsajos aprašą, klientas gali buti tikras, kad objektas pateiks prasmingą atsakymą kiekvienam metodui aprašytam sąsajoje. Daugiau nei viena klasė gali realizuoti tą pačią sąsają. Sąsaja apibrėžia tikslią kvietimo sintaksę, bet neturi metodo semantikos. Nenurodyta semantika leidžia klasės autoriui laisvai apibrėžti konkretaus objekto metodo elgseną [4].

2.5. Komponentinio IS projektavimo metodo principai

Komponentinis projektavimas teoriškai turi daug privalumų, iš kurių svarbiausias – pakartotino komponentų panaudojimo galimybė. Dėl šios savybės padidėja produktyvumas, palaikymo ir modifikavimo galimybės, o lygiagrečiai sumažėja projekto kūrimo ciklas ir kaštai.

IS projekto lygmens komponentai projektuojami pagal modeliu pagrįstą (*model-driven*) projektavimo paradigmą, kurioje komponentai paveldi aprašus iš veiklos proceso

modelio. IS komponentai turi būti visiškai save aprašantys. Tai reiškia, kad IS komponentas turi aiškiai apibrėžtą interfeisą ir atitinka nurodytą elgseną, bendrą visiems sistemos architektūros vidaus komponentams.

Aptariamas metodas aprašo architektūrinio IS projektavimo etapą, kuriame identifikuojami IS projekto komponentai ir jų sąsajos (interfeisai).

Toliau, detalaus projektavimo etape, komponentai turi būti specifikuojami, parengiant projektą IS programinės įrangos generavimui.

2.5.1. Komponentas

Komponentinio projektavimo požiūriu informacinės sistemos projekto komponentai yra skirstomi į:

- **vartotojo sąsajos komponentus** (meniu, ekrano formos, ataskaitos),
- **duomenų komponentus** (duomenų bazėse ar duomenų saugykloje talpinami informacijos vienetai),
- **funkcinius komponentus** (skaičiavimai ir taikomųjų uždavinių logika).
- Šie IS komponentai susieti **interfeisais** (sąsajomis), kuriuos galima laikyti **IS komponentais** taip pat.

IS projekto komponentus identifikuoja projektuotojas, CASE sistemos aplinkoje analizuodamas veiklos modelį (pvz., darbų sekų modelį, kuris aprašo konkrečią veiklos funkciją ar procesą).

Taip projektuotas sudaro komponentinį sistemos modelį, kuris aprašo identifikuotus IS komponentus ir jų sąveikas. Toliau sudaromi žemesnių lygmenų komponentiniai sistemos modeliai, taip tikslinama IS komponentų sudėtis ir specifikacijos.

Detalaus IS komponentų specifikavimo etape gali būti naudojami atitinkami objektiniai modeliai (pvz.: UML).

IS projekto komponentus realizuoja programinės įrangos lygmens komponentai. Programinės įrangos lygmens komponentas yra programinės įrangos objektas, sąveikaujantis su kitais komponentais, atliekantis tam tikrą funkciją ar aibę funkcijų. Komponentų valdymo ir funkcionavimo optimizavimo priemonės naudoja vieningą komponentų aprašų saugyklą.

Objektinio modeliavimo standartas UML (UML2 versija) apibrėžia sistemos komponentą kaip klasės atvejį, kurios specifikacijoje yra apibrėžti atributų, metodų ir interfeisų sluoksniai, t.y. atsiranda papildoma klasės savybių grupė (greta atributų ir metodų) – interfeisai.

Komponentas yra projektuojamos informacines sistemos dalis. Komponentas duri dvi aiškiai atskirtas dalis: sąsają ir funkcionalumą. Per sąsają komponentas sąveikauja su aplinka, su kitais komponentais. O funkcionalumas aprašo ką komponentas daro ar kaip jis yra sudarytas.

Komponentas gali buti sudarytas iš kitų komponentų.

Komponentai dar skirstomi į bendrinius komponentus (apimančius visos panašių komponentų klasės (šeimos) konkrečias savybes), bei į egzempliorius, turinčius griežtai suformuotas konkrečias savybes.

Egzemplioriai gaunami iš bendrinių komponentų, sukonkretinus jų savybes ir funkcijas.

Bendrinio komponento modelis yra nagrinėjamas dviem aspektais: funkciniu ir architektūriniu.

Funkcinį bendrinio komponento modelį sudaro bendrinė sąsaja ir bendrinis funkcionalumas. Reikia griežtai skirti sąsają ir funkcionalumą, nes tada galima paslėpti realizacijos detales nuo vartotojo ir šitaip pakelti abstrakcijos lygmenį.

Architektūrinis komponento modelis apima monolitinę (funkcionalumas aprašomas viename monolitiniame funkciniame bloke) ir hierarchine architektūrą. Hierarchinis architektūros modelis leidžia sukomponuoti sudėtingą bendrinį komponentą iš kelių dalių, kurios savo ruožtu gali turėti hierarchinius arba monolitinius architektūros modelius. Svarbu sukurti lanksčią architektūrą: kai kurie komponentai gali buti pakartotinai panaudoti.

Norint iš bendrinio komponento gauti egzempliorių, reikia per sąsają perduotus reikalavimus paversti matavimais ir jiems priskirti reikšmių sritis.

Bendrinio komponento projektavimo apibendrinimo mechanizmai:

1. platinimas (apibendrinti bendrinį komponentą išplečiant taikymo sritį);
2. siaurinimas (bendrinio komponento vartojimo sritis išplečiama mažinant funkcionalumą);
3. izoliavimas (visa bendrinį komponentą galima skaidyti į keletą paprastesnių komponentų, kurie ne tokie sudėtingi);
4. konfigūravimas (vietoj vieno sudėtingo, atitinkančio visus reikalavimus geriau sukurti aibe mažesnių, kurie gali buti tarpusavyje konfigūruojami).

Toliau, detalaus projektavimo etape, komponentai turi buti specifikuojami, parengiant projektą informacinės sistemos programinės įrangos generavimui.

Daugiausiai sunkumų sukelia objektai ir ryšiai tarp jų. Jei objektas neturi bendrų ryšių, reiktų jį išskirti kaip atskirą modelį. Iš esmės reikia optimizuoti objektų ir ryšių tarp jų

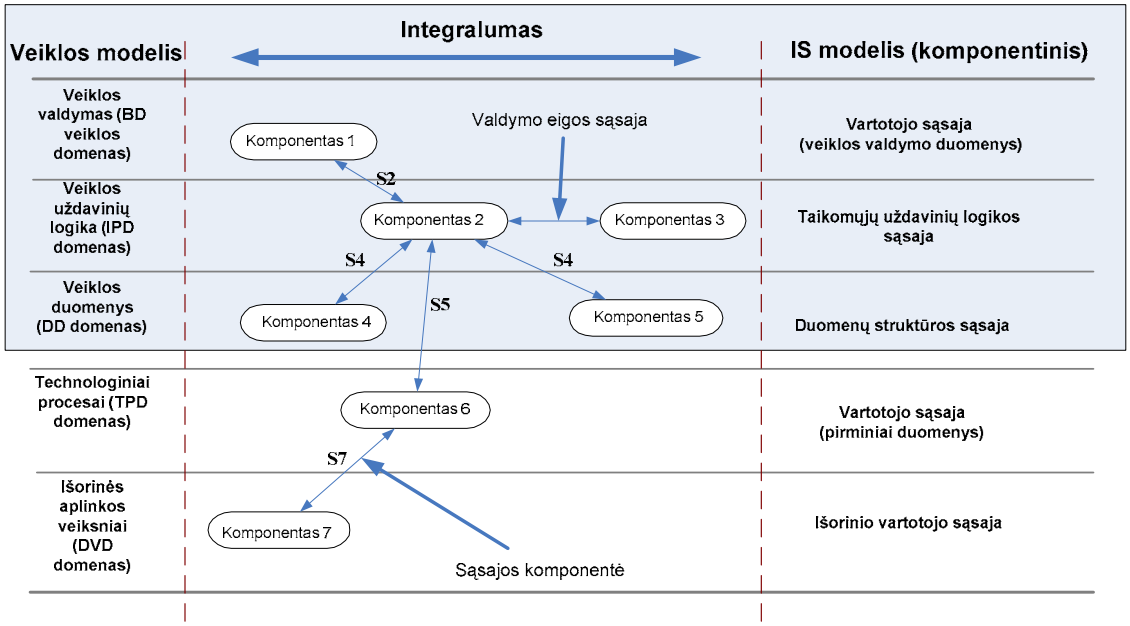
skaičių, tai yra neišskaidyti sistemos į per daug komponentų, tačiau tuo pačiu, jei bus per mažai komponentų, bus painūs ryšiai, o ir pats ryšys per daug sudėtingas ir daugialypis.

2.5.2. Komponentinio IS modelio sudėtis

Organizacijos informacijos sistemos komponentams ir sąsajoms tarp jų identifikuoti siūloma nauja grafinė notacija – komponentinis IS modelis. Šis modelis apjungia veiklos informacinės architektūros (VIA) modelio ir darbų sekos modelio savybes.

Veiklos informacinės architektūros modelis apibrėžia IS komponentų tipus, atitinkančius organizacijos veiklos domenus, kurie aprašyti 1 lentelėje. Remiantis tuo, komponentinis sistemos modelis (analogija su darbų sekų modeliu) skirstomas į penkis takelius, kurie skirti atitinkamo vieno veiklos domeno komponentams (2 pav.):

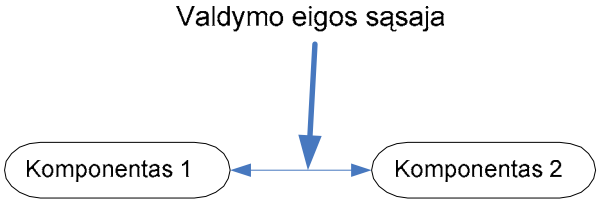
- takelis „Veiklos valdymas (BD veiklos domenas)“ atitinka verslo domeną ir skirtas šiame domene naudojamiems IS komponentams (tai IS vartotojo sąsajos komponentai) specifikuoti;
- takelis „Veiklos uždavinių logika (IPD domenas)“ atitinka informacinių procesų domeną ir skirtas IS taikomųjų uždavinių logiką (skaičiavimus ir kitokį duomenų apdorojimą) realizuojantiems komponentams (tai IS funkciniai komponentai) specifikuoti;
- takelis „Veiklos duomenys (DD domenas)“ atitinka informacijos domeną ir skirtas IS saugyklose (duomenų bazėse, duomenų sandėliuose) saugomos informacijos elementams, t.y. duomenų komponentams specifikuoti;
- takelis „Technologiniai procesai (TPD domenas)“ atitinka technologinių procesų domeną ir skirtas šiame domene naudojamiems IS komponentams (tai IS vartotojo sąsajos komponentai) specifikuoti;
- takelis „Išorinės aplinkos veiksniai (DVD domenas)“ atitinka VIA modelio aplinkos domenus (verslo rinkos, technologijų ir informacinių technologijų rinkos) ir skirtas šiuose domenuose esantiems aktualiems komponentams (sąveikaujantiems su jau aptartais IS komponentais) specifikuoti.



2 pav. Komponentinio IS modelio elementai

Komponentiniame sistemos modelyje informacijos sistemos komponentas vaizduojamas stačiakampiu suapvalintais kampais, sąsajos tarp komponentų žymimos rodyklėmis, šalia nurodomi sąsajų tipai. Sistemos komponentų sąsajų tipus aprašo 2 lentelė, kurioje yra įvertintos ir domenų sąveikos su išorine aplinka.

Komponentiniame sistemos modelyje gali būti nurodytos valdymo eigos sąsajos (3 pav.) (control flow – CF), kurios sieja vieno domeno komponentus ir nurodo priežastinius domeno komponentų ryšius. Pastebėsime, kad terminas „*valdymo eiga*“ naudojamas objektiškai orientuotame modeliavime ir reiškia proceso vykdymo valdymą.



3 pav. Valdymo eigos sąsaja

2.6. Nagrinėjamos informacinės sistemos analizė

Pasirinktoji organizacija yra restoranas. Pagrindinis veiklos procesas yra restorano lankytojų aptarnavimas. Restorane klientai gali užsisakyti maisto ir/arba gėrimų, padavėjai juos aptarnauja. Virtuvėje bei bare ruošia atitinkamai patiekalus bei gėrimus, kasoje apmokami užsakymai.

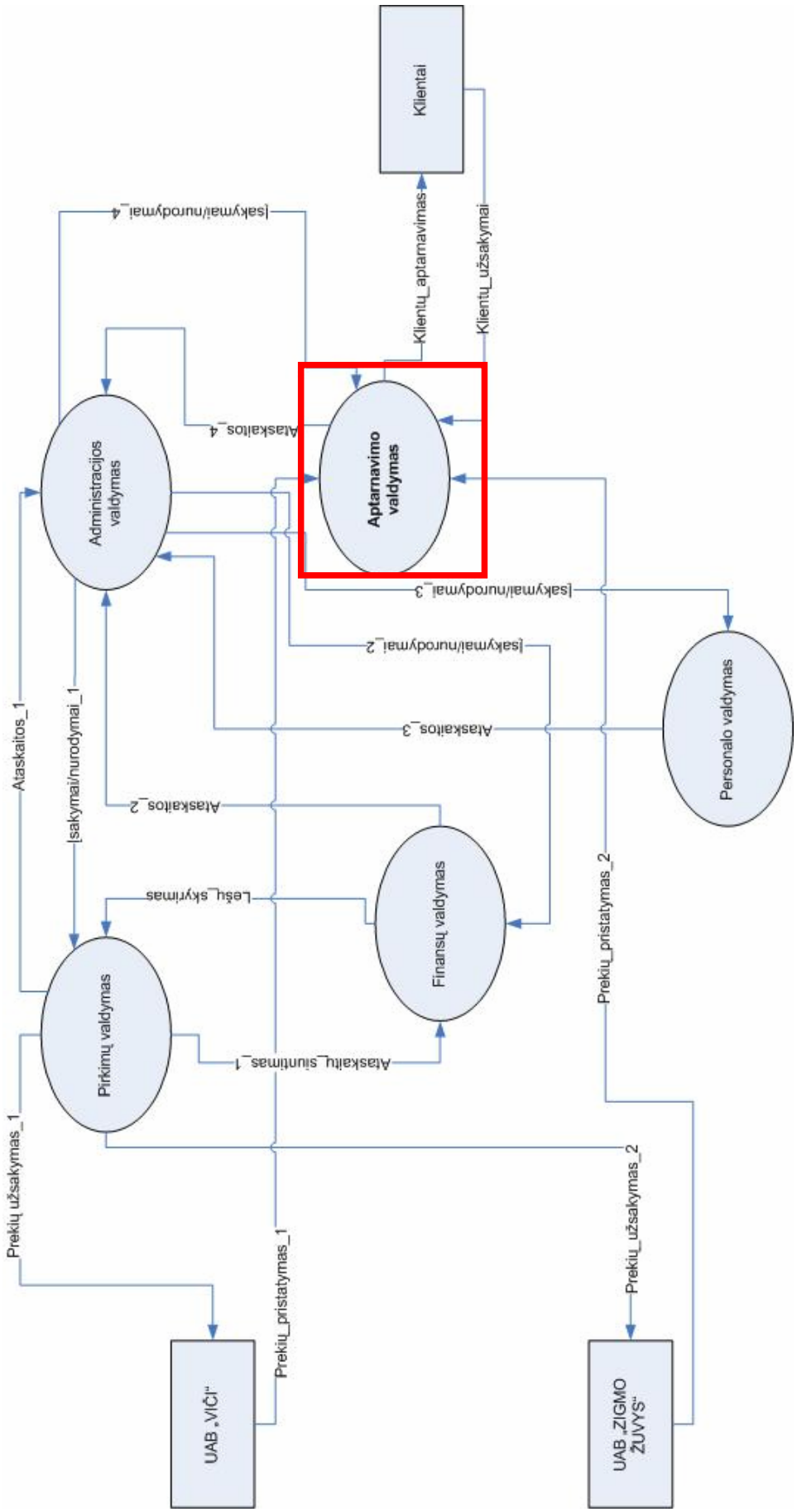
DFD – duomenų srautų diagramos skirtos probleminės srities funkciniam modeliui apibrėžti, t.y. sistemos funkcijoms (procesams) vaizduoti. Pagrindiniai DFD elementai:

- Duomenų srautai parodo, kurie duomenys naudojami procesuose, iš kur jie imami ir kur saugomi.
- Procesas transformuoja, perdirba duomenis.
- Duomenų saugykla – tai pasyvus duomenų saugotojas, neatliekantis jokiu duomenų transformacijų.
- Išorinis objektas tiekia sistemai reikiama informacija ir naudoja ją.

Nulinio lygmens DFD (4 pav.) nurodo pagrindinius analizuojamos veiklos (organizacijos) procesus ir juos siejančius srautus. Kaip matoma 4 paveiksle, modelio vidurinė sritis atstovauja vartotojo biznio veiklos dalį (business domain - biznio sfera) be to ši sritis svarbiausia modelyje. Ją sudaro:

- pirkimų skyrius;
- finansų skyrius;
- personalo skyrius;
- administracijos skyrius;
- aptarnavimo skyrius.

UAB „Viči“ bei UAB „Zigmo žuvys“ – vieni iš pagrindinių produktų tiekėjų restoranui.



4 pav. Restorano nulinio lygmens DFD

2.7. Veiklos objektų modelis

Projektuojamą sistemą būtų galima skaidyti į tris modulius:

- Komponentinio IS modelio transformavimo į sistemos procedūrų logikos modelį sistema.
- Komponentinio IS modelio transformavimo į sistemos duomenų modelį sistema.
- Komponentinio IS modelio transformavimo į detalų vartotojo sąsajos objektinį modelį sistema.

Į tai bus atsižvelgta realizuojant sistemą.

2.8. Sistemai keliami nefunkciniai reikalavimai

- Sistemos prototipas bus realizuotas naudojantis .NET technologija, C# objektine programavimo kalba bei „Visual Studio 2005“ aplinka.
- Komponentinio modelio sudarymui naudoti „MS Visio 2003“ bei „Magic Draw 12.5“ paketai.
- Workflow modeliui sudaryti naudotas „ProVision Workbench v3.1“ paketas.
- Sistemos veikimui būtinas „MS .NET Framework“.

2.9. Darbo tikslas

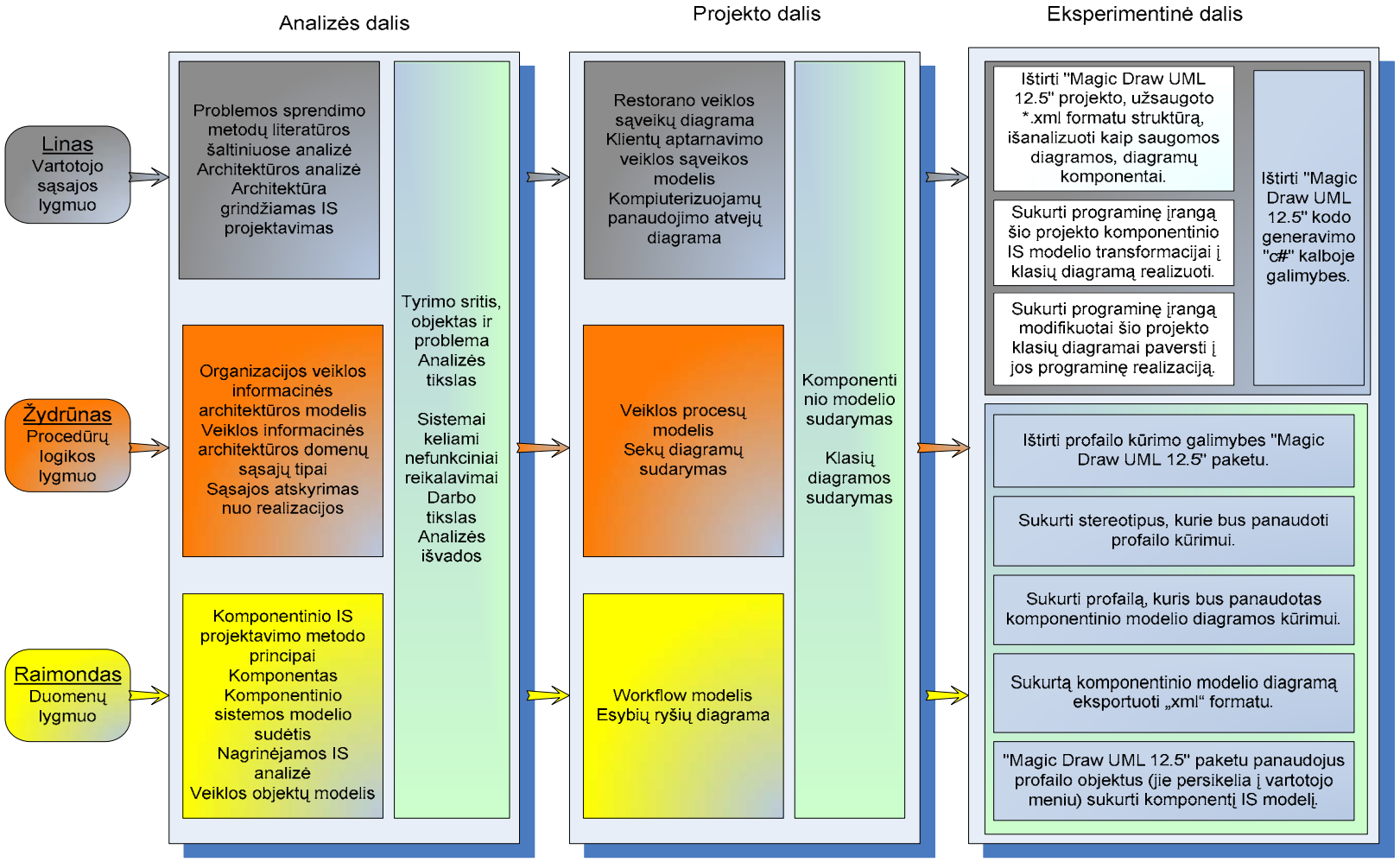
Integruoti veiklos modelį ir detalų IS projekto modelį, panaudojant komponentinį sistemos projektavimo metodą.

2.10. Darbo uždaviniai

- Panaudojant MagicDraw UML 12.5 paketo galimybes sukurti metodiką, profailus bei panaudojant MS Visual Studio 2005 sukurti programinę įrangą, kuri įgalina:
  - Transformuoti komponentinio IS modelio dalį, kuri aprašo vartotojo sąsają, į detalų vartotojo sąsajos objektinį modelį (**Linas Kozlovskis**).
  - Transformuoti komponentinio IS modelio dalį, kuri aprašo duomenis, į sistemos duomenų modelį (**Raimondas Mačionis**).
  - Transformuoti komponentinio IS modelio dalį, kuri aprašo procedūrų logiką, į sistemos procedūrų logikos modelį (**Žydrūnas Alšauskas**).
- Sudaryti komponentinio IS modelio transformavimo i UML klasių modelį algoritmą bei prototipą.
- Ištirti bei išanalizuoti komponentinį modelį.
- Pateikti konkretų IS projektavimo pavyzdį pagal sukurtą metodiką.

2.11. Darbų pasiskirstymas

5 pav. pavaizduota kokie darbai atlikti kiekvieno grupės nario. Bendrai atlikta analizės dalis. Projekto dalyje pasiskirstyta tam tikrais darbais: pagal atitinkamą lygmenį suprojektuotos diagramos. Komponentinis modelis bei klasių diagramos sudarytos apjungus visus tris lygmenis. Eksperimentinėje dalyje taip pat buvo pasiskirstyta atitinkamais darbais, kurie pavaizduoti darbų pasiskirstymo schemoje (5 pav.).



5 pav. Darbų pasiskirstymo schema

2.12. Analizės išvados

Projektuojant integruotas kompiuterizuotas informacijos sistemas tikslinga apjungti IS kūrimą informacinės architektūros modelio pagrindu bei komponentinį IS projektavimą, siekiantį surinkti IS iš kompiuterizuotų veiklos komponentų.

Pasiūlytas metodas susieja IS architektūros modelį, duomenų srautų diagramą ir atvaizduoja juose esančią informaciją į naujo tipo modelį – komponentinį sistemos modelį, kuriame išskiriami tokio tipo komponentai:

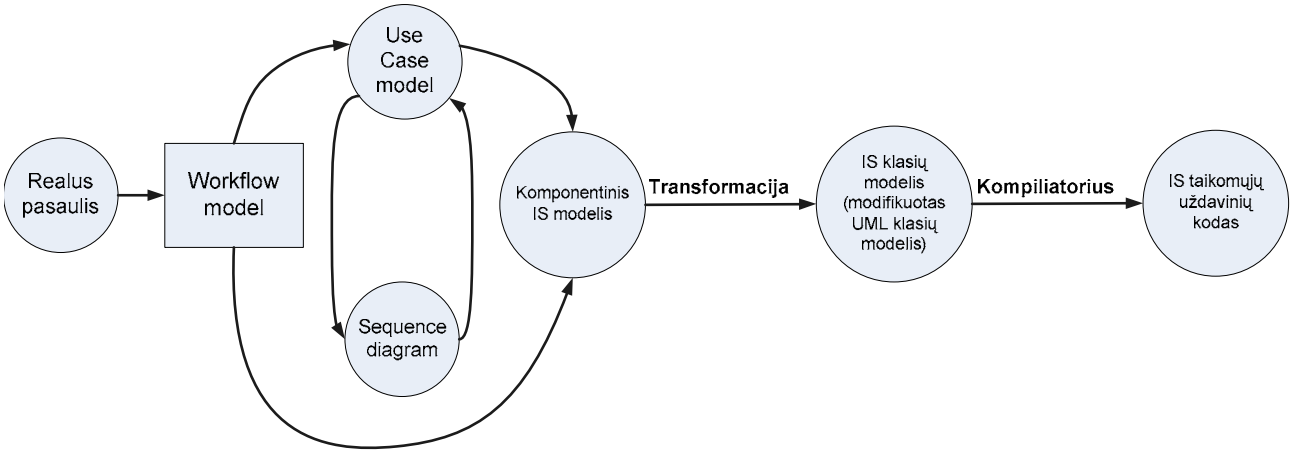
- valdymo funkcijos (BD);
- skaičiavimai arba funkciniai komponentai (IPD);
- duomenų struktūros (DD).

Kaip pavyzdys pasirinktas restoranas. Siekiant sukurti komponentų sąveikos modelį, parinktas procesas „Pateikti užsakymą“.

3. KOMPONENTINIO IS MODELIO TRANSFORMAVIMO SISTEMOS PROJEKTAS

3.1. Projekto eiga

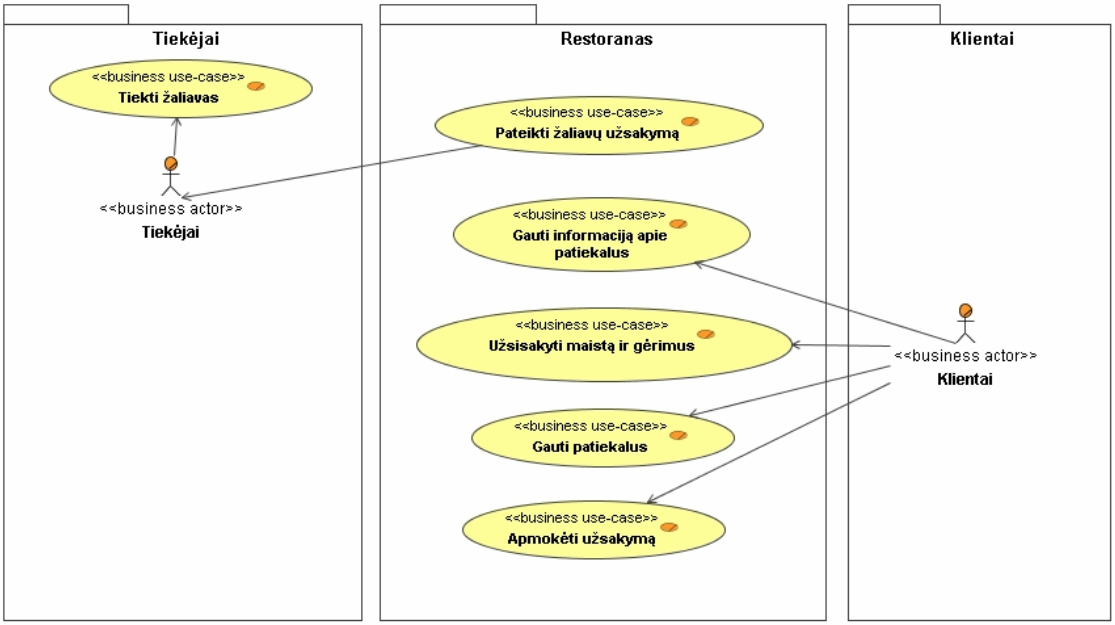
6 pav. matoma, jog komponentinis IS modelis gaunamas iš trijų modelių – IS sekų, IS darbų sekų bei IS panaudojimo atvejų modelių. Iš komponentinio IS modelio sukuriamas IS klasių modelis. Realizacija vykdoma naudojantis modifikuotu IS klasių modeliu.



6 pav. Projekto eiga

3.2. Restorano veiklos sąveikų diagrama

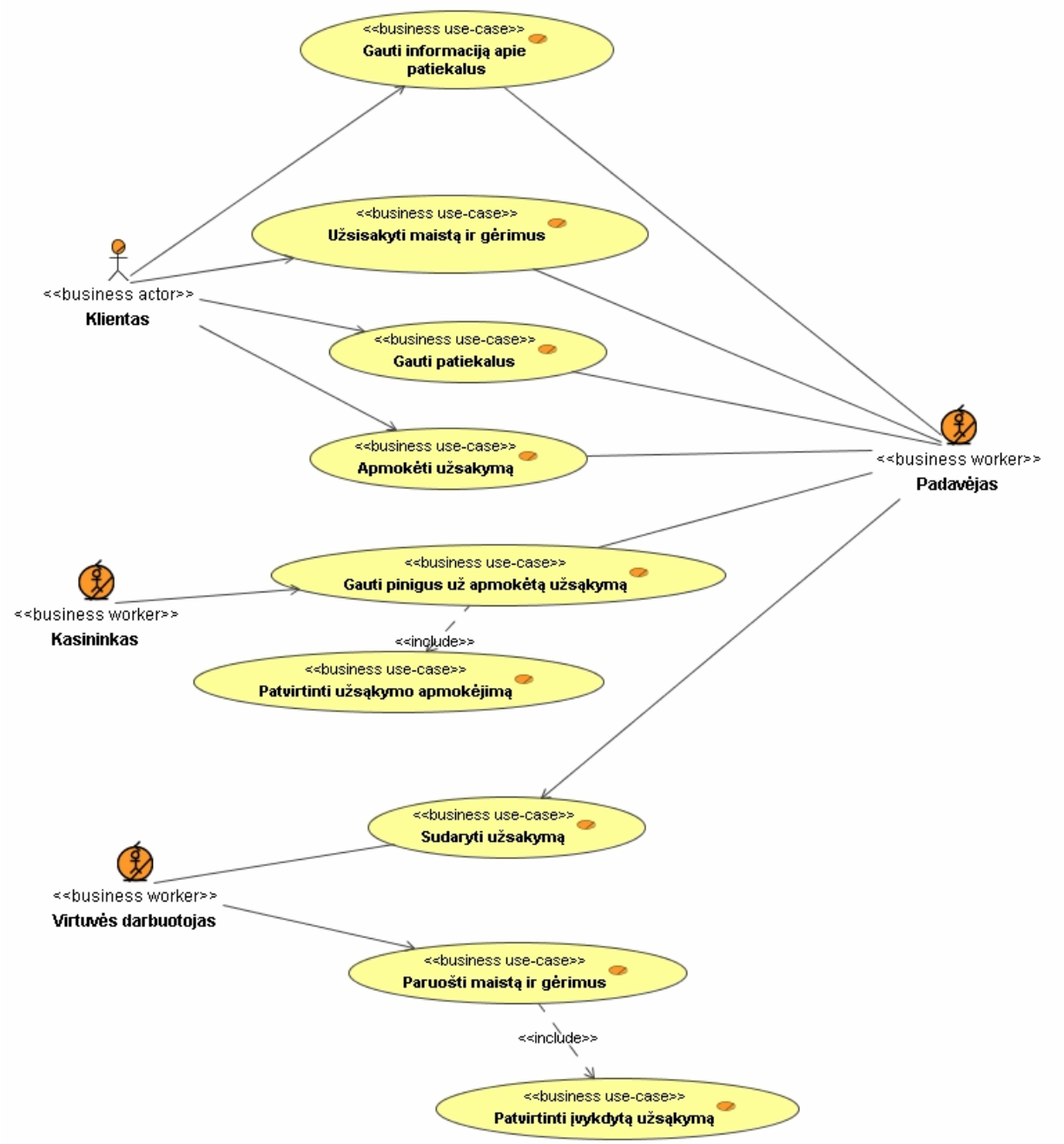
Restorano veiklos sąveikų diagrama pateikiama 7 paveiksle. Bendras restorano veiklos sąveikų modelis parodo tiekėjų, restorano ir klientų tarpusavio ryšį. Restoranas tiekėjams pateikia žaliavų užsakymus. Tiekėjai tiekia žaliavas. Klientai iš restorano gauna informaciją apie patiekalus, gali užsisakyti maistą ir gėrimus, gauna patiekalus bei apmoka užsakymą.



7 pav. Restorano veiklos sąveikų diagrama

3.3. Klientų aptarnavimo veiklos sąveikos modelis

Klientų aptarnavimo veiklos sąveikos modelis pateikiamas 8 paveiksle. Restorane klientui pateikiama informaciją apie patiekalus, jis gali užsisakyti maistą ir gėrimus, gauti patiekalus bei apmokėti užsakymą. Klientams visą informaciją suteikia bei juos aptarnauja padavėjas. Padavėjas taip pat pateikia užsakymo apmokėjimą kasininkui bei surado užsakymą, kurį paruošia Virtuvės darbuotojas. Kasininkas apmoka bei patvirtina užsakymo apmokėjimą. Klientų aptarnavimo veiklos sąveikos modelis parodo ryšį tarp restorano darbuotojų ir klientų.



8 pav. Klientų aptarnavimo veiklos sąveikos modelis

3 lentelė. Vartotojų funkcijos

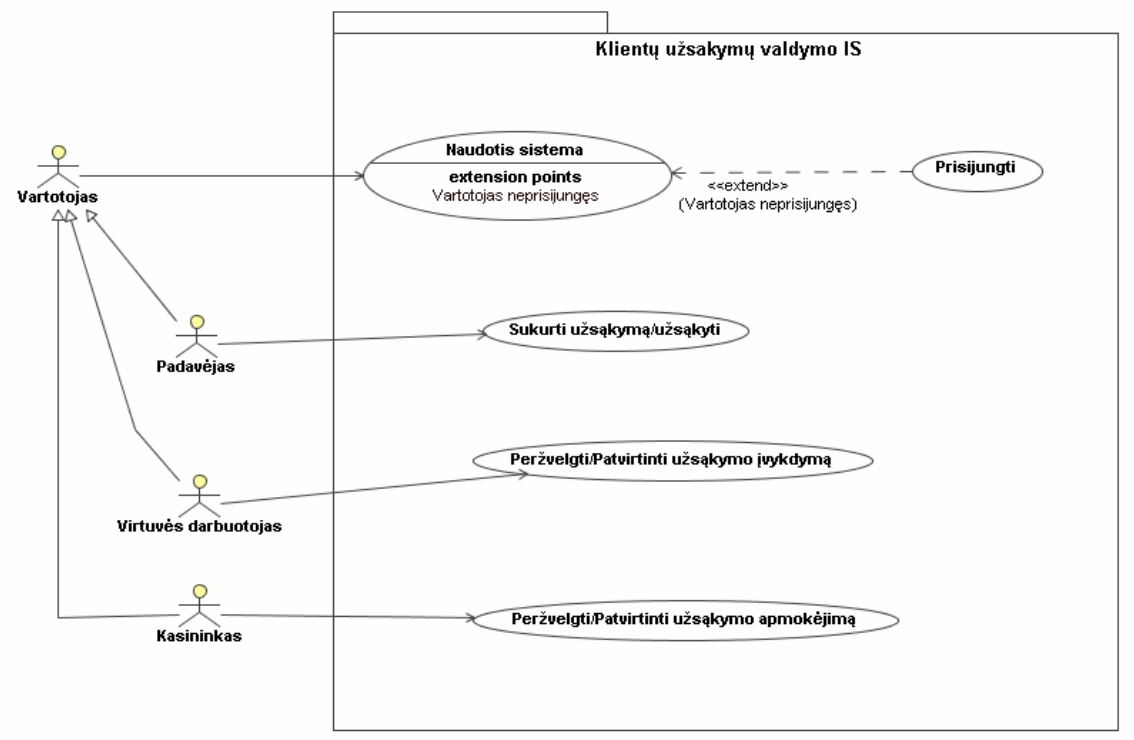
Atvejis	Klientas	Padavėjas	Kasininkas	Virtuvės darbuotojas
Gauti informaciją apie patiekalus	✅	✅	-	-
Užsisakyti maistą ir gėrimus	✅	✅	-	-
Gauti patiekalus	✅	✅	-	-
Apmokėti užsakymą	✅	✅	-	-
Gauti pinigus už apmokėtą užsakymą	-	✅	✅	-
Patvirtinti užsakymo apmokėjimą	-	-	✅	-
Sudaryti užsakymą	-	✅	-	✅
Paruošti maistą ir gėrimus	-	-	-	✅
Patvirtinti įvykdytą užsakymą	-	-	-	✅

3 lentelėje pateiktas palyginimas, kokią operaciją/veiksmus gali atlikti atitinkamas teises turintis vartotojas.

3.4. Kompiuterizuojamų panaudojimo atvejų diagrama

9 paveiksle pateikiama restorano klientų užsakymų valdymo informacinės sistemos kompiuterizuojamų panaudojimo atvejų diagrama. Pagrindiniai panaudojimo atvejai:

- „Sukurti užsakymą/užsakyti“,
- „Peržvelgti/patvirtinti užsakymo įvykdymą“,
- „Peržvelgti/ patvirtinti užsakymo apmokėjimą“.



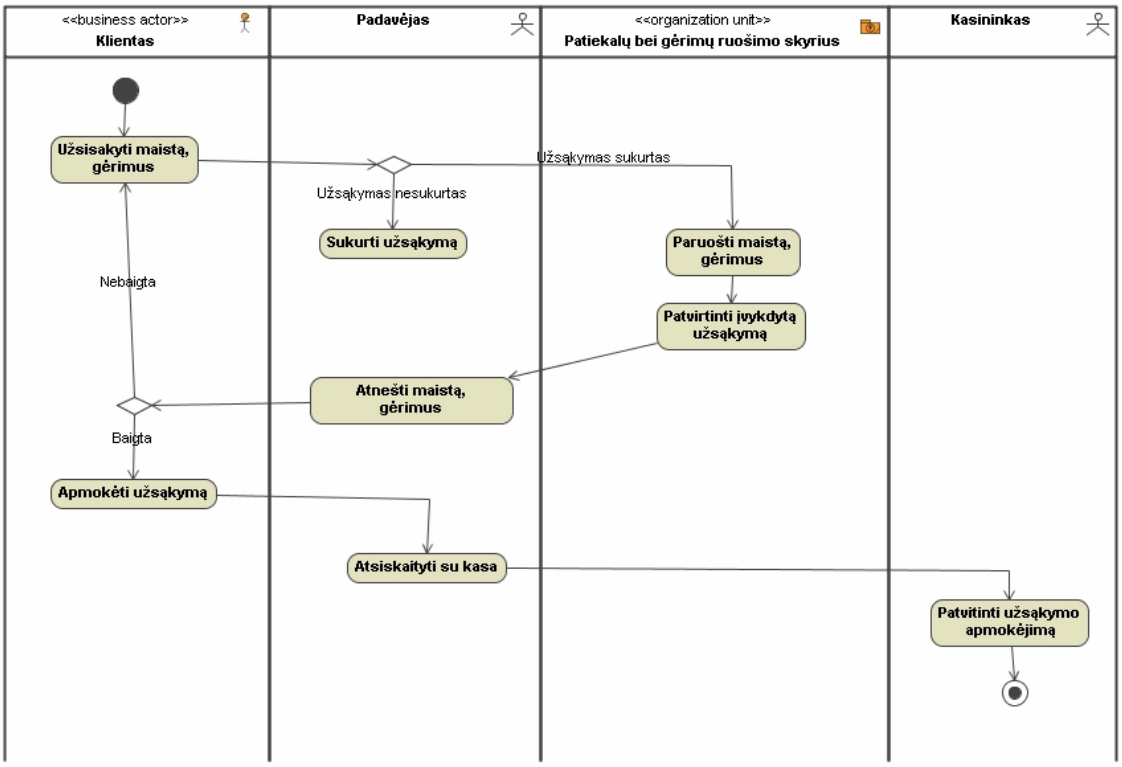
9 pav. Kompiuterizuojamų panaudojimo atvejų diagrama

3.5. Veiklos procesų modelis

Pagrindinis restorano veiklos procesas yra kliento aptarnavimas. Tokio proceso modelis parodytas 10 paveiksle. Šį modelį galima suskaidyti į keturias skirtingas grupes: klientas, padavėjas, patiekalų bei gėrimų skyrius (kitaip – virtuvė) ir kasininkas. Klientas neturi priėjimo prie užsakymo sistemos – užsakymą priima padavėjas, priėjęs prie staliuko. Atitinkama vartotojų grupė turi tik jiems skirtus prisijungimo vardus bei slaptažodžius. Atitinkamai skiriasi vartotojų teisės.

Užsakymo procesas prasideda klientui pakvietus padavėją. Klientas iš pateikto meniu išsirenka patiekalus, gėrimus. Padavėjas prisijungęs prie sistemos visų pirma sukuria

užsakymą ir tik po to siunčia patiekalų bei gėrimų ruošimo skyriui, kuris pagal pateiktą sąrašą paruošia patiekalus bei gėrimus. Kai užsakymas paruoštas patiekti klientui, užsakymas patvirtinamas. Gavus užsakymo patvirtinimą iš virtuvės padavėjas atneša patiekalus bei gėrimus klientui. Pateikiama sąlyga, ar tai galutinis užsakymas. Jeigu taip, tuomet klientas apmoka užsakymą, jeigu ne – klientas gali užsakymą kartoti (iš pateikto meniu pasirinkti tuos pačius arba kitus patiekalus bei gėrimus). Klientui apmokėjus užsakymą, padavėjas atsiskaito su kasa bei kasininkas patvirtina užsakymo apmokėjimą.



10 pav. Veiklos procesų modelis

3.6. Workflow modelis

Darbų sekos modelis (workflow model) (11 pav.) yra kuriamas per BIM modelį (4 pav.), nes į šį modelį yra įtraukiamos pagrindinės organizacinės struktūros, kurios atlieka tam tikrą darbų seką. Naudojant darbų sekos modeliavimą leidžiama sukurti labiau detalesnį veiklų modelį, kuris apima vartotojo biznio procesą.

Šis modelis atvaizduoja biznio procesus išreiškiant juos veiklos komponentais ir darbų seka tarp tų veiklų. Darbų sekos modelis koncentruotas į darbų seką nuo biznio pradžios iki galo. Tai atspindi pagrindinis aukščiausio lygio modelis. Jei koks nors procesas

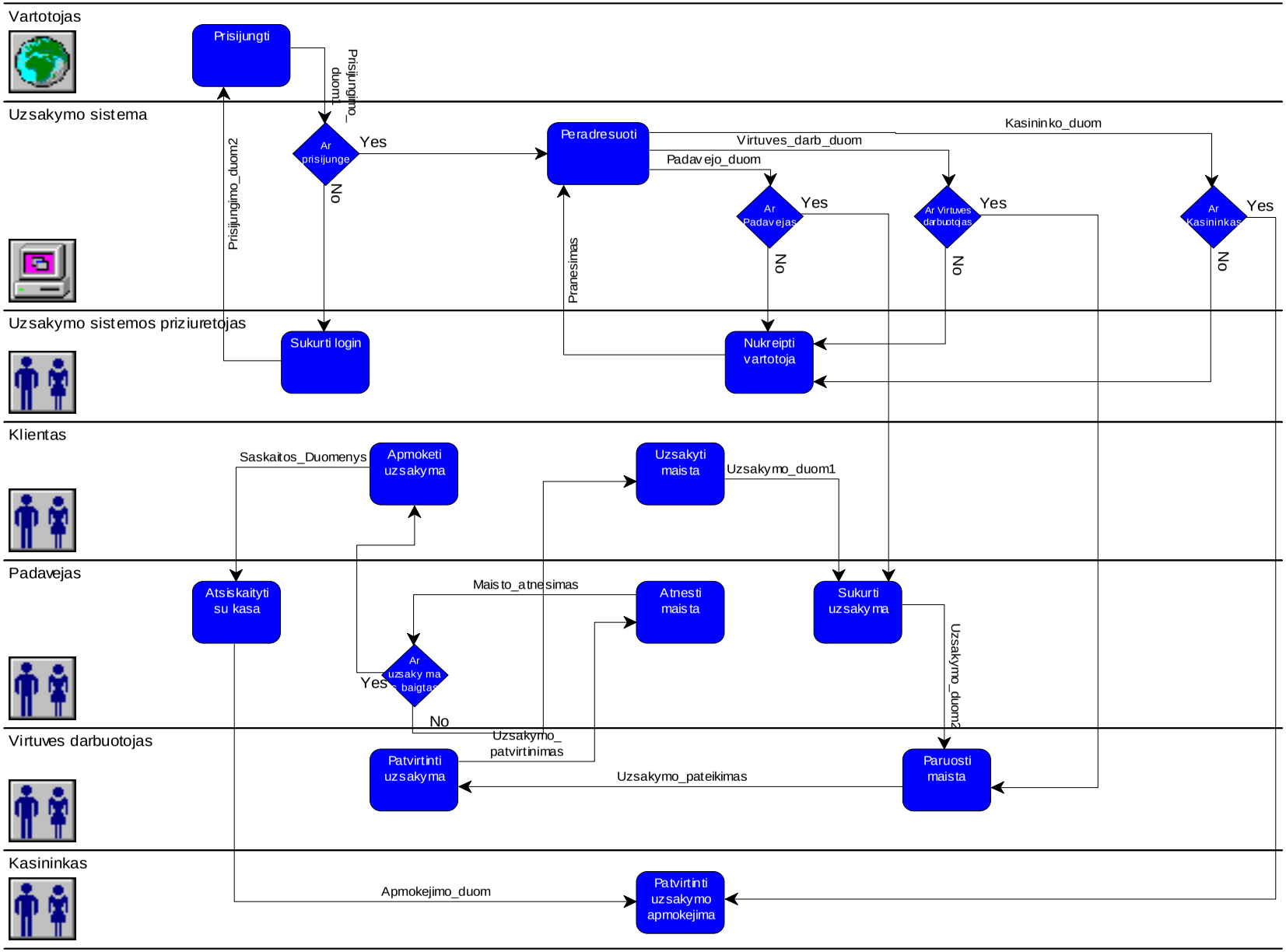
ar darbas yra sudėtinis kitų darbų atžvilgiu, tai šiam procesui yra sukuriamas detalesnis žemesnio lygio darbų sekos modelis.

Vartotojui įvedus prisijungimo vardą bei slaptažodį prisijungimo duomenys siunčiami į užsakymo sistemą. Tikrinama ar vartotojas prisijungė. Keliami du sąlygos variantai:

- Jeigu neprisijungė, užsakymo sistemos prižiūrėtojas sukuria prisijungimo vardą bei slaptažodį, tam tikros kategorijos vartotojui (padavėjui, virtuvės darbuotojui arba kasininkui). Sukurto vartotojo prisijungimo duomenys pateikiami vartotojui.
- Jeigu prisijungė, vartotojas peradresuojamas atitinkamai pagal kategoriją.

Jeigu prisijungia padavėjas, jis sukuria iš kliento priimtą užsakymą. Užsakymo duomenys siunčiami virtuvės darbuotojui, kuris paruošia maistą bei patvirtina, jog užsakymas yra paruoštas. Padavėjas gauna užsakymo patvirtinimą iš virtuvės darbuotojo bei pateikia užsakymą (atneša maistą, gėrimus) klientui. Keliama sąlyga ar užsakymas baigtas:

- Jeigu ne, klientas papildo užsakymą. Padavėjas pagal kliento užsakymo duomenis, sukuria užsakymą (kartojamas užsakymo procesas).
- Jeigu taip, klientas apmoka užsakymą. Sąskaitos duomenys pateikiami padavėjui, padavėjas atsiskaito su kasa, o kasininkas patvirtina užsakymo apmokėjimą.



11 pav. Darbų sekos modelis

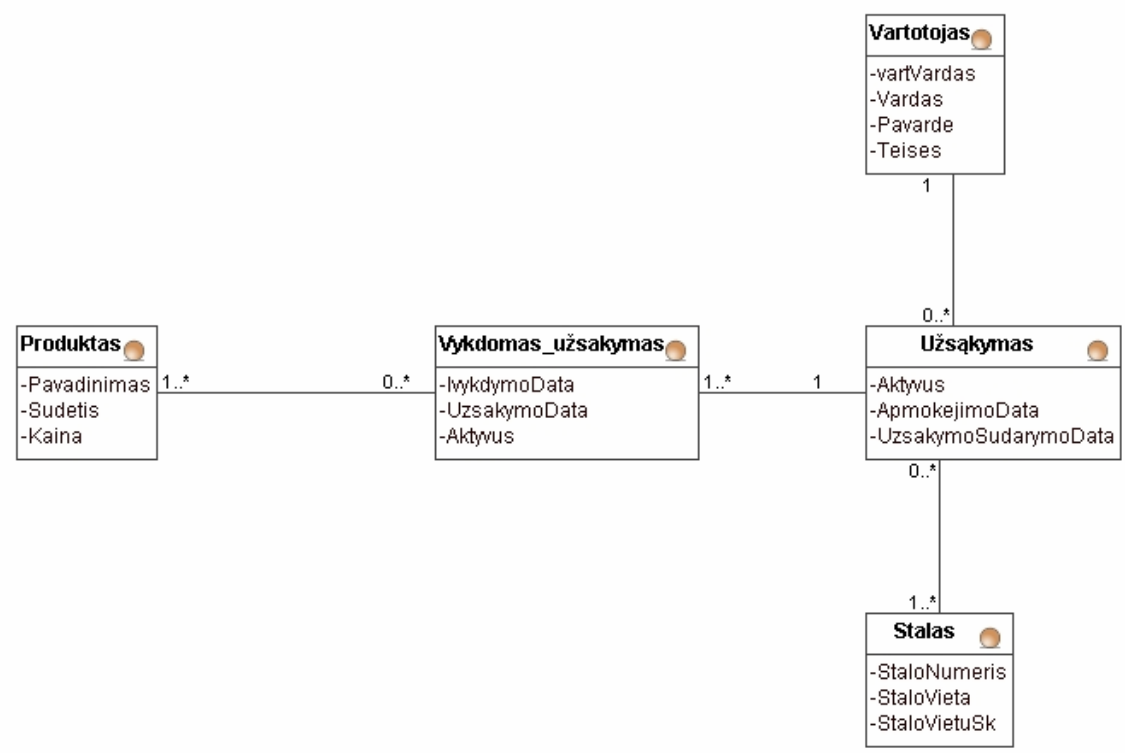
3.7. Esybių ryšių diagrama

Tipiniai ER modelio elementai yra šie:

Esybė (angl. *entity*) - tai realus ar įsivaizduojamas prasminis objektas, apie kurį kuriamoji informacinė sistema turės saugoti informaciją.

Atributas (angl. *attribute*) - tai esybės savybė ar detalė, kuri kokybiškai ar kiekybiškai gali identifikuoti, klasifikuoti ar išreikšti esybės būseną. Kiekvieną esybę turi aprašyti bent vienas atributas, o kiekvienas atributas aprašo tam tikrą vieną esybę.

Ryšys (angl. *relationship*) - tai binarinė įvardyta reikšminga asociacija tarp esybių.



12 pav. Dalykinės srities esybių ryšių modelis

12 pav. pavaizduotame modelyje esybės yra:

- Produktas
- Vykdomas_užsakymas
- Vartotojas
- Užsakymas
- Stalas

Pvz. esybės *Produktas* atributai yra:

- Pavadinimas

- Sudėtis
- Kaina

Ryšius su nurodytais kardinalumais reikėtų suprasti taip:

a. Vienas *vykdomas_ užsakymas* gali turėti vieną ir daugiau *produktų* (žymima – (1..*));

b. Tas pats *produktas* gali būti priskirtas keliems *vykdomiems_užsakymams* arba neturėti *vykdomo_užsakymo* (žymima – (0..*));

c. Vienas *vartotojas* gali sukurti keletą *užsakymų* arba *užsakymo* gali ir nesukurti (žymima – (0..*));

d. *Užsakymas* negali egzistuoti be *stalo*, t.y. turi būti bent vienas *stalas*, kad būtų įvykdytas užsakymas (žymima – (1..*));

e. *Stalas* gali egzistuoti be *užsakymo* (pvz. tuo metu prie stalo nėra kliento, kuris pateiktų užsakymą padavėjui (žymima – (0..*)).

3.8. Komponentinio modelio sudarymas

Komponentų sąveikos modelis procesui „Pateikti užsakymą“ (13 pav.):

BD (Valdymo funkcijos)

Prisijungti prie sistemos – vartotojas prisijungia prie sistemos. Prisijungimo teises prie užsakymo sistemos turi padavėjas, virtuvės darbuotojas bei kasininkas.

Sudaryti užsakymą – užsakymą iš kliento priima padavėjas ir užsakymo sistemoje sukuria užsakymą. Prisijungęs virtuvės darbuotojas padavėjo sudarytą užsakymą priima (kitaip tariant irgi sukuria užsakymą, bet jau realų – paruošia maistą, gėrimus).

Patvirtinti įvykdytą užsakymą – paruošęs maistą ir gėrimus, virtuvės darbuotojas patvirtina, jog užsakymas įvykdytas ir padavėjas gali patiekti užsakymą klientui.

Patvirtinti užsakymo apmokėjimą – klientui apmokėjus sąskaitą padavėjui, pastarasis sąskaitą pateikia kasininkui, kuris patvirtina, jog užsakymas apmokėtas.

IPD (Skaičiavimai)

Tikrinti teises – pagal prisijungimo vardą tikrinama, koks vartotojas (padavėjas, virtuvės darbuotojas ar kasininkas) prisijungė.

Peradresuoti - pagal prisijungimo vardą patikrinus vartotojo teises, vartotojas nukreipiamas į tik jam skirtą sistemos dalį.

Valdyti užsakymą – tvarko, valdo visą užsakymo procesą (nuo kliento užsakymo iki sąskaitos apmokėjimo bei atsiskaitymo su kasininku).

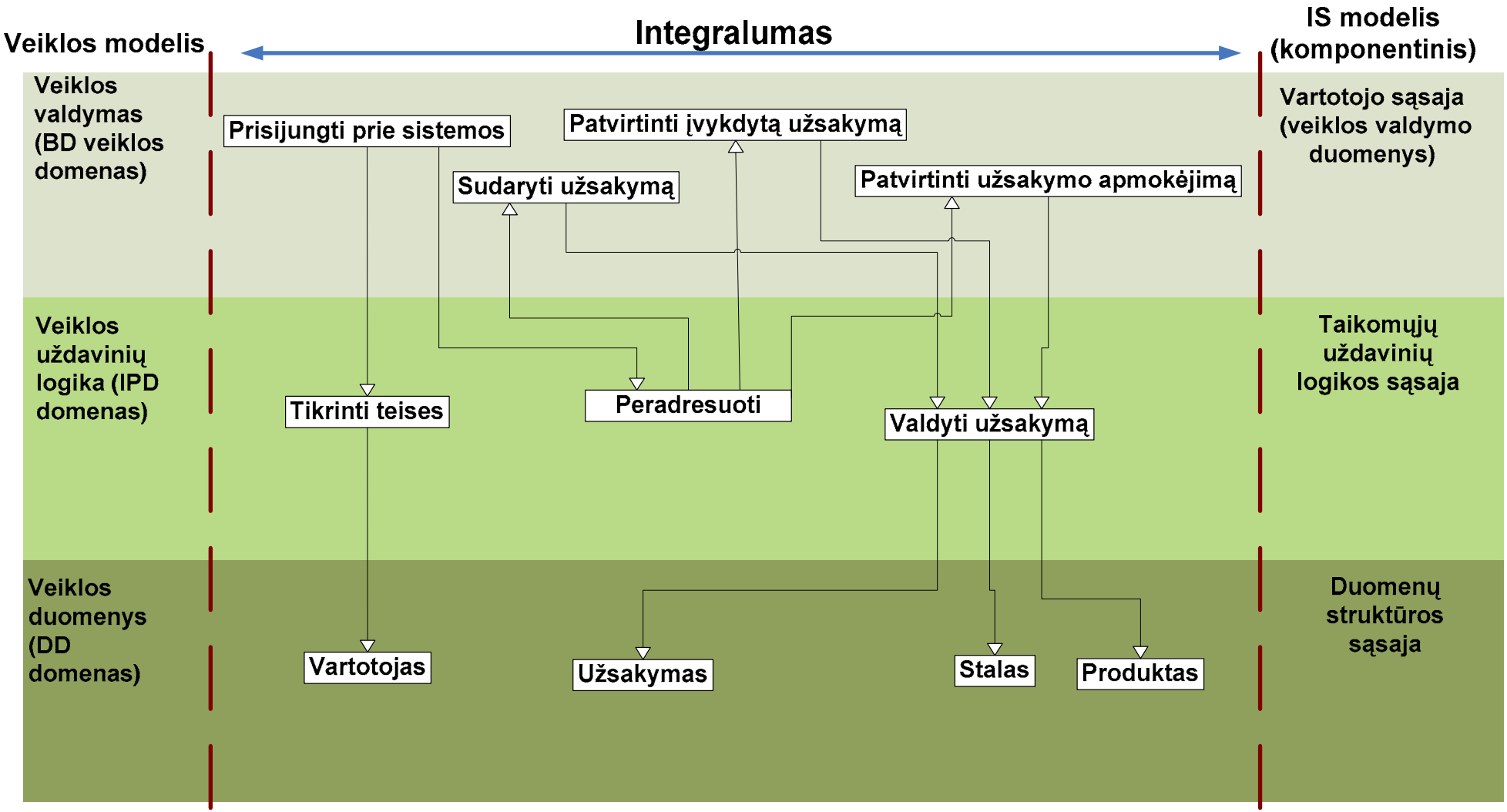
DD (Duomenų struktūros)

Vartotojas - klasė *vartotojas* naudoja esybės „vartotojas“ duomenis.

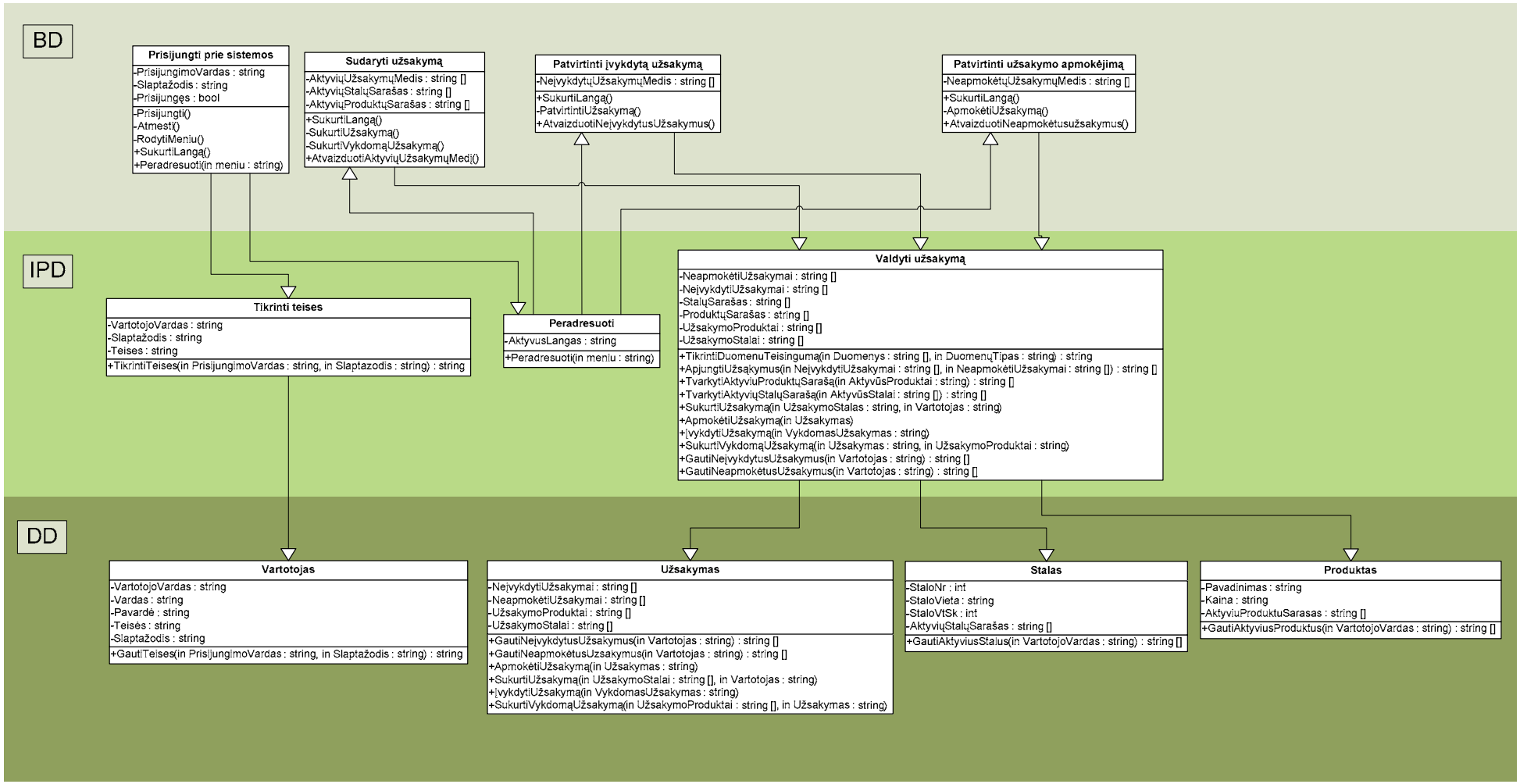
Užsakymas - klasė *užsakymas* naudoja esybės „užsakymas“ duomenis.

Stalas - klasė *stalas* naudoja esybės „stalas“ duomenis.

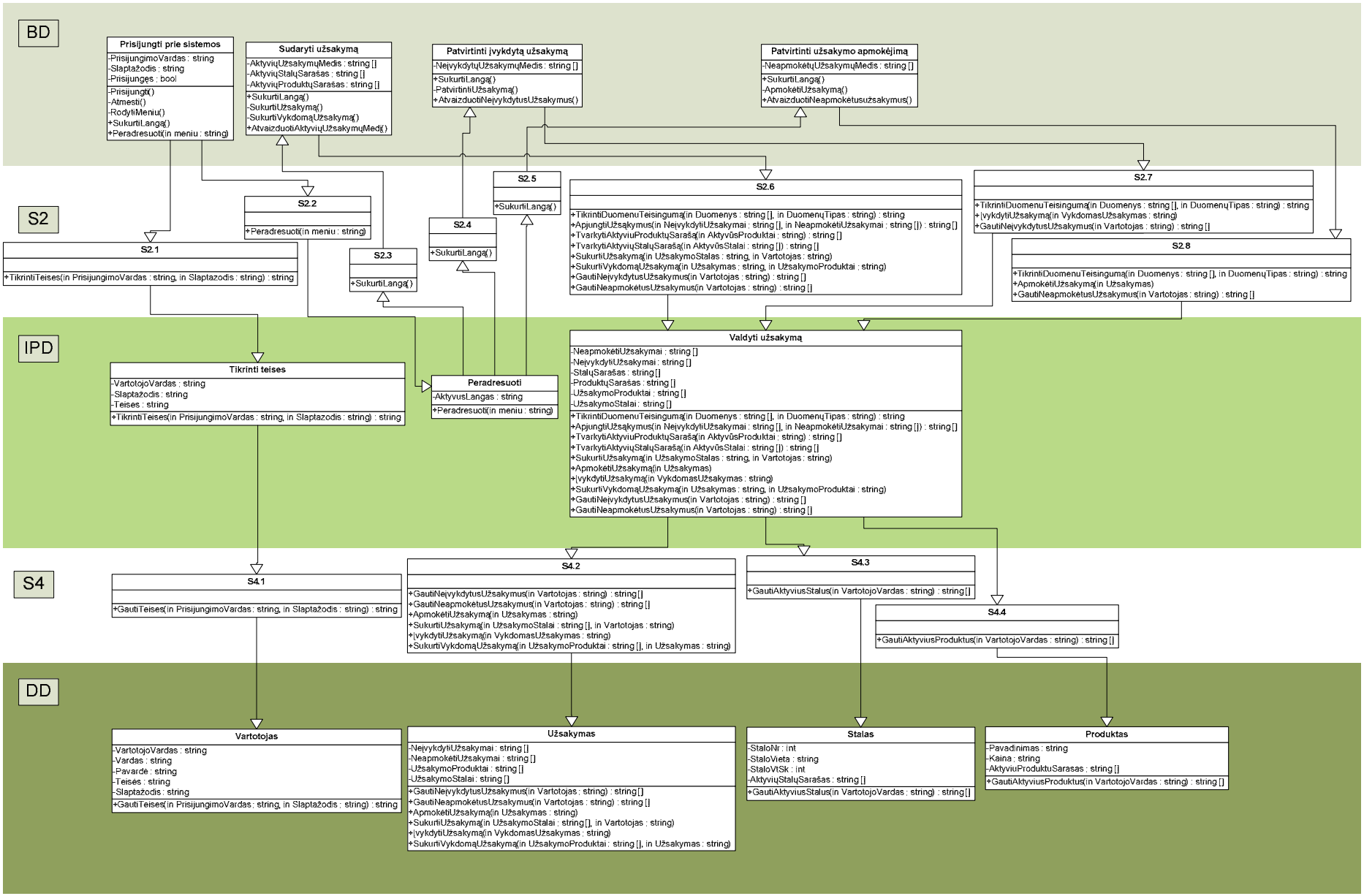
Produktas - klasė *produktas* naudoja esybės „produktas“ duomenis.



13 pav. Supaprastintas komponentų sąveikos modelis procesui „Pateikti užsakymą“



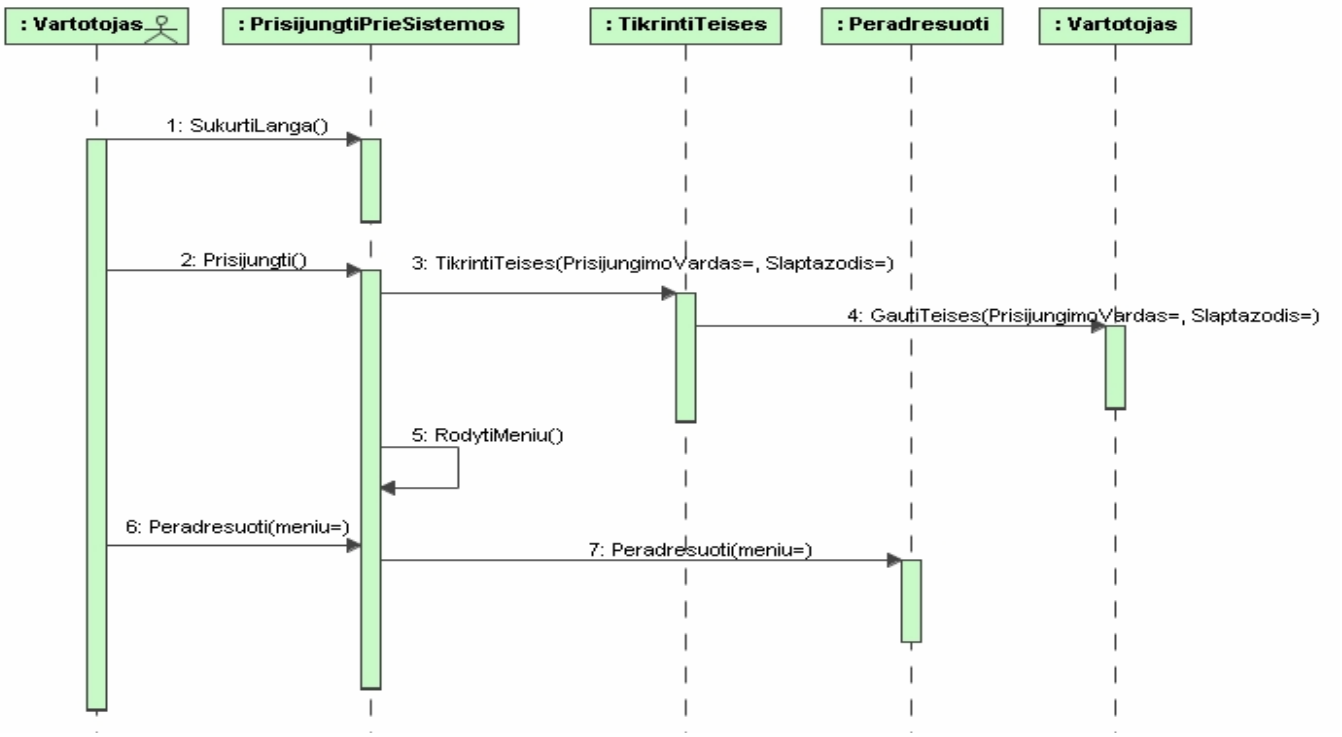
14 pav. Komponentų sąveikos modelis procesui „Pateikti užsakymą“ su atributais ir metodais



15 pav. Komponentų sąveikos modelis procesui „Pateikti užsakymą“ su atributais, metodais bei sąsajomis

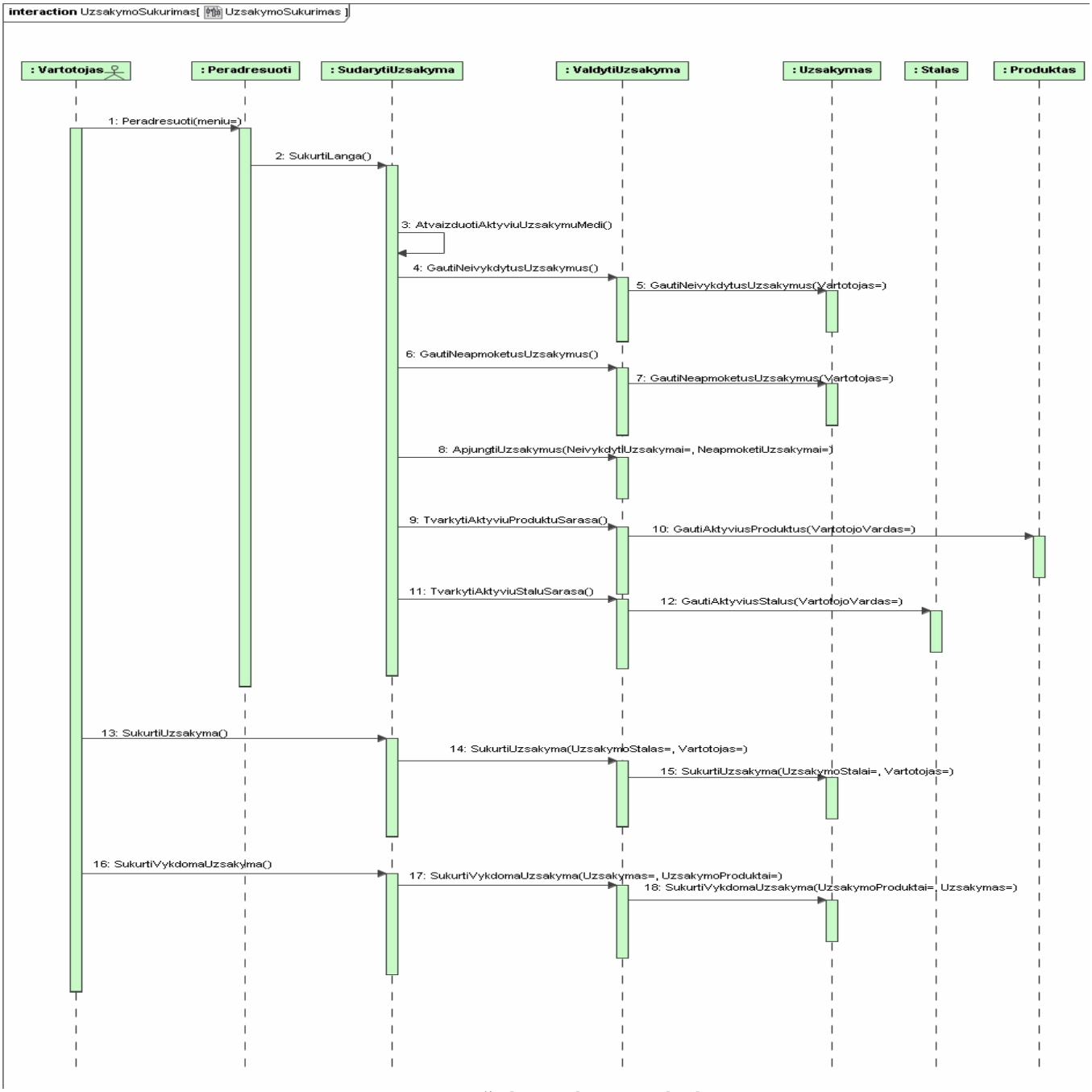
3.9. Sekų diagramų sudarymas

Sekų diagramos sudaromos atsižvelgiant į veiklos ypatumus bei komponentinį modelį, jos specifikuoja sąveikas tarp komponentų. 16 pav. suprojektuota sekų diagramos dalis, vaizduojanti vartotojo prisijungimo, jo teisių tikrinimo bei peradresavimo procesus. Ši diagrama padeda specifikuoti komponentų tarpusavio sąveiką. Kaip matome iš diagramos, vartotojas paleisdamas sistemą sukuria pagrindinį langą („Prisijungti prie sistemos“ komponentas), vartotojui jungiantis prie sistemos, tikrinamos teises komponente „TikrintiTeises“, kurios yra gaunamos iš komponento „Vartotojas“, pasinaudojus pastarojo komponento operacija „GautiTeises“. Kai vartotojas jau prisijungęs, pagal jo teises yra suformuojamas meniu. Vartotojui, pasirinkus vieną iš meniu komponentų, atitinkama operacija iškviečiama komponente „Peradresuoti“.



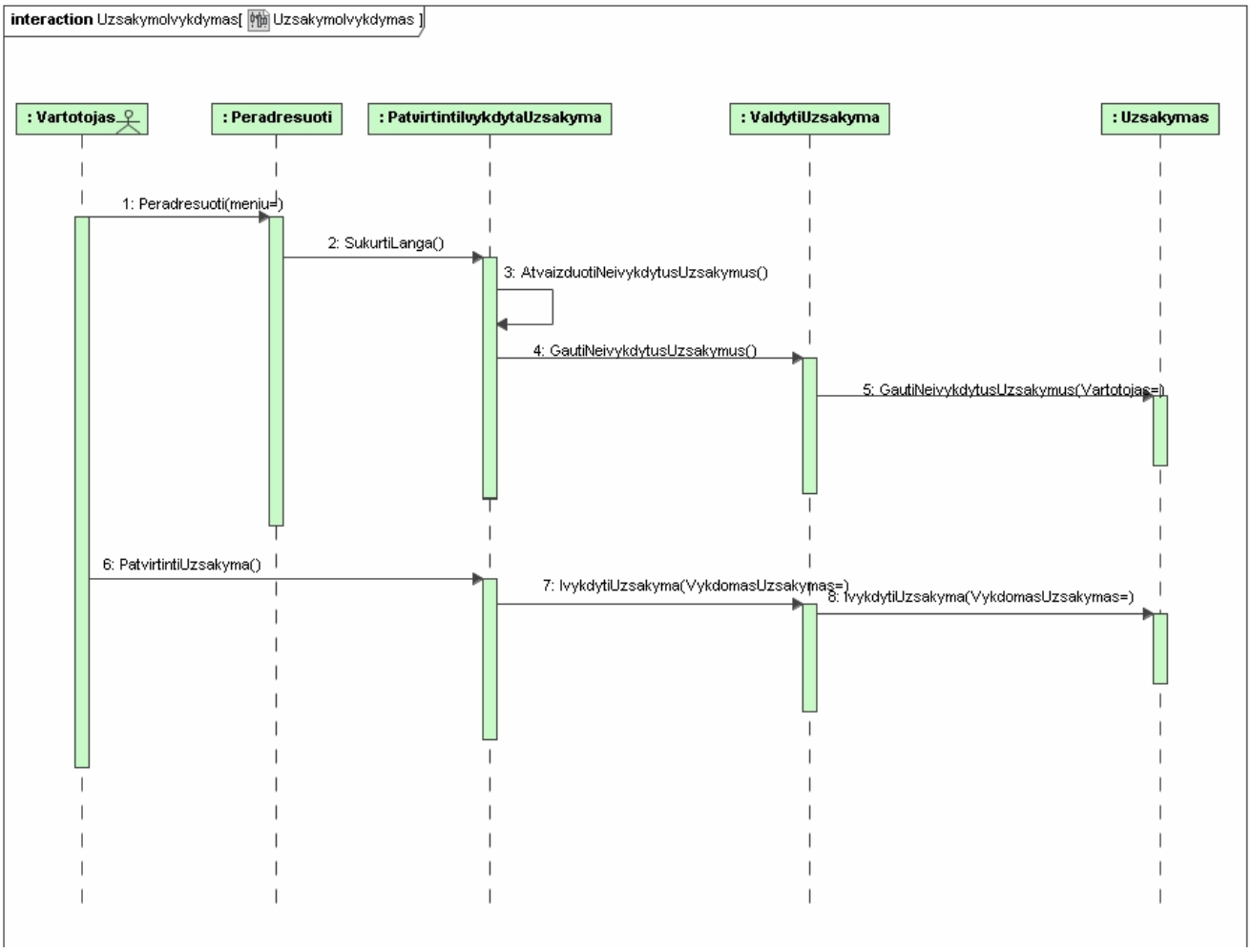
16 pav. Sekų diagrama

17 pav. pavaizduota užsakymo sukūrimo sekų diagrama. Joje vaizduojami užsakymo sudarymo formos duomenų užkrovimas, užsakymo, bei vykdomo užsakymo sukūrimas. Iš diagramos matome, kad naudojamos komponentų „Užsakymas“, „Stalas“ ir „Produktas“ operacijos. Komponentas „ValdytiUžsakyma“ skirtas vykdyti duomenų apdorojimą (pvz. užsakymo duomenų tikrinimui, apjungimui ar užsakymo kainos skaičiavimui).



17 pav. „Užsakymo sukūrimo“ sekų diagrama

„Užsakymo vykdymo“ sekų diagrama pateikiama 18 paveiksle. Čia, kai sukuriamas užsakymo patvirtinimo langas, naudojantis komponentu „Užsakymas“ yra gaunami vartotojo neįvykdyti užsakymai, jie yra apdorojami komponente „ValdytiUzsakyma“, taip pat šioje sekų diagramoje pavaizduotas užsakymo įvykdymo patvirtinimas.



18 pav. „Užsakymo vykdymo“ sekų diagrama

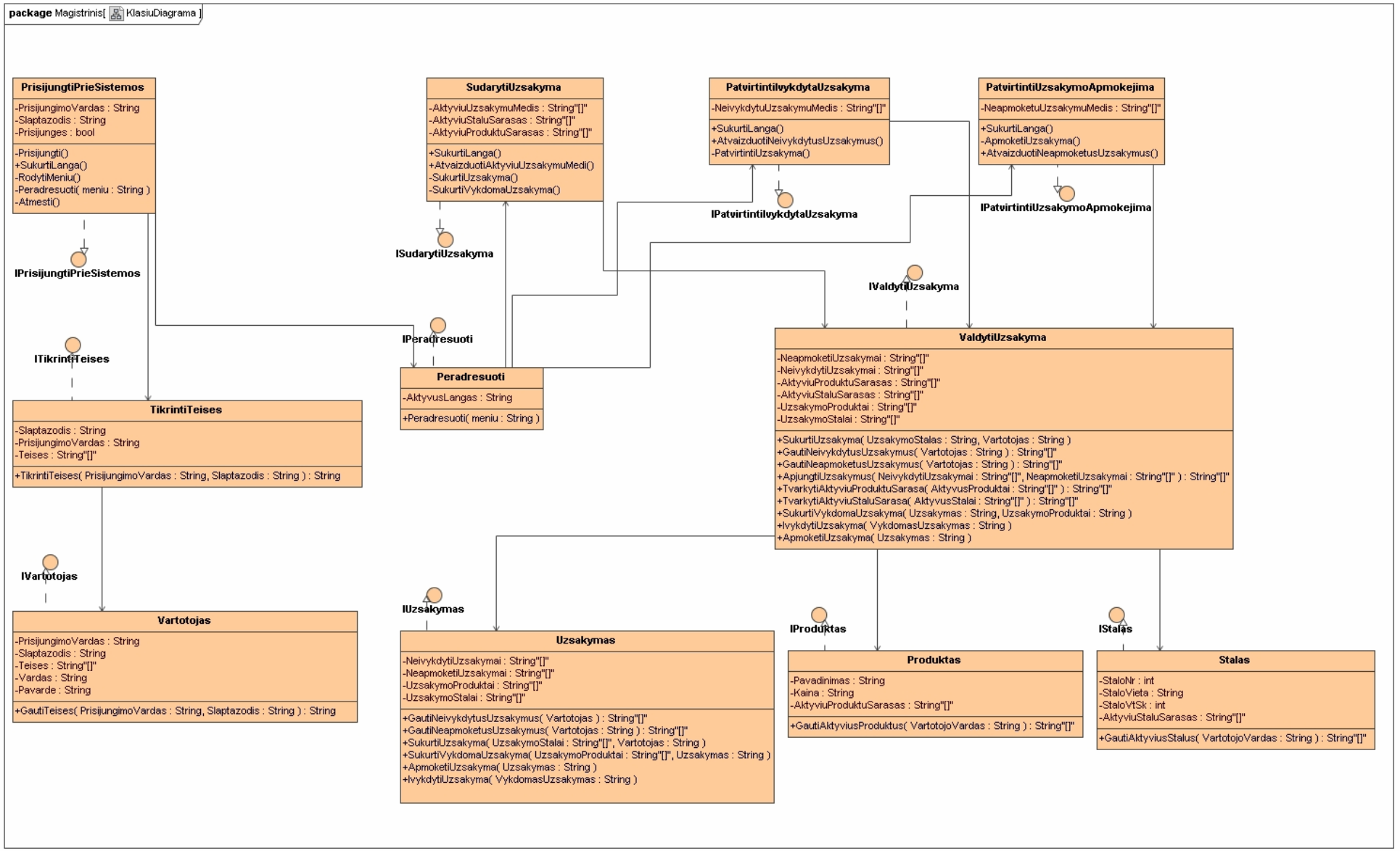
„Užsakymo apmokėjimo patvirtinimo“ sekų diagrama pateikiama 19 paveiksle. Čia, kai sukuriamas užsakymo apmokėjimo patvirtinimo langas, naudojantis komponentu „Užsakymas“ yra gaunami vartotojo neįvykdyti užsakymai, jie yra apdorojami komponente „ValdytiUzsakyma“, taip pat šioje sekų diagramoje pavaizduotas užsakymo įvykdymo patvirtinimas.

19 pav. „Užsakymo apmokėjimo“ sekų diagrama

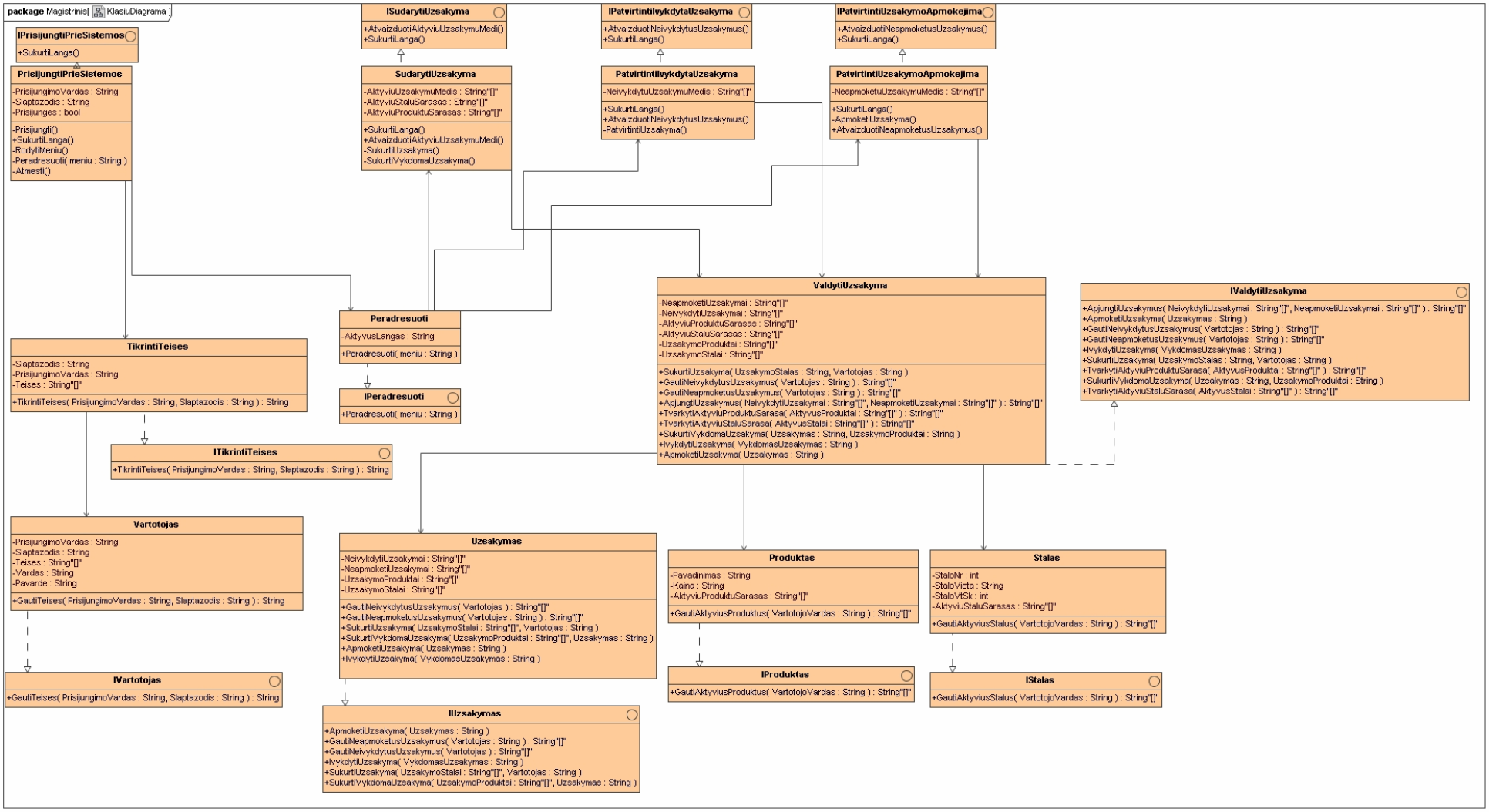
3.10. Klasių diagramos sudarymas

Sistemos klasių diagramoje yra detaliai aprašomi komponentai, sekantis žingsni yra kodo generavimas, kuris yra generuojamas iš klasių bei sekų diagramų. Tai atliekama tada, kai jau sudarytas komponentinis modelis, komponentų aprašai yra perkeliami į klasių diagramą, sukuriami „interfeisai“ kiekvienai klasei (20 paveikslas). Po to sukuriami interfeisų aprašai. Interfeisuose yra įrašomi visi klasių vieši metodai bei atributai. Tai yra komponentinio modeliavimo specifika, skirta tam, kad komponentą galimą butų lengvai integruoti į kitą sistemos dalį ar net į visiškai kitą sistemą, jį pildyti ar keisti, bet kad tai

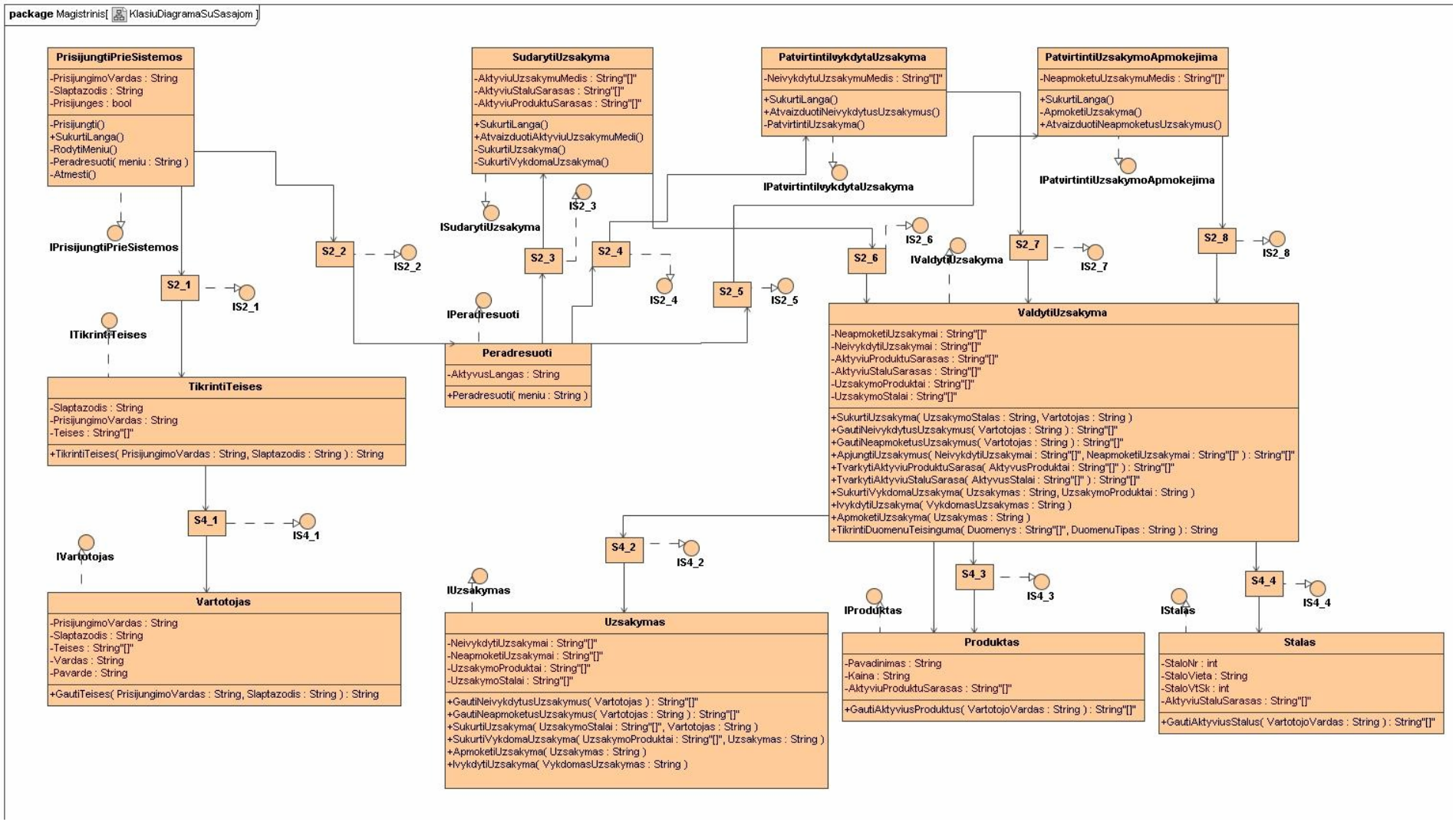
darant sistemos palaikomos funkcijos nesikeistų. 21 paveiksle pateikiama sistemos klasių diagrama su klasių interfeisų aprašais. Sekantis etapas yra sąsajų tarp komponentų sudarymas. Sąsajos tai tokie komponentai, kurie skirti duomenų perdavimui tarp komponentų bei šių duomenų apjungimui ar transformavimui. Sąsajos taip pat gali būti laikomos pilnaverčiais komponentais, todėl joms taip pat turi realizuoti interfeisus. Klasių diagrama su išskirtomis sąsajom yra pateikiama 22 paveiksle. Sekantis etapas yra šių komponentų aprašų sudarymas. Klasių diagramos su sudarytais aprašais pateikiamos 24 paveiksle. 23 paveiksle pateikiama šios klasių diagramos dalis, realizuojanti vartotojo prisijungimą, jo teisių tikrinimą, bei peradresavimo procesus. Duomenų bazės schema, sudaryta iš transformuotos esybių klasės diagramos (25 paveikslas) pateikiama 26 paveiksle.



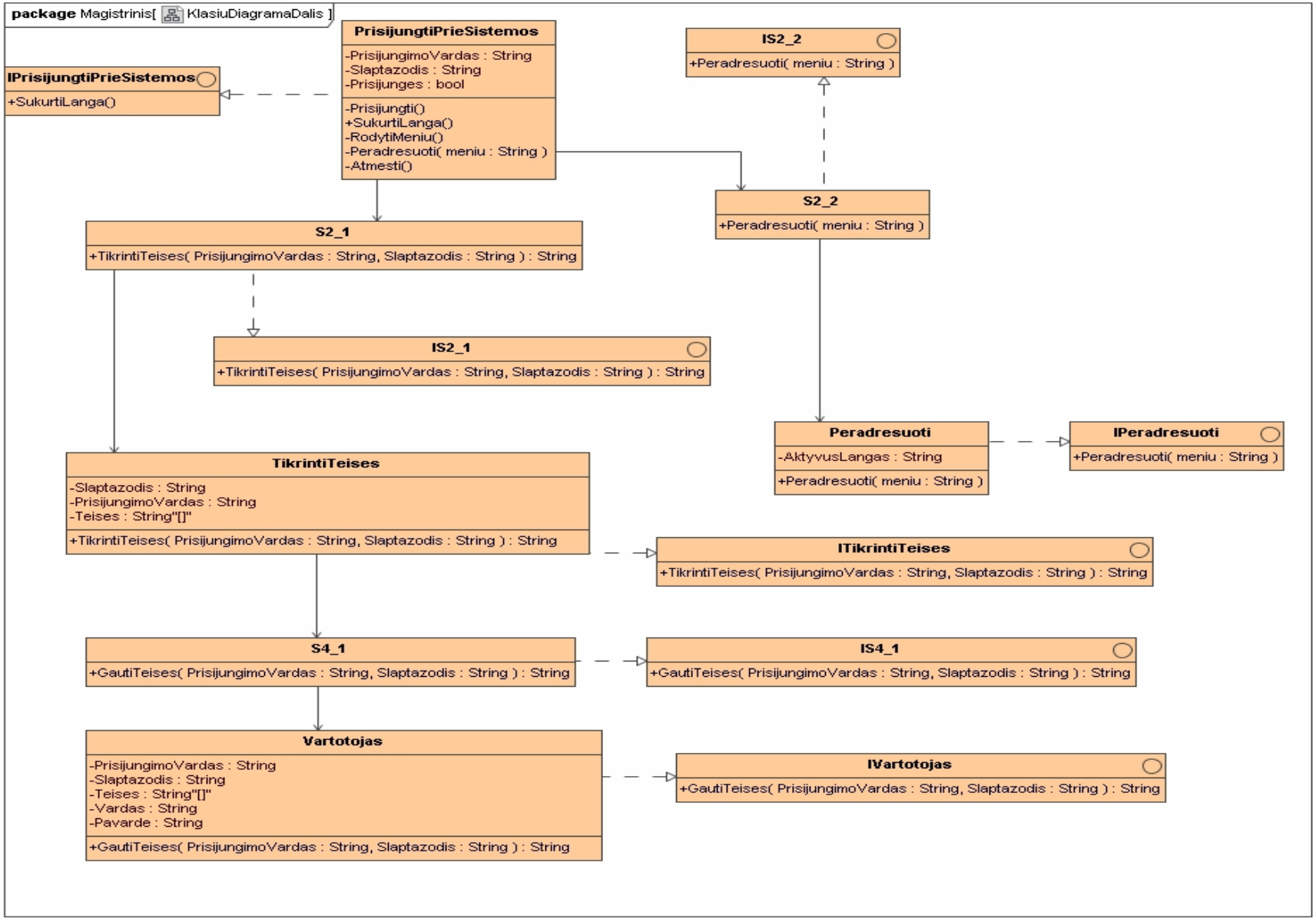
20 pav. Klasių diagrama su interfeisais



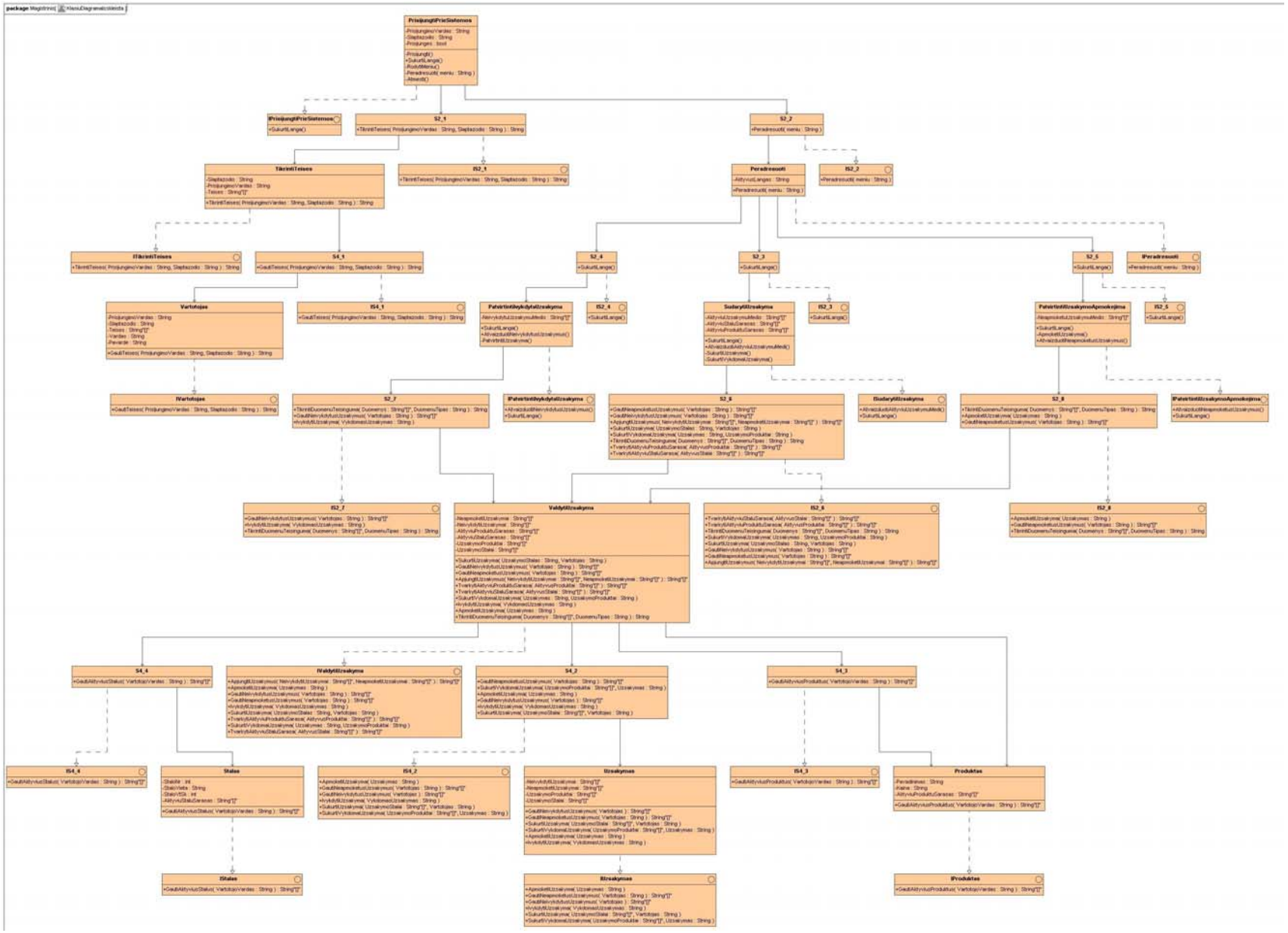
21 pav. Klasių diagrama su interfeisų aprašais



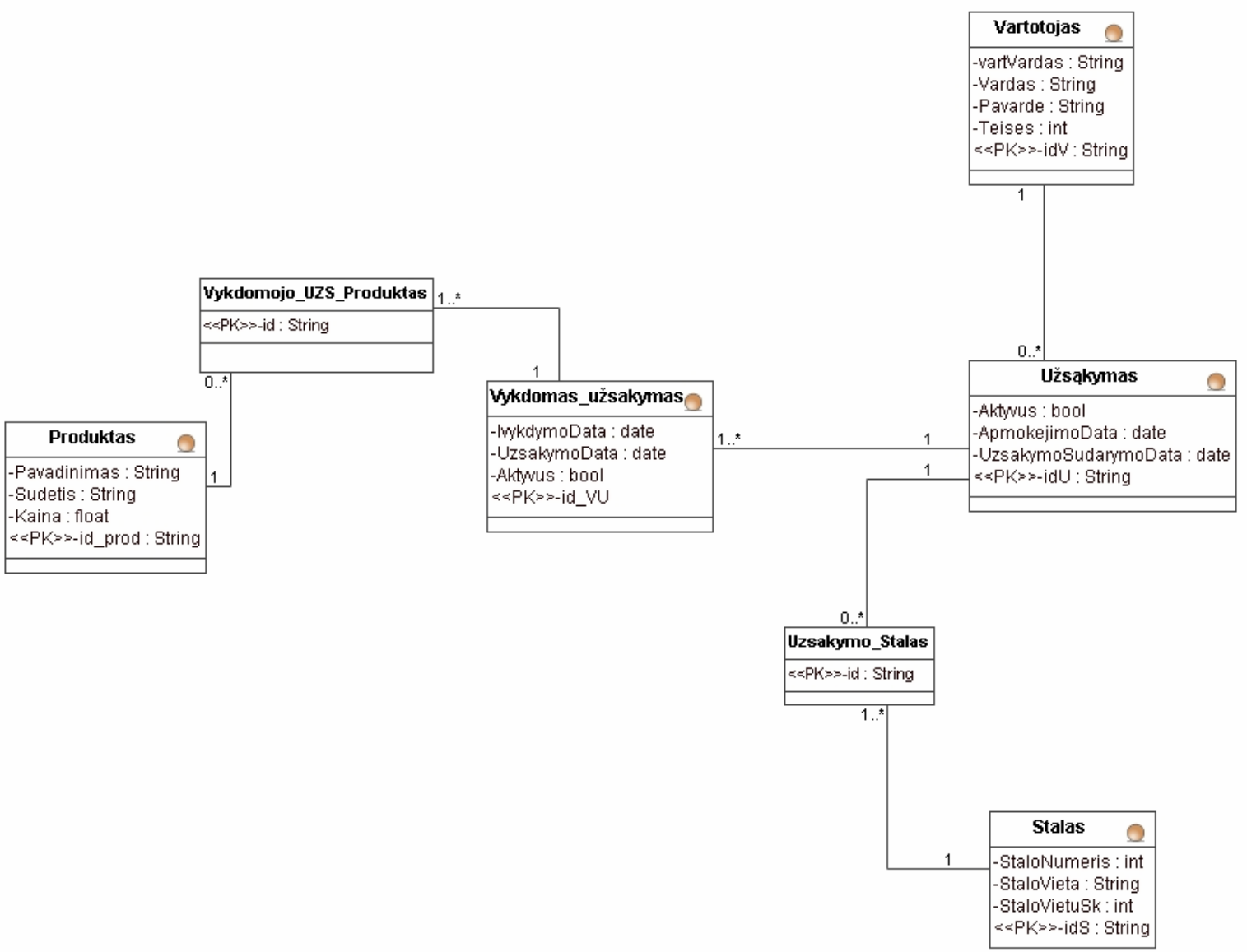
22 pav. Klasių diagrama su suformuotomis sąsajų klasėmis



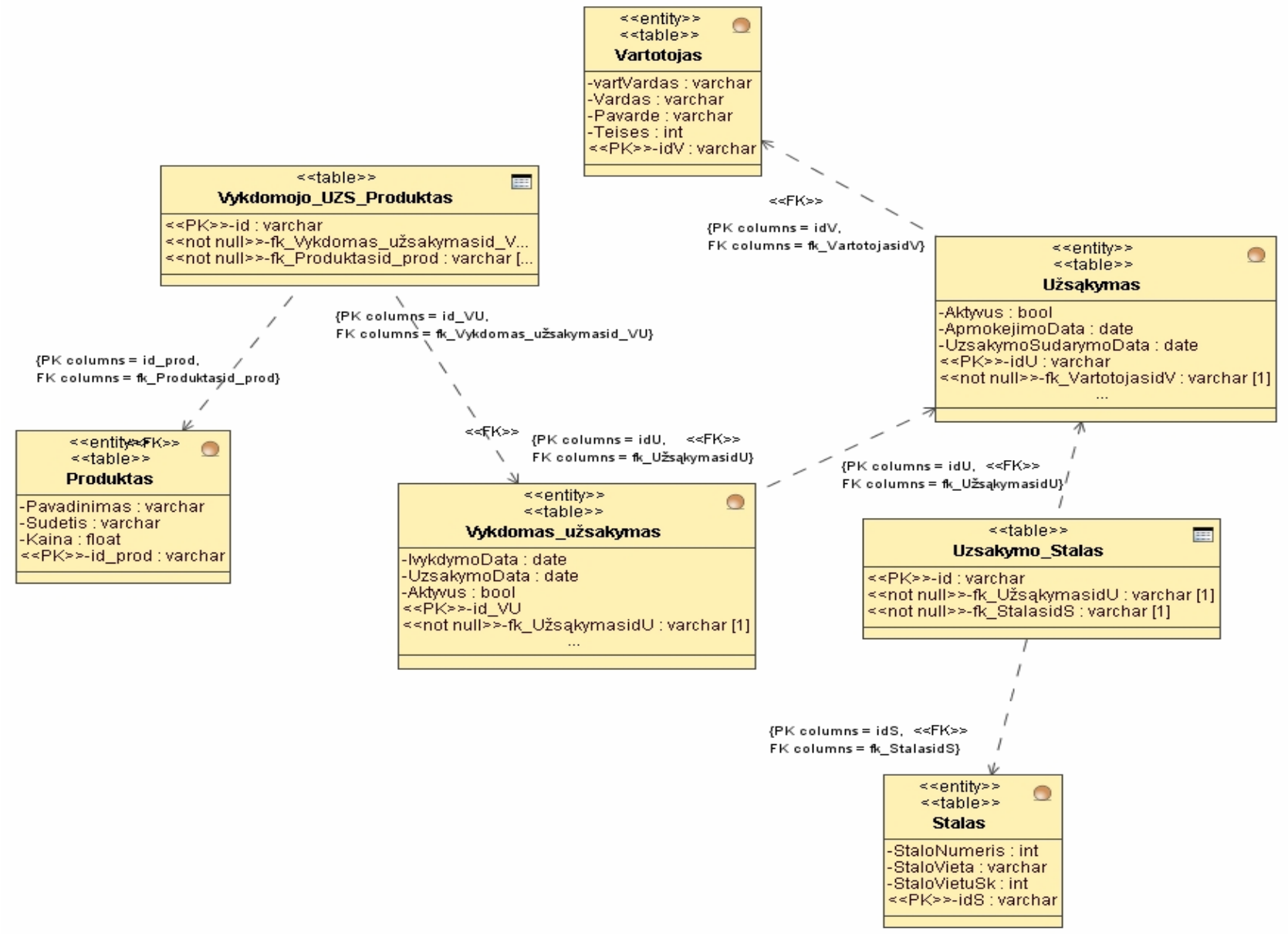
23 pav. Detalios klasių diagramos fragmentas, atitinkantis prisijungimo procesą



24 pav. Detalios klasių diagramos aprašas



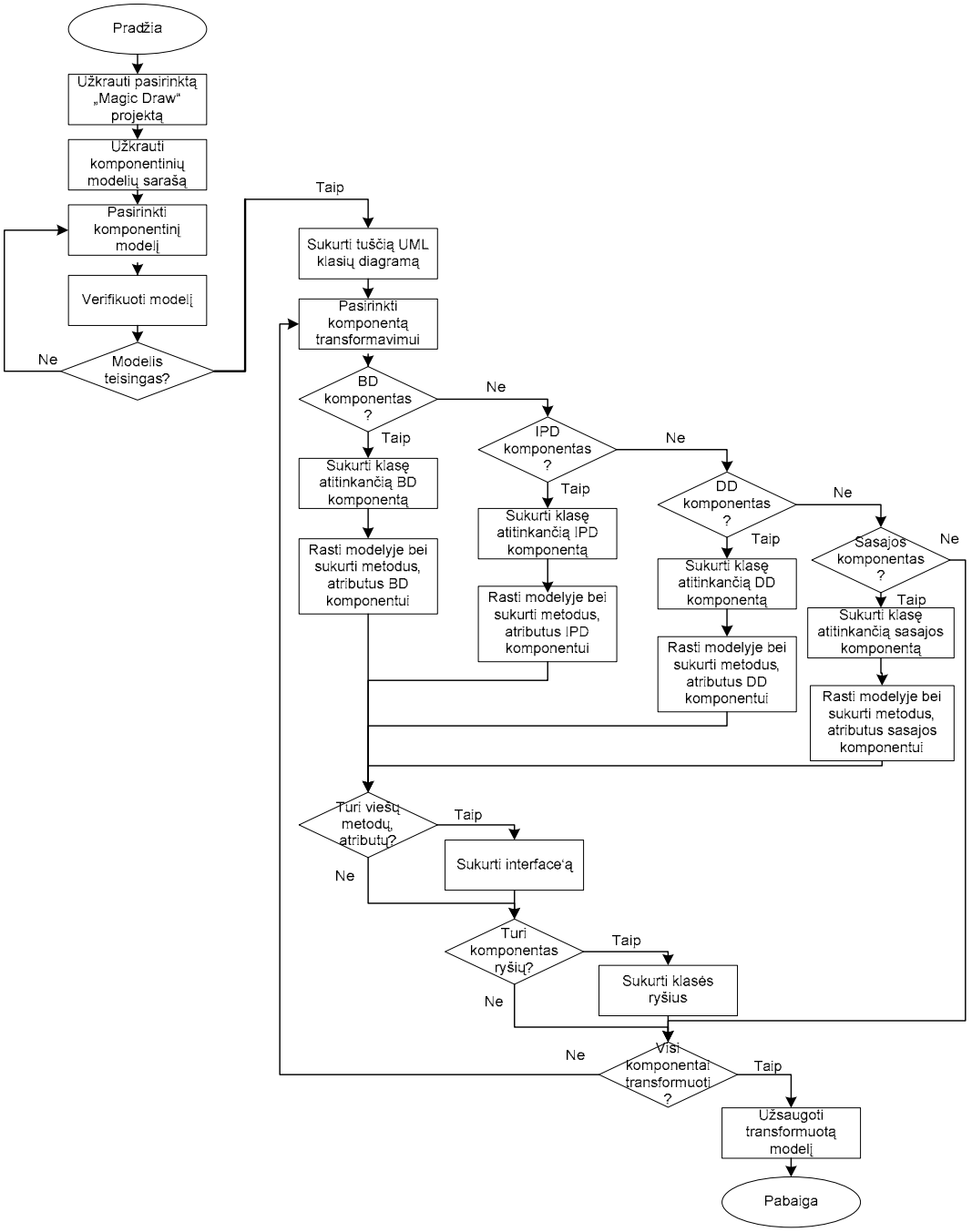
25 pav. Esybių-klasių diagrama



26 pav. Sistemos duomenų bazės schema

3.11. Komponentinio IS modelio transformavimo į UML klasių modelį algoritmas

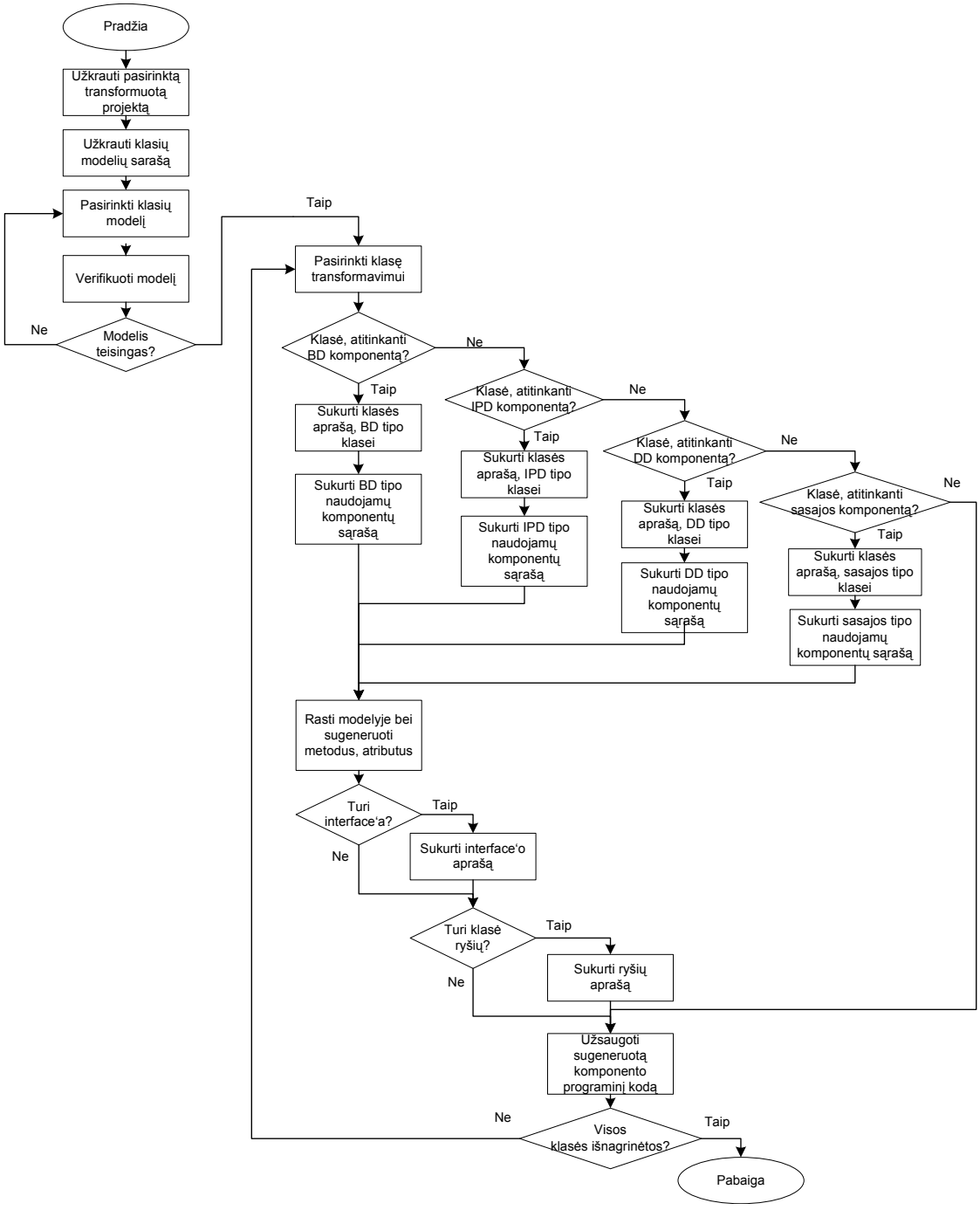
Sudarytas komponentinio IS modelio transformavimo į UML klasių modelį algoritmas, pagal kurį planuojama realizuoti programinę įrangą, sugebančią atlikti komponentinio IS modelio transformavimą į UML klasių modelį. Algoritmo blokinė schema pateikiama 27 paveikslėlyje.



27 pav. Kompnentinio IS modelio transformavimo į modifikuotą UML klasių diagramą algoritmo blokinė schema

3.12. Programinio kodo generavimo iš UML klasių diagramos algoritmas

Sudarytas programinio kodo generavimo iš modifikuotos UML klasių diagramos algoritmas, pagal kurį planuojama realizuoti programinę įrangą, sugebančią atlikti programinio kodo generavimą į UML klasių modelį. Algoritmo blokinė schema pateikiama 28 paveikslėlyje.



28 pav. Programinio kodo generavimo iš modifikuotos UML klasių diagramos algoritmo blokinė schema

3.13. Projekto išvados

Projektuojant integruotas kompiuterizuotas informacijos sistemas tikslinga apjungti IS kūrimą informacinės architektūros modelio pagrindu bei komponentinį IS projektavimą, siekianti surinkti IS iš kompiuterizuotų veiklos komponentų.

Pasiūlytas metodas susieja IS architektūros modelį su duomenų srautų diagrama ir atvaizduoja juose esančią informaciją į naujo tipo modelį – komponentinį sistemos modelį, kuriame išskiriami tokio tipo komponentai:

• valdymo funkcijos (BD);

• skaičiavimai arba funkciniai komponentai (IPD);

• duomenų struktūros (DD).

Kaip pavyzdys pasirinktas restoranas. Siekiant sukurti komponentų sąveikos modelį, parinktas procesas „Pateikti užsakymą“.

Sukurti komponentai, bei sąsajų komponentės, kurios sieja informacinių procesų domeno (IPD) komponentą su duomenų domeno(DD) komponentu bei informacinių procesų domeno (IPD) komponentą su biznio domeno (BD) komponentu.

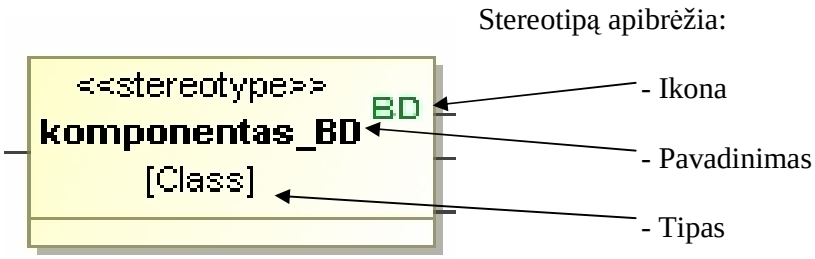
Sukurtos sekų bei klasių diagramos, iš kurių bus generuojamas programos kodas.

4. KOMPONENTINIO IS MODELIO TRANSFORMAVIMO SISTEMOS REALIZACIJA

4.1. Komponentinio modelio profailas

Profailas UML kalboje yra speciali paketų rūšis, naudojama surinkti stereotipų rinkinį, konkrečias duomenų sritis ir bibliotekas. Profailas gali būti sukurtas bet kokiame projekte, tačiau dažnai tas pats profailas naudojamas daugelyje projektų. Tam, kad būtų galima pakartotinai panaudoti profailą, jis turi būti sukurtas kaip nepriklausomas failas su laisvai prieinamais duomenimis vadinamais moduliu.

Stereotipas – modelio ypatybių visuma, kuria nusakomos tipinės objekto savybės. Stereotipai leidžia išplėsti UML, taigi galima kurti naujus modelio elementus, gautus iš jau egzistuojančių, tačiau turinčius konkrečias savybes.



29 pav. Stereotipo struktūra

4 lentelė. Stereotipų aprašymai

Stereotipo pavadinimas	Tipas	Lygmuo	Ikona	Aprašymas
komponentas_BD	class	BD	BD	Verslo domeno komponentas
komponentas_IPD	class	IPD	IPD	Informacinių procesų domeno komponentas
komponentas_DD	class	DD	DD	Duomenų domeno komponentas
komponentas_TPD	class	TPD	TPD	Technologinių procesų domeno komponentas
komponentas_DVD	class	DVD	DVD	Darbo vietų domeno komponentas
komponentas_S1	class	BD←→DD	S1	Verslo domeno (verslo komponentų) ir duomenų domeno (duomenų komponentų) sąsajos komponentas
komponentas_S2	class	BD←→IPD	S2	Verslo domeno (verslo komponentų) ir informacinių procesų domeno (funkcinių komponentų) sąsajos komponentas
komponentas_S3	class	BD←→TPD	S3	Verslo domeno (verslo komponentų) ir technologinių procesų domeno sąsajos komponentas

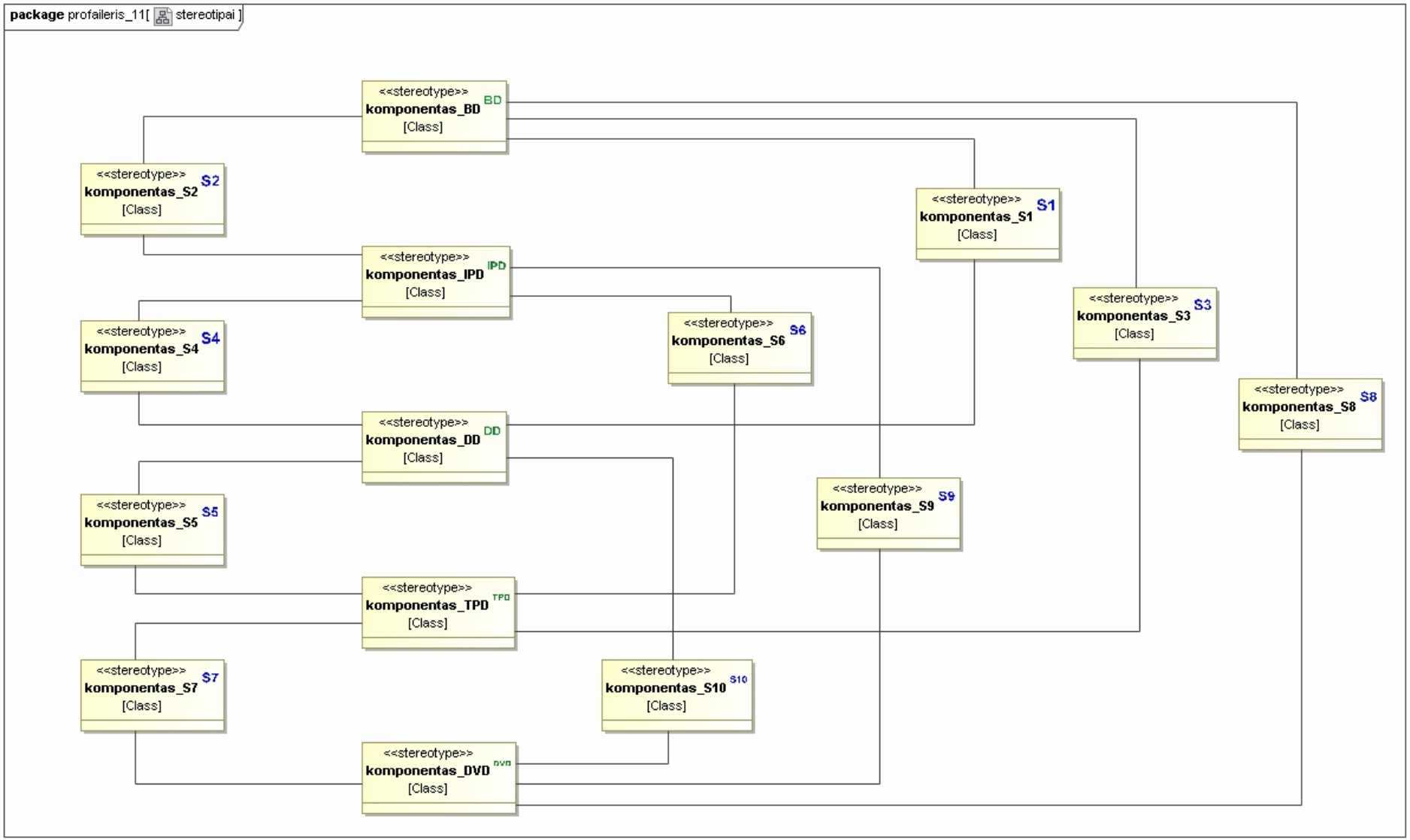
komponentas_S4	class	IPD←→DD	**S4**	Informacinių procesų domeno (funkcinių komponentų) ir duomenų domeno (duomenų komponentų) sąsajos komponentas
komponentas_S5	class	DD←→TPD	**S5**	Duomenų domeno (duomenų komponentų) ir technologinių procesų domeno sąsajos komponentas
komponentas_S6	class	IPD←→TPD	**S6**	Informacinių procesų domeno (funkcinių komponentų) ir technologinių procesų domeno sąsajos komponentas
komponentas_S7	class	TPD←→DVD	**S7**	Technologinių procesų domeno ir darbo vietų domeno sąsajos komponentas
komponentas_S8	class	BD←→DVD	**S8**	Verslo domeno (verslo komponentų) ir darbo vietų domeno sąsajos komponentas
komponentas_S9	class	IPD←→DVD	**S9**	Informacinių procesų domeno (funkcinių komponentų) ir darbo vietų domeno sąsajos komponentas
komponentas_S10	class	DD←→DVD	**S10**	Duomenų domeno (duomenų komponentų) ir darbo vietų domeno sąsajos komponentas

4.2. Magic Draw UML 12.5 paketas

Stereotipų bei profailo kūrimui pasirinktas Magic Draw UML 12.5 paketas, kadangi tai – visame pasaulyje gerai žinomas UML modeliavimo ir kodo inžinerijos įrankis, leidžiantis:

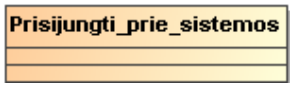
- Braižyti įvairias UML diagramas;
- Naudoti kodo inžineriją;
- Analizuoti UML modelį;
- Dirbti komandoje su didelės apimties UML modeliais;
- Integruoti Magic Draw UML 12.5 su įvairiais kitais programinės įrangos kūrimo produktais.

Sukurti stereotipai (30 pav.), kurie bus panaudoti profailo kūrimui.

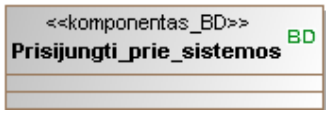


30 pav. Stereotipai, reikalingi profailo kūrimui

Komponentas, nepritaikius stereotipo:



Komponentas, pritaikius stereotipą (pakeista spalva, rodomas kokio lygmens komponentas, bei komponento paveiksliukas):

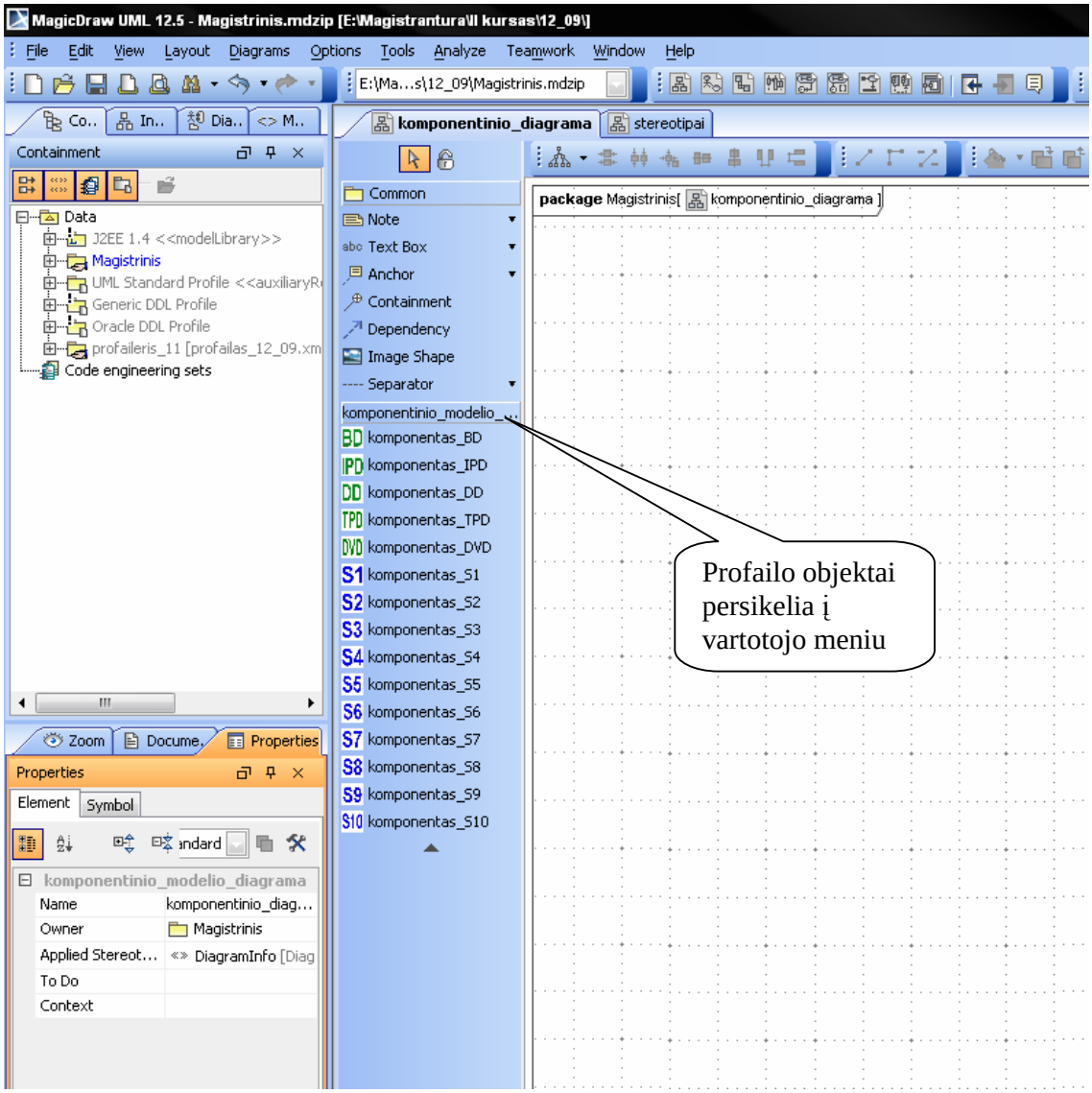


5 lentelė. Komponentų aprašymai

Stereotipo pavadinimas	Komponento pavadinimas	Ryšiai	Lygmuo
komponentas_BD	PrisijungtiPrieSistemos	S2_1 S2_2	BD
komponentas_BD	SudarytiUzsakyma	S2_3 S2_6	BD
komponentas_BD	PatvirtintiIvykdytaUzsakyma	S2_4 S2_7	BD
komponentas_BD	PatvirtintiUzsakymoApmokejima	S2_5 S2_8	BD
komponentas_IPD	TikrintiTeises	S2_1 S4_1	IPD
komponentas_IPD	Peradresuoti	S2_2 S2_3 S2_4 S2_5	IPD
komponentas_IPD	ValdytiUzsakyma	S2_6 S2_7 S2_8	IPD
komponentas_DD	Vartotojas	S4_1	DD
komponentas_DD	Uzsakymas	S4_2	DD
komponentas_DD	Stalas	S4_3	DD
komponentas_DD	Produktas	S4_4	DD
komponentas_S2	S2_1	PrisijungtiPrieSistemos TikrintiTeises	BD←→IPD
komponentas_S2	S2_2	PrisijungtiPrieSistemos Peradresuoti	BD←→IPD
komponentas_S2	S2_3	SudarytiUzsakyma Peradresuoti	BD←→IPD
komponentas_S2	S2_4	PatvirtintiIvykdytaUzsakyma Peradresuoti	BD←→IPD
komponentas_S2	S2_5	PatvirtintiUzsakymoApmokejima Peradresuoti	BD←→IPD
komponentas_S2	S2_6	SudarytiUzsakyma ValdytiUzsakyma	BD←→IPD
komponentas_S2	S2_7	PatvirtintiIvykdytaUzsakyma ValdytiUzsakyma	BD←→IPD

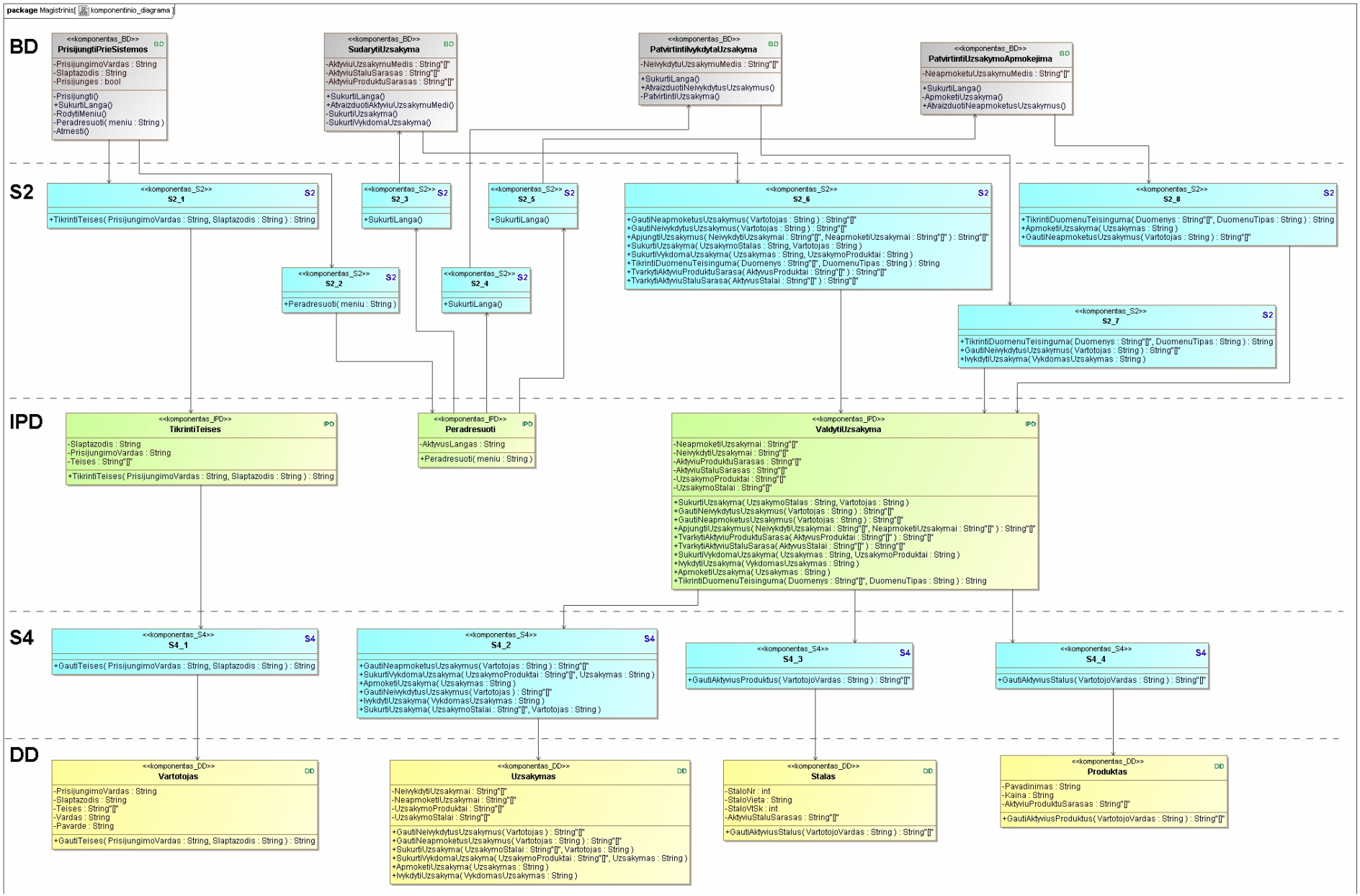
komponentas_S2	S2_8	PatvirtintiUzsakymoApmokejima ValdytiUzsakyma	BD←→IPD
komponentas_S4	S4_1	TikrintiTeises Vartotojas	IPD←→DD
komponentas_S4	S4_2	ValdytiUzsakyma Uzsakymas	IPD←→DD
komponentas_S4	S4_3	ValdytiUzsakyma Stalas	IPD←→DD
komponentas_S4	S4_4	ValdytiUzsakyma Produktas	IPD←→DD

Sukūrus stereotipus profailo objektai persikelia į vartotojo meniu (31 pav.).



31 pav. Profailo objektai Magic Draw UML 12.5 aplinkoje

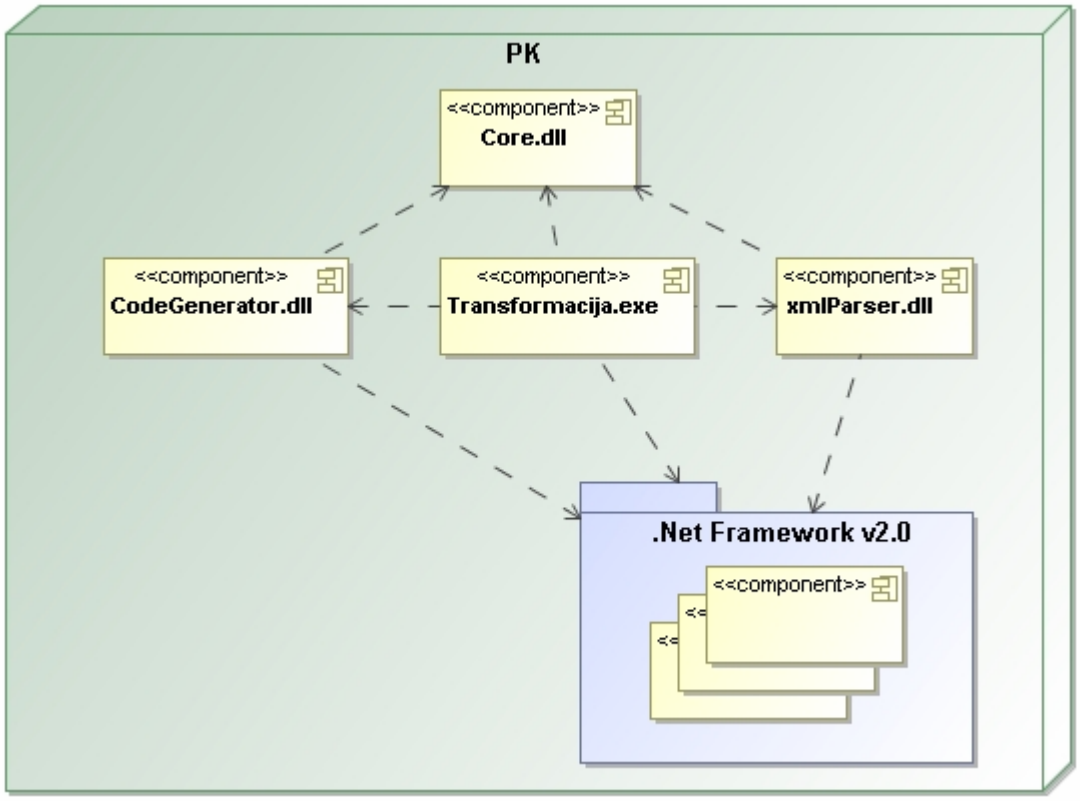
Komponentinis modelis sukurtas panaudojus profailą bei komponentinio modelio diagramą (žr. 32 pav.).



32 pav. Komponentinis modelis

4.3. Komponentų ir įdiegimo modeliai, komponentų specifikacijos

Komponentų diagramoje parodytas fizinis sistemos vaizdas: komponentai bei jų tarpusavio priklausomybės. Įdiegimo diagramoje atvaizduojami komponentai, kurių pagalba sistema veikia. Kad veiktų ši sistema, kompiuteryje būtinai turi būti įrašytas programinių klasių biblioteka „.Net Framework v2.0“, aišku tuo pačiu ir operacinė sistema „Windows“. Diegimo diagrama pateikta 33 paveikslėlyje.



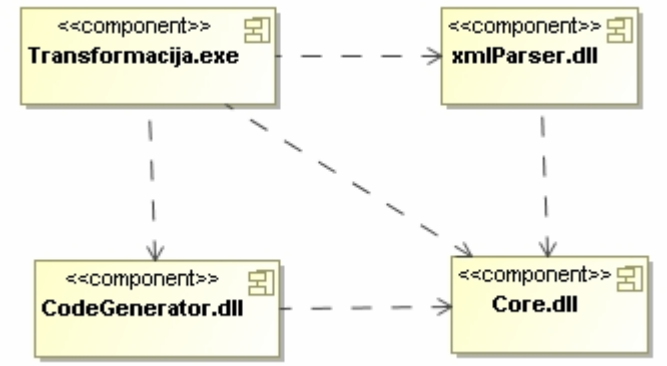
33 pav. Diegimo diagrama

Pagrindinis komponentas, siejantis visą sistemą yra „Transformacija.exe“. Jis naudoja dvi programines bibliotekas „xmlParser.dll“ bei „CodeGenerator.dll“. Komponentą „Core.dll“ naudoja visi trys likę komponentai.

- xmlParser.dll – realizuotos funkcijos, susijusios su paketu „Magic Draw UML 12.5“ užsaugoto projekto („.xml“ formatu) failo struktūros nagrinėjimu, skaitymu, rašymu ir t.t.
- CodeGenerator.dll – atlieka funkcijas, susijusias su programinių komponentų generavimu.

- Core.dll – bendros duomenų struktūros, reikalingos visiem trim komponentams.

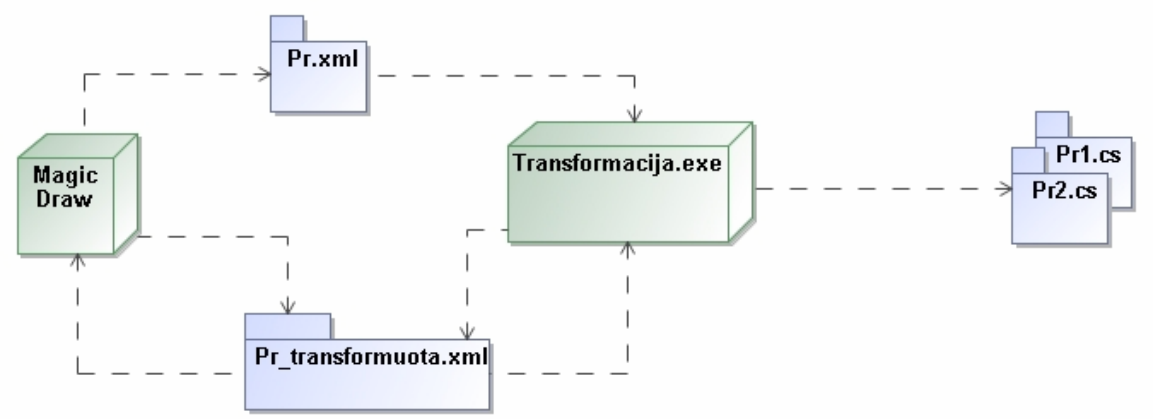
Komponentų diagrama pateikiama 34 paveikslėlyje.



34 pav. Komponentų diagrama

4.4. Sistemos testavimas, duomenys ir rezultatai

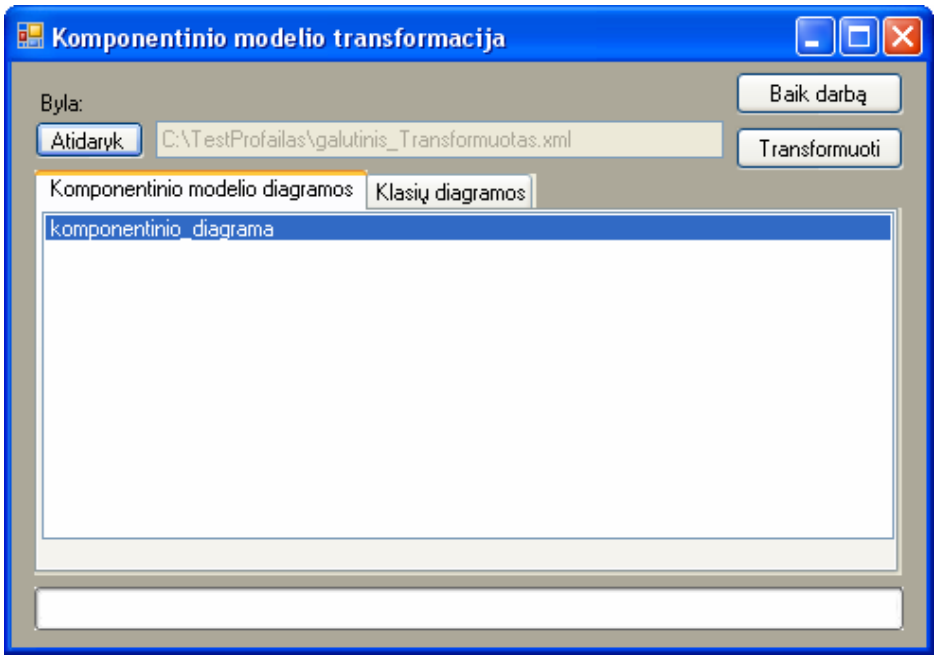
Sistemos veikimo principinė schema pateikiama šiame 35 paveikslėlyje.



35 pav. Testavimo scenarijus

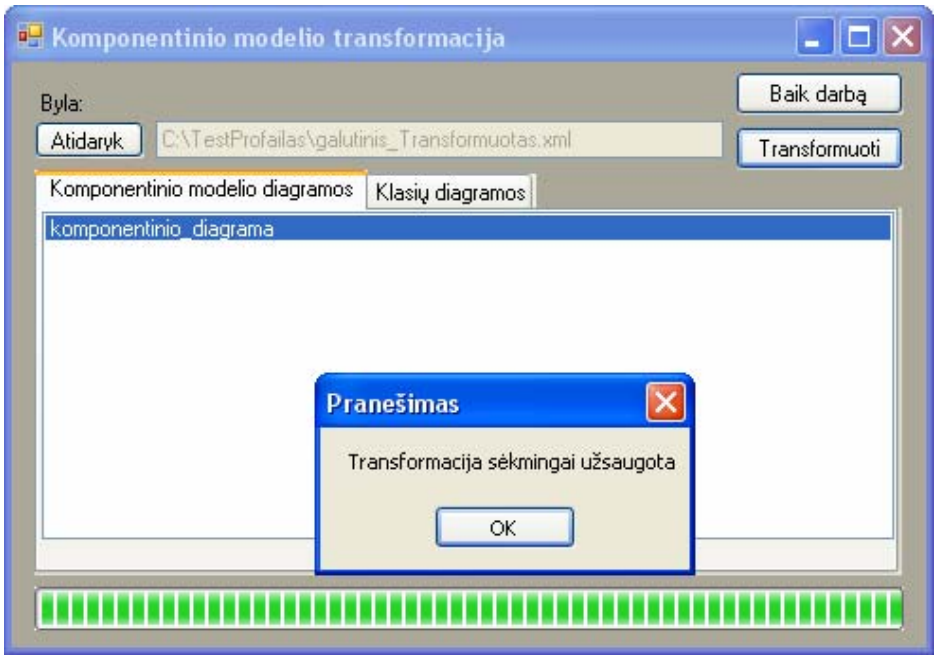
Sistema testuojama taip:

1. Sukuriamas „Magic Draw UML 12.5“ projektas, jame sukuriamas komponentinis modelis. Visa tai atliekama pagal aprašymą. Aprašymas pateiktas skyriuje „*Komponentinio modelio profailas.*“
2. „Magic Draw UML 12.5“ projektas išsaugomas *.xml formatu, pvz. „magistrinis.xml“.
3. „Magic Draw UML 12.5“ projektas atidaromas su aplikacija „Transformacija.exe“ (36 pav.).



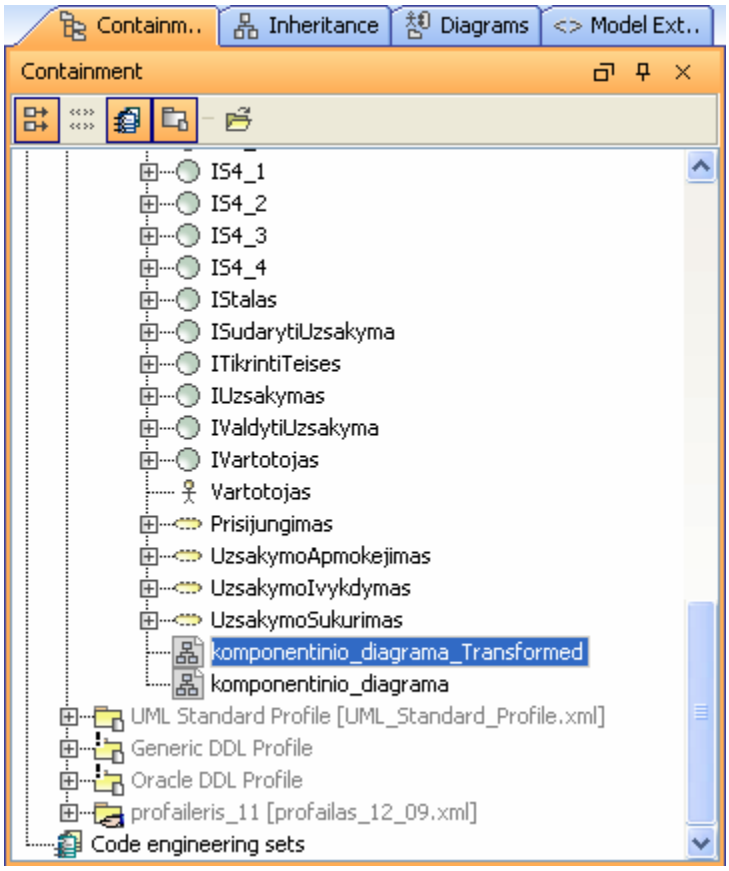
36 pav. „Magic Draw UML 12.5“ projektas, užkrautas Transformacija.exe faile

4. Pasirenkamas norimas transformuoti modelis iš atsiradusio sąrašo (šiuo atveju komponentinio_diagrama).
5. Atliekama komponentinio IS modelio transformacija į IS klasių modelį. Transformuota schema užsaugoma *.xml formatu, pvz. „magistrinis_Transformuotas.xml“ (37 pav.).



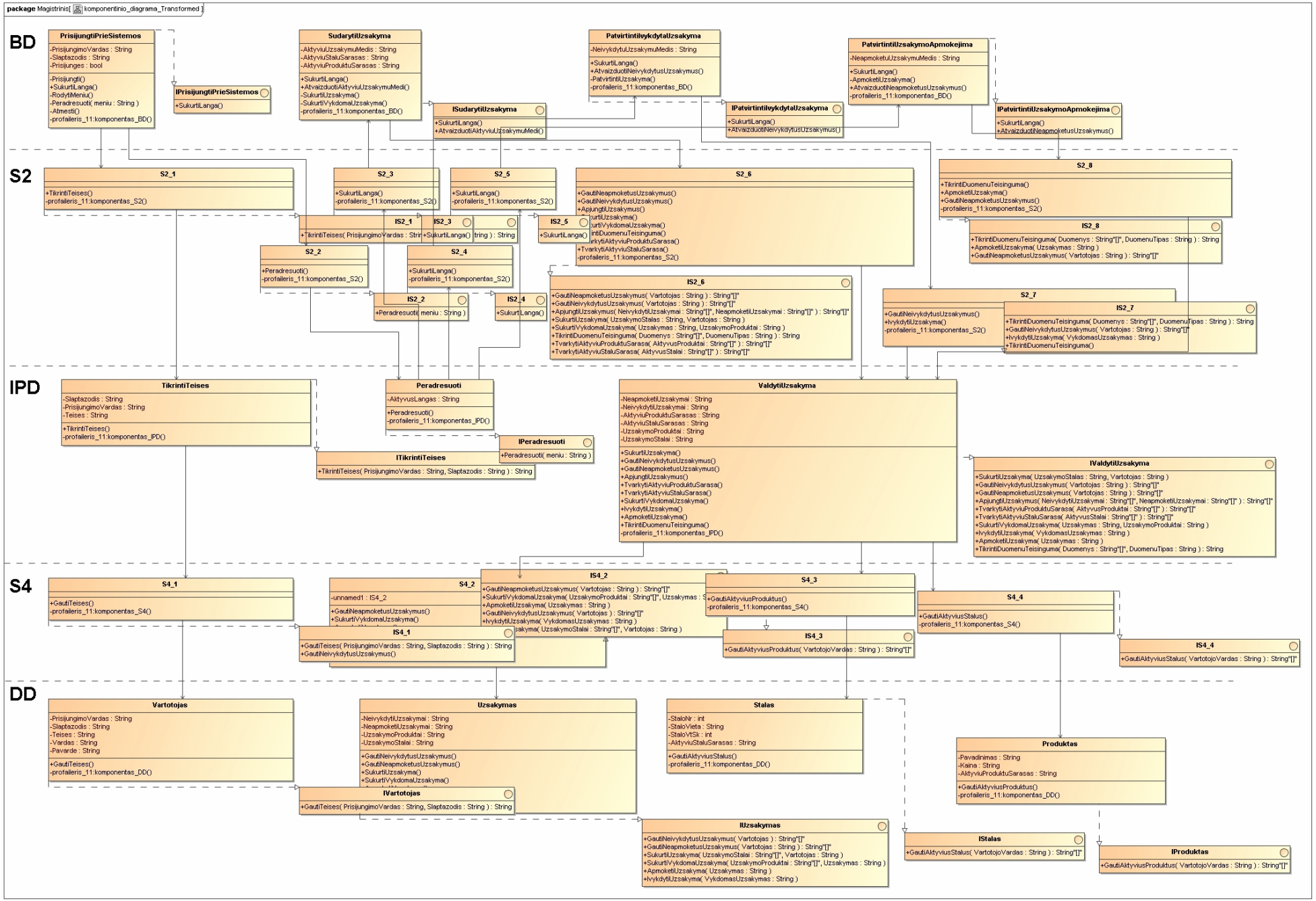
37 pav. Transformacija sėkmingai įvykdyta

6. Atidaromas transformuotas failas su „Magic Draw UML 12.5“. Transformuota schema randama tame pačiame kataloge, kur buvo pradinis modelis (38 pav.). Jai prie pavadinimo pridedama galūnė „_Transformed“.



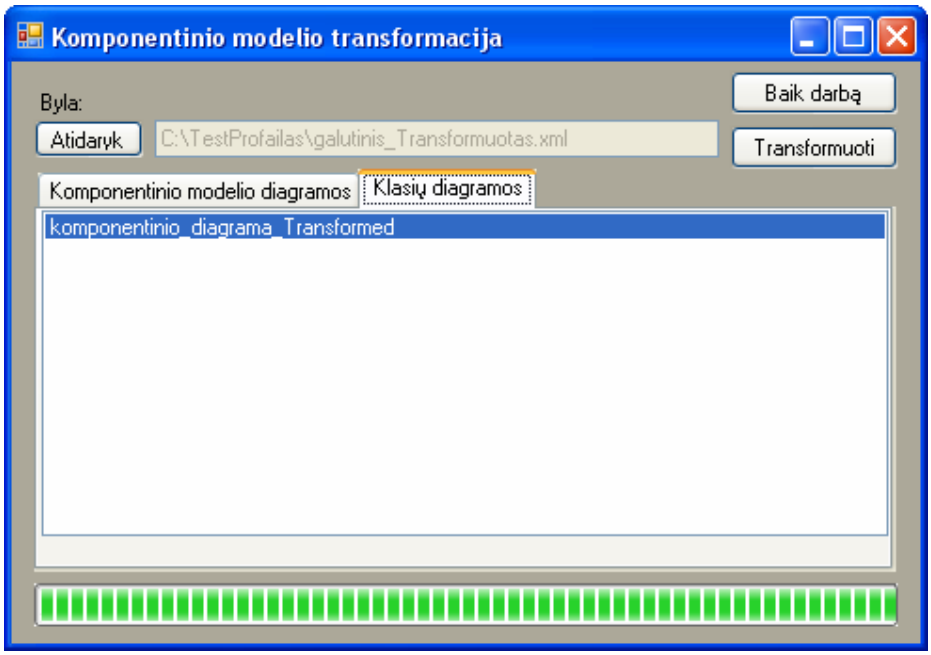
38 pav. Transformuotos diagramos vieta

Transformuotos klasių diagramos pavyzdys pateikiamas 39 paveikslėlyje.



39 pav. Transformuotas komponentinis modelis į klasių diagramą

7. Atidaromas transformuotas „*.xml“ failas su programine įranga „Transformacija.exe“, pateikiamas šiame faile esamų modifikuotų klasių diagramų sąrašas (40 pav.).

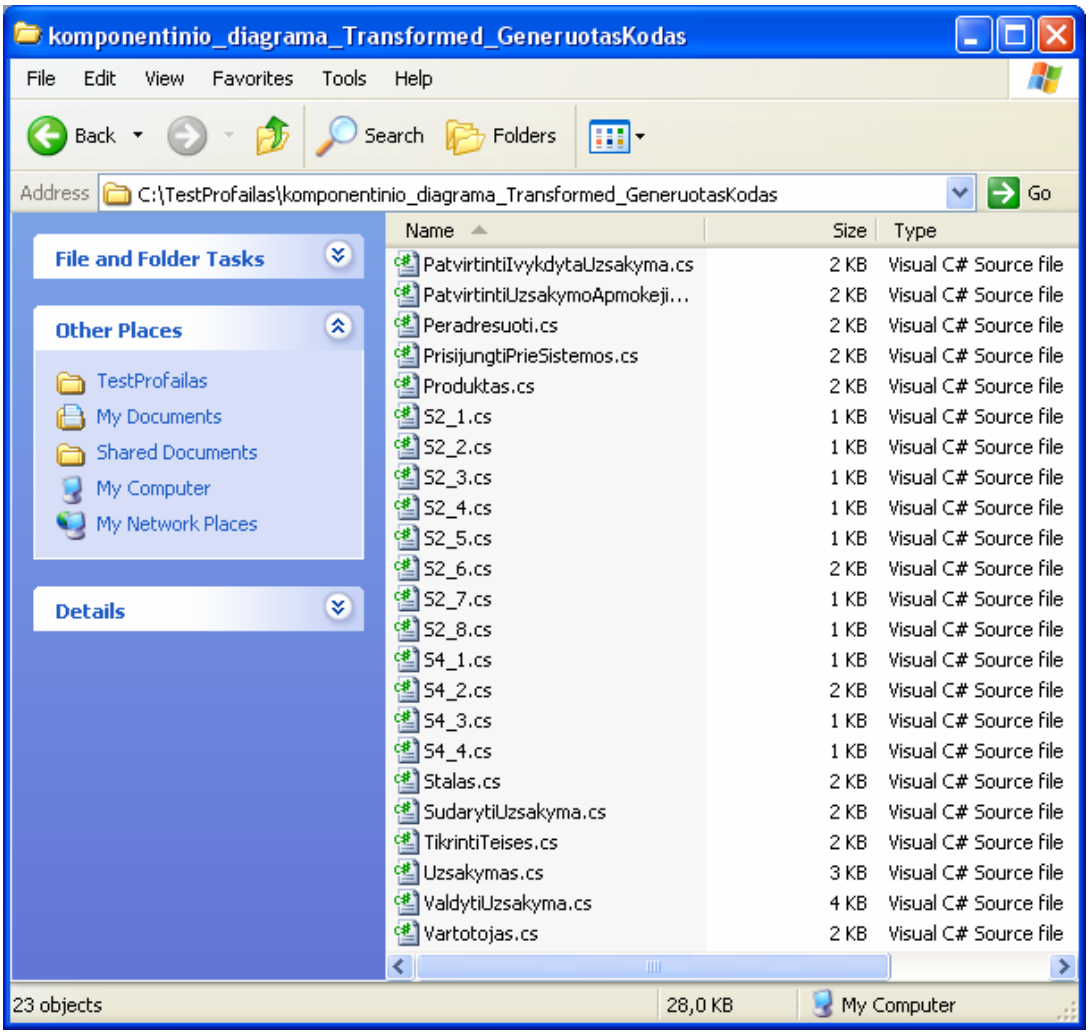


40 pav. Transformuotas „Magic Draw“ projektas, užkrautas Transformacija.exe faile

8. Pasirinkus norimą transformuoti diagramą, spaudžiamas mygtukas „Transformuoti“. Atlikus generavimą (41 pav.), kietajame diske, sukuriamas aplankas.

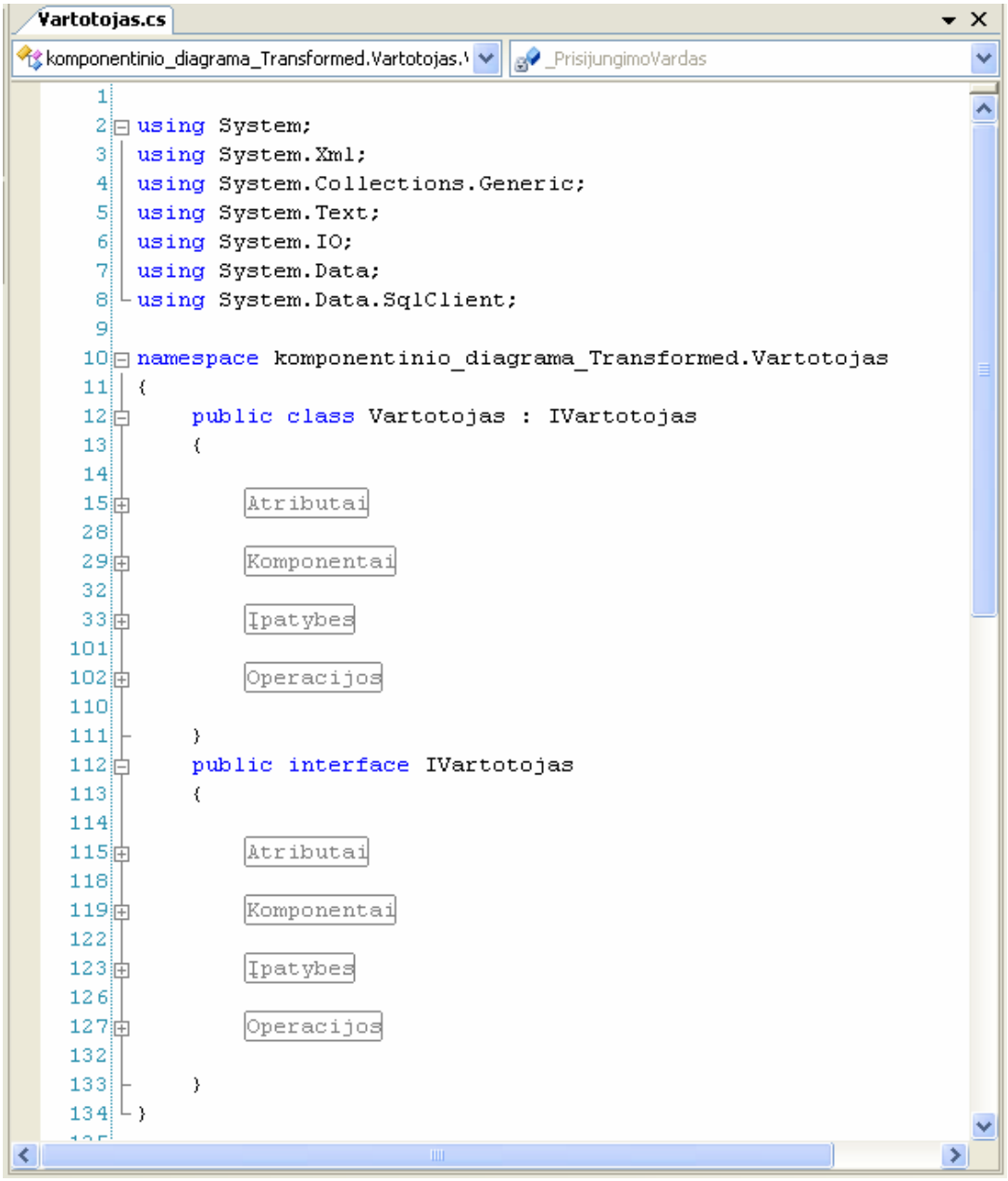
41 pav. Kodas sugeneruotas sėkmingai

Aplankas sukuriamas toje vietoje, kur randasi transformuojamas „*.xml“ failas (pavyzdyje: magistrinis_Transformuotas.xml), kurio pavadinimas sudaromas iš transformuoto modelio pavadinimo, kurio gale pridedamas žodis „_GeneruotasKodas“ (pavyzdžio atveju tai būtų „komponentinio_diagrama_Transformed_GeneruotasKodas“, 42 pav.).



42 pav. Aplankas, kuriame užsaugoti sugeneruoti komponentai

Šiame aplanke randamas sugeneruotų komponentų kodas (43 pav.).



43 pav. Sugeneruoto komponento programinis kodas

4.5. Reikalavimai sistemos funkcionavimo palaikymui bei sistemos diegimui būtini žingsniai

Sistemos moduliui, kuriame realizuota transformacija iš paketu „Magic Draw UML 12.5“ užsaugoto projekto („.xml“ formatu) failo į programinį kodą ir transformacija iš komponentinio modelio į klasių modelį, turi būti įdiegta operacinė sistema iš „Windows“ šeimos. Kadangi sistema realizuota naudojantis „MS Visual Studio 2005“ įrankiu, naudojant

„C#“ programavimo kalbą, kompiuteryje turi būti įdiegtas „*.Net Framework v2.0“* . Jeigu tame pačiame kompiuteryje bus modeliuojamas taip pat ir komponentinis modelis, jame turi būti įrašytas modeliavimo įrankis „Magic Draw UML 12.5“, pageidautina naujausia versija (šiuo metu tai 14.0) [8]. Serverio techninės įrangos reikalavimai yra tokie, kaip paminėtų technologijų funkcionavimo minimalūs techniniai reikalavimai. Kompiuteryje yra būtina turėti atitinkamas įdiegimo teises.

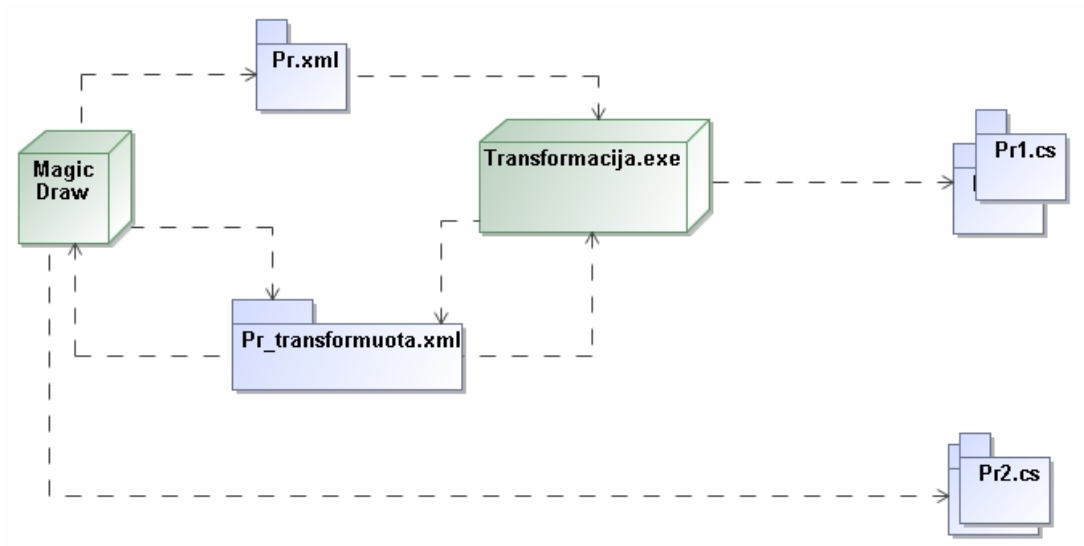
5. KOMPONENTINIO IS MODELIO TRANSFORMAVIMO SISTEMOS EKSPERIMENTAS

5.1. Eksperimento eiga

Eksperimento tikslas yra palyginti įprastu būdu modeliuotos sistemos sugeneruotą programinį kodą su komponentiniu modeliavimo principu sumodeliuotos sistemos sugeneruotu programiniu kodu. Eksperimento eiga yra pateikiama 44 paveiksle. „Magic Draw UML 12.5“ paketu, naudojantis sukurtu komponentinio modelio „profailu“, sukuriamas sistemos komponentinis modelis, jis yra užsaugomas „xml“ pavidale (44 paveiksle „Pr.xml“).

Su programine įranga („Transformacija.exe“) sugeneruojama sistemos modifikuota klasių diagrama, bei užsaugoma faile (44 paveiksle „Pr_transformuota.xml“), ši klasių diagrama užkraunama į „Magic Draw UML 12.5“ paketą bei su juo sugeneruojamas programinis kodas. Tuo pat metu iš modifikuotos klasių diagramos su programine įranga „Transformacija.exe“ sugeneruojamas programinis kodas.

Šiais dviem budais generuotas programinis kodas yra palyginamas. Eksperimento rezultatai pateikiami skyriaus pabaigoje.



44 pav. Eksperimento eiga

Paveiksluose 45, 46, 47, 48 pateikiami sugeneruotų programine įranga „Transformacija.exe“ komponentų programinis kodas, jis lyginamas su „Magic Draw UML 12.5“ paketu sugeneruotų klasių aprašais (49 pav.).

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Web;

using komponentinio_diagrama_Transformed.S2_1;
using komponentinio_diagrama_Transformed.S2_2;
namespace komponentinio_diagrama_Transformed.PrisijungtiPrieSistemos
{
		public class PrisijungtiPrieSistemos : Form, IPrisijungtiPrieSistemos
		{

			#region Atributai
			private string _PrisijungimoVardas;
			private string _Slaptazodis;
			private bool _Prisijunges;
			#endregion

			#region Komponentai
			private S2_1 _S2_1 = null;
			private S2_2 _S2_2 = null;
			#endregion

			#region Ipatybes
			private string PrisijungimoVardas
			{
				set
				{
					_PrisijungimoVardas = value;
				}
				get
				{
					 return _PrisijungimoVardas;
				}
			}
			private string Slaptazodis
			{
				set
				{
					_Slaptazodis = value;

				}
				get
				{
					 return _Slaptazodis;
				}
			}
			private bool Prisijunges
			{
				set
				{
					_Prisijunges = value;
				}
				get
				{
					 return _Prisijunges;
				}
			}
			#endregion
			#region Operacijos
			private void Prisijungti()
			{
			}
			public void SukurtiLanga()
			{
			}
			private void RodytiMeniu()
			{
			}
			private void Peradresuoti(string meniu)
			{
			}
			private void Atmesti()
			{
			}
			#endregion
		}
		public interface IPrisijungtiPrieSistemos
		{
			#region Atributai

			#endregion
			#region Komponentai
			#endregion
			#region Ipatybes
			#endregion
			#region Operacijos
```

```
                        public void SukurtiLanga();
                        #endregion
                }
}
```

45 pav. paveikslas. „BD“ lygio komponentas „PrisijungtiPrieSistemos“, sugeneruotas su programine įranga „Transformuoti.exe“.

```
using System;
using System.Data;
using System.Collections;
using System.Diagnostics;

using komponentinio_diagrama_Transformed.S2_3;
using komponentinio_diagrama_Transformed.S2_4;
using komponentinio_diagrama_Transformed.S2_5;
namespace komponentinio_diagrama_Transformed.Peradresuoti
{
                public class Peradresuoti : IPeradresuoti
                {
                        #region Atributai
                        private string _AktyvusLangas;
                        #endregion
                        #region Komponentai
                        private S2_3 _S2_3 = null;
                        private S2_4 _S2_4 = null;
                        private S2_5 _S2_5 = null;
                        #endregion
                        #region Įpatybes
                        private string AktyvusLangas
                        {
                                set
                                {
                                        _AktyvusLangas = value;
                                }
                                get
                                {
                                         return _AktyvusLangas;
                                }
                        }
                        #endregion
                        #region Operacijos
                        public void Peradresuoti()
                        {
                        }
                        #endregion

                }
                public interface IPeradresuoti
                {
                        #region Atributai
                        #endregion
                        #region Komponentai
                        #endregion
                        #region Įpatybes
                        #endregion
                        #region Operacijos
                        public void Peradresuoti(string meniu);
                        #endregion
                }
}
```

46 pav. „IPD“ lygio komponentas „Peradresuoti“, sugeneruotas su programine įranga „Transformuoti.exe“.

```
using System;
using System.Xml;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Data;
using System.Data.SqlClient;

namespace komponentinio_diagrama_Transformed.Vartotojas
{
                public class Vartotojas : IVartotojas
                {
                        #region Atributai
                        private string _PrisijungimoVardas;
                        private string _Slaptazodis;
                        private string _Teises;
                        private string _Vardas;
```

```
                        private string _Pavarde;
                        #endregion
                        #region Komponentai
                        #endregion
                        #region Ipatybes
                        private string PrisijungimoVardas
                        {
                                set
                                {
                                        _PrisijungimoVardas = value;
                                }
                                get
                                {
                                         return _PrisijungimoVardas;
                                }
                        }
                        private string Slaptazodis
                        {
                                set
                                {
                                        _Slaptazodis = value;
                                }
                                get
                                {
                                         return _Slaptazodis;
                                }
                        }
                        private string Teises
                        {
                                set
                                {
                                        _Teises = value;
                                }
                                get
                                {
                                         return _Teises;
                                }
                        }
                        private string Vardas
                        {
                                set
                                {
                                        _Vardas = value;
                                }
                                get
                                {
                                         return _Vardas;
                                }
                        }
                        private string Pavarde
                        {
                                set
                                {
                                        _Pavarde = value;
                                }
                                get
                                {
                                         return _Pavarde;
                                }
                        }
                        #endregion
                        #region Operacijos
                        public void GautiTeises()
                        {
                        }
                        #endregion
                }
                public interface IVartotojas
                {
                        #region Atributai
                        #endregion
                        #region Komponentai
                        #endregion
                        #region Ipatybes
                        #endregion
                        #region Operacijos
                        public string GautiTeises(string PrisijungimoVardas);
                        #endregion
                }
}
```

47 pav. „DD“ lygio komponentas „Vartotojas“, sugeneruotas su programine įranga „Transformuoti.exe“.

```
using System;
using System.Data;
using System.Collections;
using System.Diagnostics;
using komponentinio_diagrama_Transformed.TikrintiTeises;
namespace komponentinio_diagrama_Transformed.S2_1
{
                public class S2_1 : IS2_1
                {
                                #region Atributai
                                #endregion
                                #region Komponentai
                                private TikrintiTeises _TikrintiTeises = null;
                                #endregion
                                #region Įpatybes
                                #endregion
                                #region Operacijos
                                public void TikrintiTeises()
                                {
                                }
                                #endregion
                }
                public interface IS2_1
                {
                                #region Atributai
                                #endregion
                                #region Komponentai
                                #endregion
                                #region Įpatybes
                                #endregion
                                #region Operacijos
                                public    string    TikrintiTeises(string    PrisijungimoVardas,    string
Slaptazodis);
                                #endregion
                }
}
```

48 pav. Sąsajos lygio komponentas „S2_1“, sugeneruotas su programine įranga „Transformuoti.exe“.

```
namespace komponentinio_modelio_diagrama_Transformed.Magistrinis
{public class PrisijungtiPrieSistemos : IPrisijungtiPrieSistemos
                {
                                String PrisijungimoVardas;
                                String Slaptazodis;
                                bool Prisijunges;
                                S2_1 ;
                                S2_2 ;
                                S2_1 ;
                                S2_2 ;
                                void Prisijungti(  )
                                {
                                }
                                public void SukurtiLanga(  )
                                {
                                }
                                void RodytiMeniu(  )
                                {
                                }
                                void Peradresuoti( String meniu )
                                {
                                }
                                void Atmesti(  )
                                {
                                }
                }
}
```

49 pav. Klasė „PrisijungtiPrieSistemos“ kodas, sugeneruotas su integruotu „Magic Draw UML 12.5“ programinio kodo generatoriumi.

5.2. Eksperimento rezultatai

Lyginant programine įranga „Transformacija.exe“ sugeneruotų komponentų programinį kodą su „Magic Draw UML 12.5“ programine įranga generuotų klasių aprašais, galime konstatuoti, kad:

- Programinis kodas suskirstytas į regionus, kad programinis kodas būtų lengviau skaitomas bei papildomas.
- Kiekvienam atributui sukurta ypatybė, dėl to atributus yra lengviau naudoti, nutrynus „set“ arba „get“ ypatybę, galima užtikrinti tik rašymo ar tik skaitymo leidimą atributui.
- Kiekvieno komponento pakartotiniam panaudojimui užtikrinti sukurta sąsaja.
- Nustatyti komponente naudojami kiti komponentai, sukurtos nuorodos į tuos komponentus.
- Kiekvienam komponentui nustatyti paveldimi komponentai bei interface‘ai atsižvelgiant į komponento tipą.
- Kiekvienam komponentui nustatyti naudojami standartiniai komponentai („using“ sekcija).

6. IŠVADOS

1. Išanalizuotos Magic Draw UML 12.5 galimybės ir nustatyta, kad stinga integralumo tarp veiklos modeliavimo ir detalaus IS projektavimo etapų.
2. Išnagrinėtos CASE priemonės (Magic Draw UML 12.5, MS Visio 2003, ProVision Workbench 3.0). Pasirinkta Magic Draw UML 12.5 kaip pasaulyje aukštą reitingą turinti CASE sistema, firma „No Magic“ suteikė visas galimybes naudotis šiuo paketu.
3. Sukurtas Magic Draw UML 12.5 profailas (komponentiniam IS projektavimui), tinkantis daugkartiniam panaudojimui. Tai galėtų būti pasiūlytas kaip paketo Magic Draw UML 12.5 išplėtimas (plug in).
4. Sudarytas komponentinio IS modelio transformavimo į UML klasių modelį algoritmas.
5. Sudarytas programinio kodo generavimo iš modifikuotos UML klasių diagramos algoritmas.
6. Realizuota komponentinio modelio transformacijos į klasių modelį programinė įranga, sukurta MS Visual Studio .NET 2005, kuri sugeba atlikti komponentinio IS modelio transformavimą į UML klasių modelį.
7. Realizuota taikomųjų uždavinių kodo generavimo iš modifikuoto klasių modelio, programinė įranga, sukurta MS Visual Studio .NET 2005, kuri sugeba atlikti programinio kodo generavimą iš modifikuoto UML klasių modelio.
8. Parengtas komponentinio IS projekto kūrimo pavyzdys, naudojant sukurtą metodą ir programines priemones. Sukurta restorano užsakymo valdymo informacinė sistema komponentinio modeliavimo principu. Siekiant sukurti komponentų sąveikos modelį, pasirinktas bei išnagrinėtas užsakymo valdymo procesas.
9. Atliktas eksperimentas, palygintas sukurta programine įranga sugeneruotų komponentų programinis kodas su „Magic Draw UML 12.5“ programine įranga generuotais klasių aprašais bei nustatyta, kad sukurta programine įranga generuotas programinis kodas yra tinkamesnis naudojimui.

7. LITERATŪRA

1. S. Gudas, E. Pakalnickas. Component-based modelling at the system design stage. [žiūrėta 2006-10-12]. Konferencijos „Informacinės technologijos 2006” pranešimų medžiaga.
2. S. Gudas, I. Lagzdinytė. Veiklos informacinės architektūros sudarymo technologijos elementai. [žiūrėta 2006-10-20]. Konferencijos „Informacinės technologijos 2001” pranešimų medžiaga.
3. E. Pakalnickas, S. Gudas. Informacijos sistemos projektavimas ir realizavimas komponentiniu metodu. [žiūrėta 2006-11-04]. Informacijos mokslai, 2003, t. 24, p. 59-68.
4. E. Pakalnickas, S. Gudas. Sąsajomis grįstas IS projektavimas. [žiūrėta 2007-03-12]. Konferencijos „Informacinės technologijos 2006” pranešimų medžiaga.
5. S. Gudas. The component-based information system requirements modeling. Business operation and its legal environment: processes, tendencies and results. Riga „Biznesa augstskola Turiba“ SIA 2002, p. 98-102. [žiūrėta 2007-04-20].
6. Meservy T. Fenstermacher K.D. Transforming Software Development: An MDA Road Map. [žiūrėta 2007-09-10]. Prieiga per internetą: http://www.compuware.com/dl/mdaroadmap.pdf
7. .Net Framework funkcijų aprašas. [žiūrėta 2007-10-13]. Prieiga per internetą: http://www.msdn.lt
8. Diagramų braižymo įrankio (Magic Draw UML 12.5) svetainė. [žiūrėta 2007-04-23]. Prieiga per internetą: www.magicdraw.com
9. „MS Visio“ braižymo programos aprašymas. [žiūrėta 2007-04-25]. Prieiga per internetą: http://office.microsoft.com/visio/
10. .Net Framework funkcijų panaudojimo pavyzdžiai. [žiūrėta 2007-10-13]. Prieiga per internetą: http://www.codeproject.com/

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

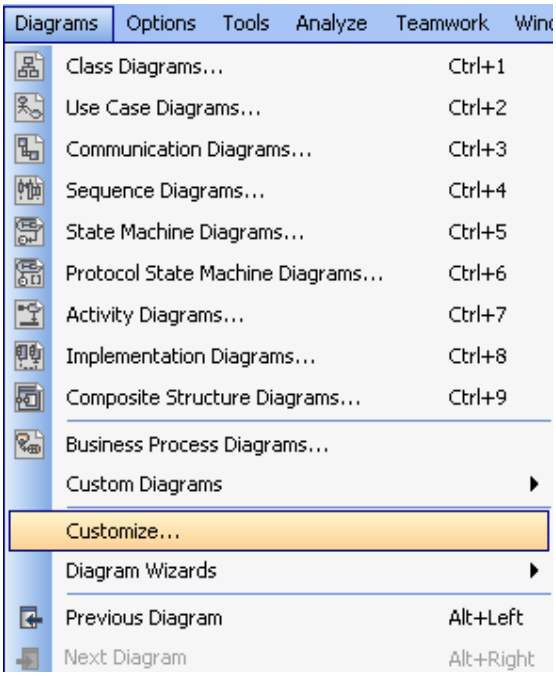
6 lentelė. Terminai ir santrumpos

Terminas / Santrumpa	Pilnas pavadinimas	Vertimas / Paaiškinimas
IS	Information System	Informacinė sistema
UML	Unified Modeling Language	Unifikuota / vieninga modeliavimo kalba
MDA	Model Driven Architecture	Modeliais grįsta architektūra
OMG	Object Management Group	Modeliavimo ir metaduomenų specifikacijų, standartų kūrimo konsorciumas
CASE	Computer Aided System Engeneering	Kompiuterizuota programų inžinerija
IA		Informacinė architektūra
VIA		Veiklos informacinė architektūra
BD		Biznio domenas
IPD		Informacinių procesų domenas
DD		Duomenų domenas
TPD		Technologinių procesų domenas
DVD		Darbo vietų domenas
KDV		Kompiuterizuota darbo vieta
S1, S2, S3 ir kt.		Sąsajų tipai
CF	Control Flow	Valdymo eiga
DFD	Data Flow Diagram	Duomenų srautų diagrama
BIM	Business Interaction Model	Biznio sąveikų modelis
Workflow model		Darbų sekų modelis
ER	Entity Relationship	Esybių ryšiai
XML	eXtensible Markup Language	Išplėstinė dokumentų aprašų kalba
XMI	XML Metadata Interchange	XML metaduomenų keitimosi formatas

9. PRIEDAI

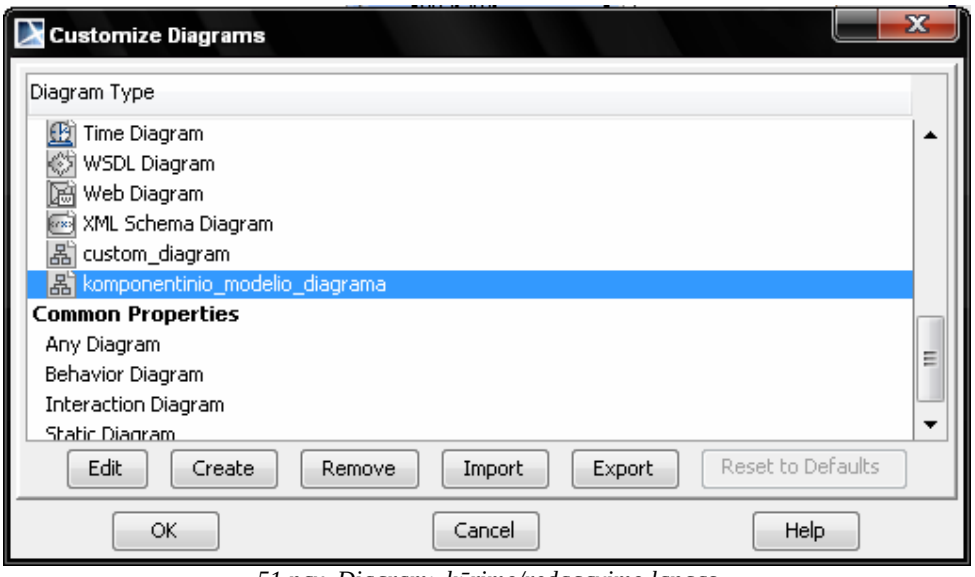
9.1. Priedas Nr.1 VARTOTOJO VADOVAS (komponentinio modelio diagramos sukūrimas)

Norint kurti naują komponentinio modelio diagramą pakete Magic Draw UML 12.5, meniu juostoje spaudžiama Diagrams→Customize (50 pav.).



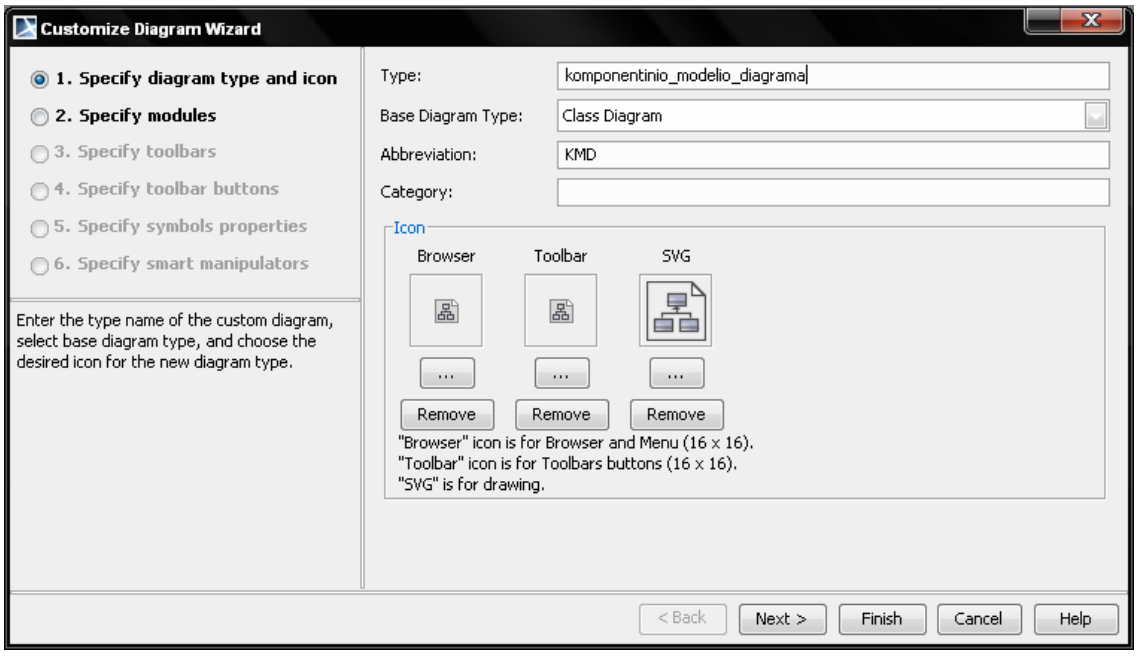
50 pav.Meniu komanda „Diagrams“

Galima redaguoti jau esamą diagramą (Edit funkcija) arba kurti naują (Create funkcija) (51 pav.).

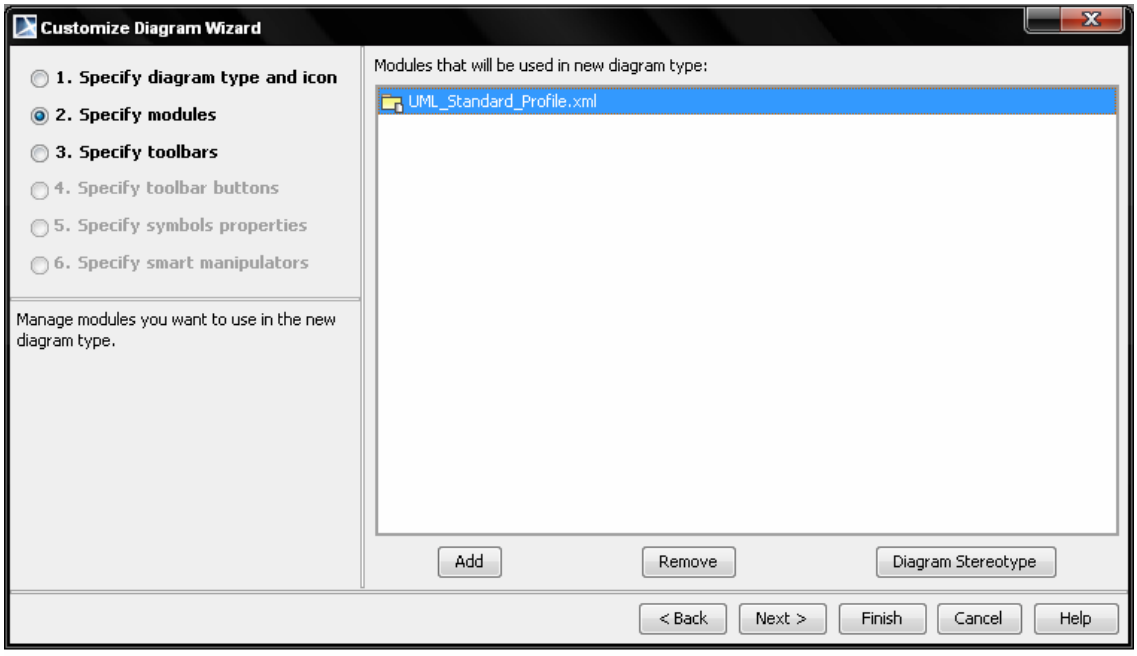


51 pav. Diagramų kūrimo/redagavimo langas

Šiuo atveju bus kuriama nauja diagrama, todėl spaudžiamas mygtukas „Create“ bei užpildomi laukai kaip parodyta 52 pav. Spaudžiama Next.

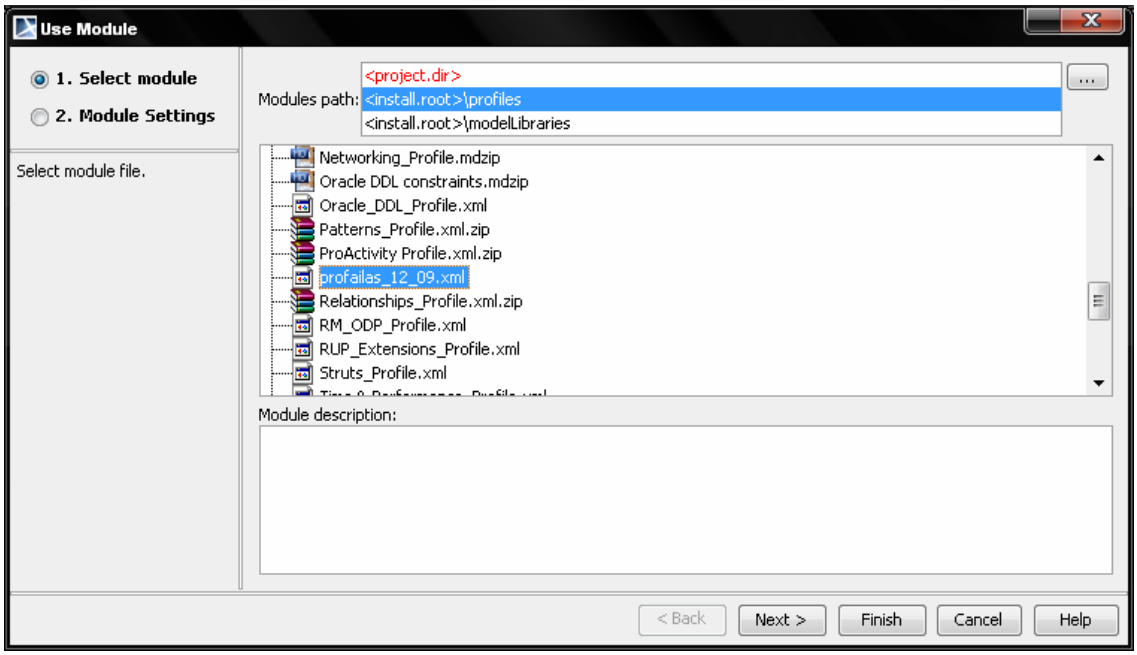


52 pav. Diagramos tipo ir ikonos parinkimo langas



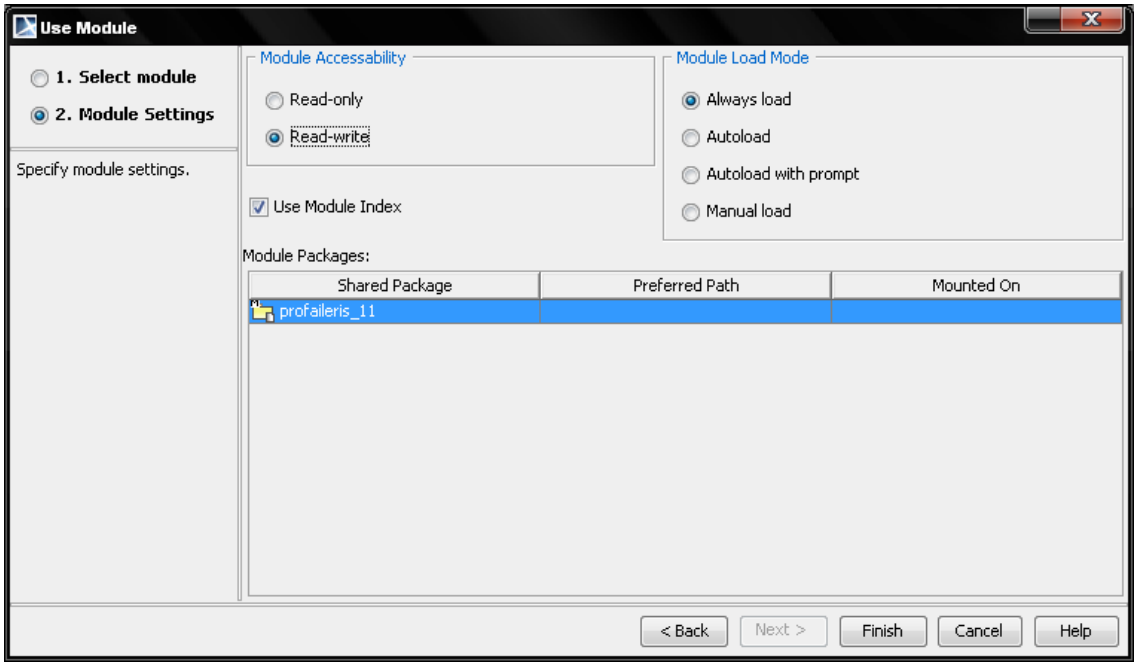
53 pav. Modulio pasirinkimo langas (modulis dar neįkeltas)

Spaudžiamas Add. Pasirenkamas modulis „profailas_12_09.xml“. Spaudžiama Next.



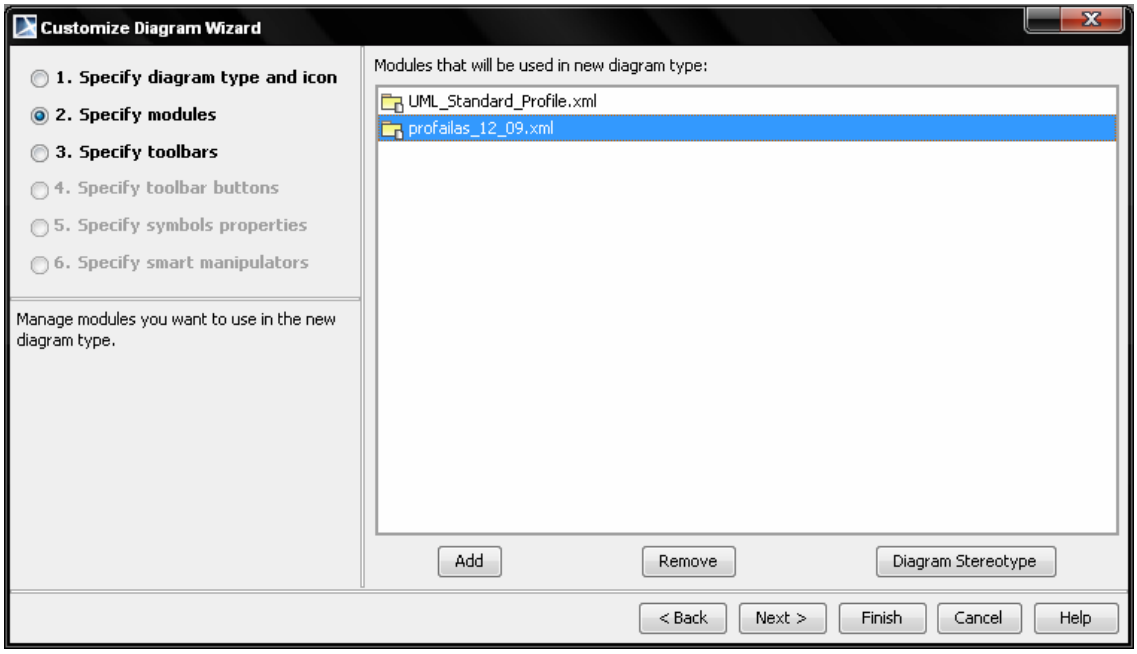
54 pav. Modulio parinkimas iš sąrašo

Pažymima, jog būtų galima modulį ne tik skaityti, bet ir redaguoti (uždedama varnelė ant „Read-write“). Spaudžiama Finish.



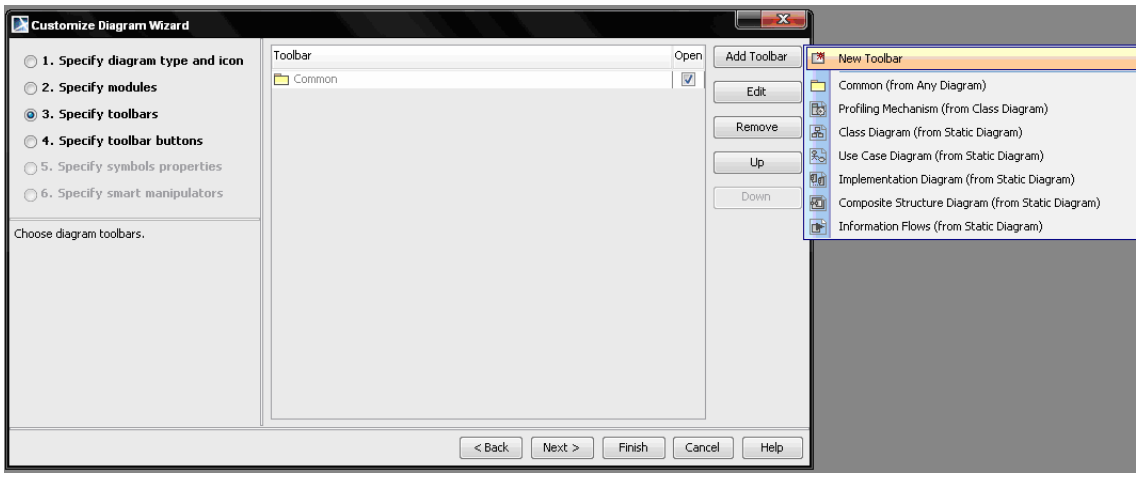
55 pav. Modulio nustatymų langas

Pažymimas atsiradęs modulis „profailas_12_09.xml“. Spaudžiama Next.



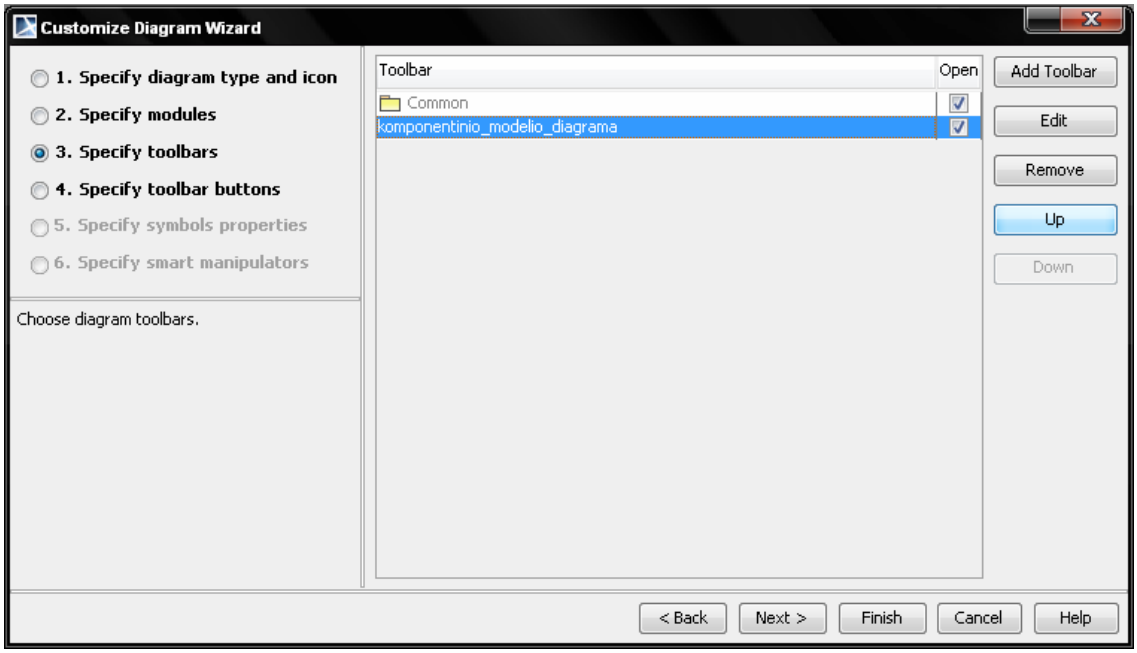
56 pav. Modulio pasirinkimo langas (modulis jau įkeltas)

Atsiradusioje lentelėje spaudžiama Add Toolbar → New Toolbar.



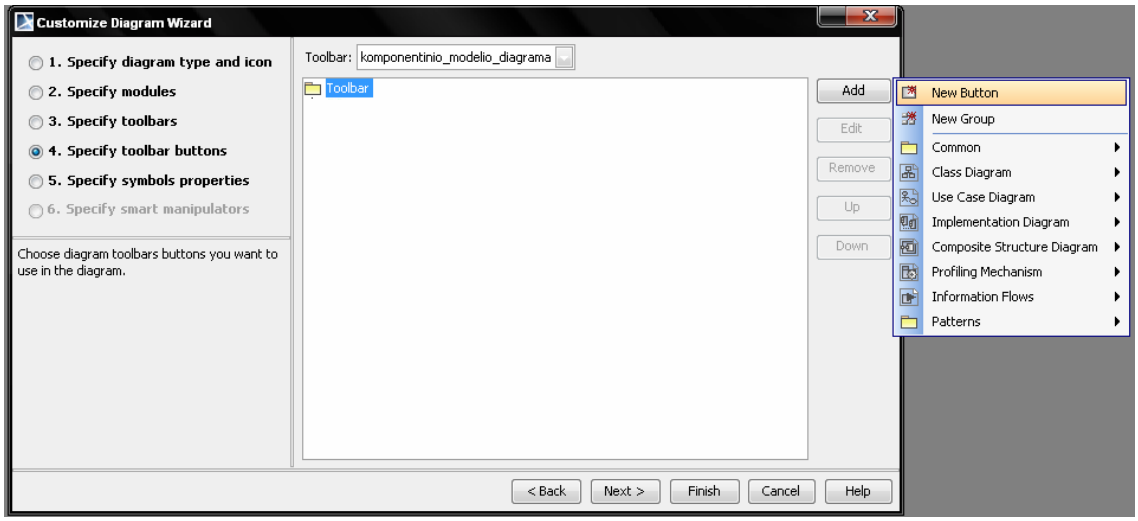
57 pav. Įrankių juostos kūrimo langas

Įvedamas įrankių juostos pavadinimas. Šiuo atveju – „komponentinio_modelio_diagrama“. Spaudžiama Next.



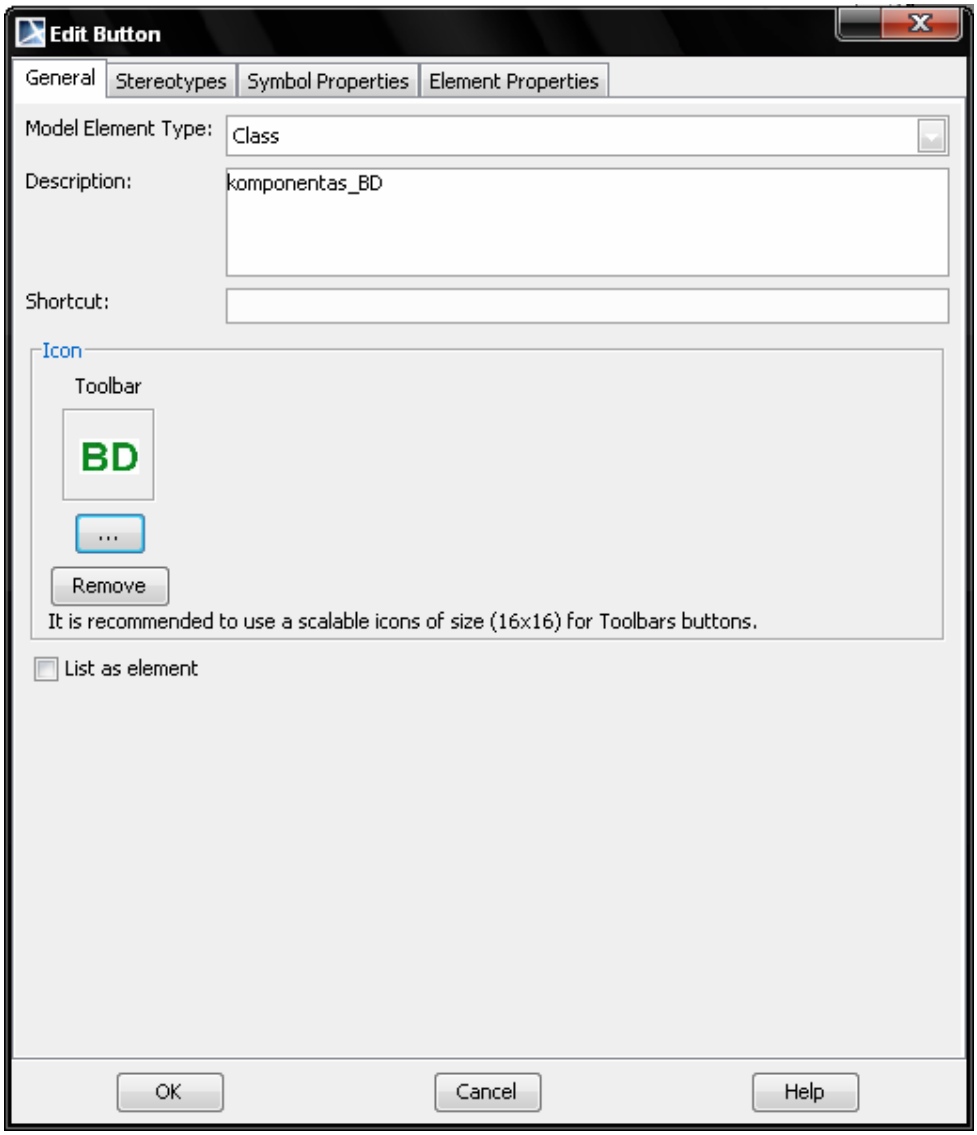
58 pav. Sukurtos įrankių juostos langas

Norint sukurti naują komponentą spaudžiama Add → New Button.



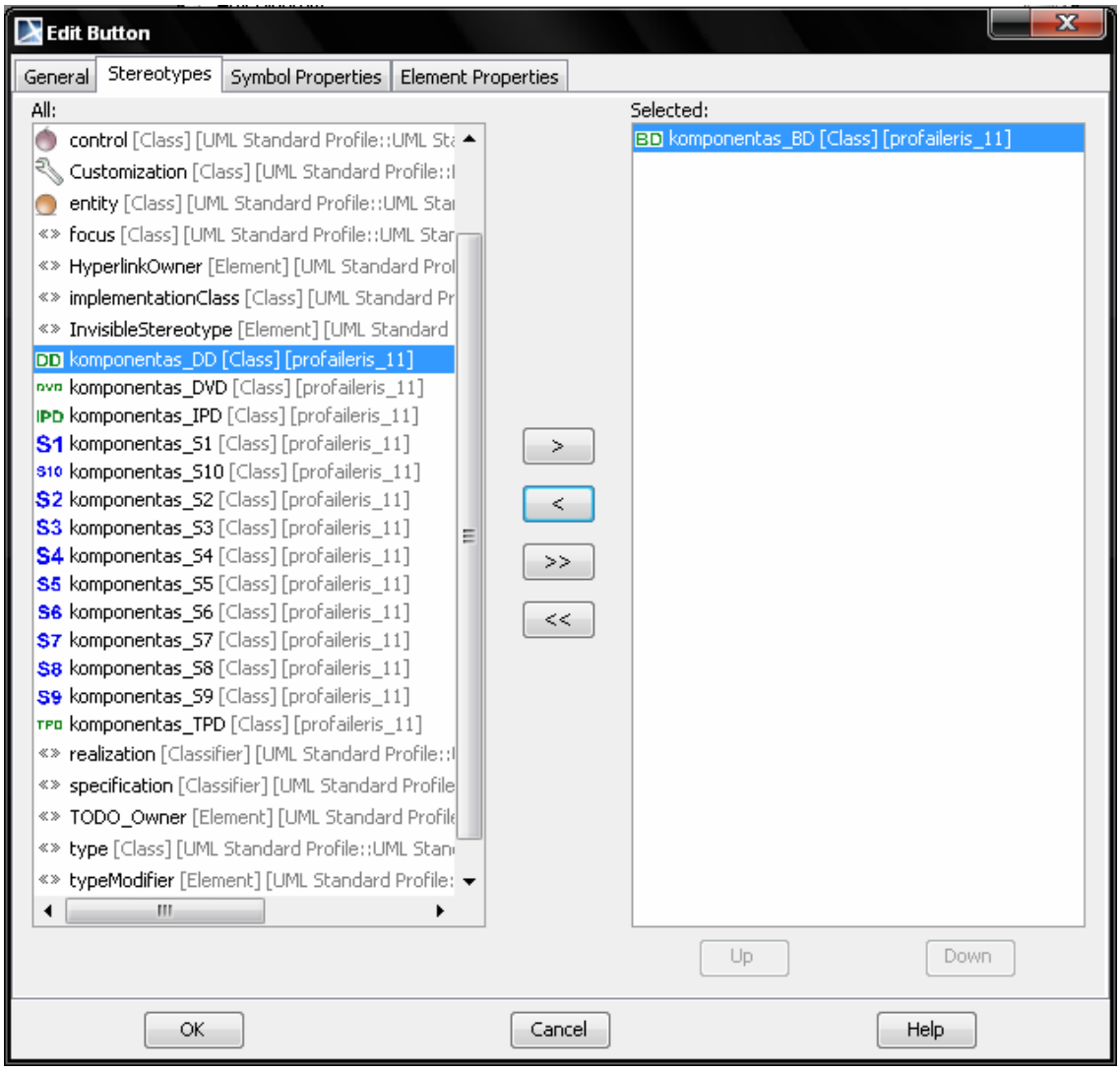
59 pav. Naujo komponento kūrimo langas

Pasirenkama Model Element Type – Class. Į „Description“ laukelį įvedamas komponento pavadinimas. Laukelyje „Icon“ parenkamas komponentą apibūdinantis paveiksliukas.



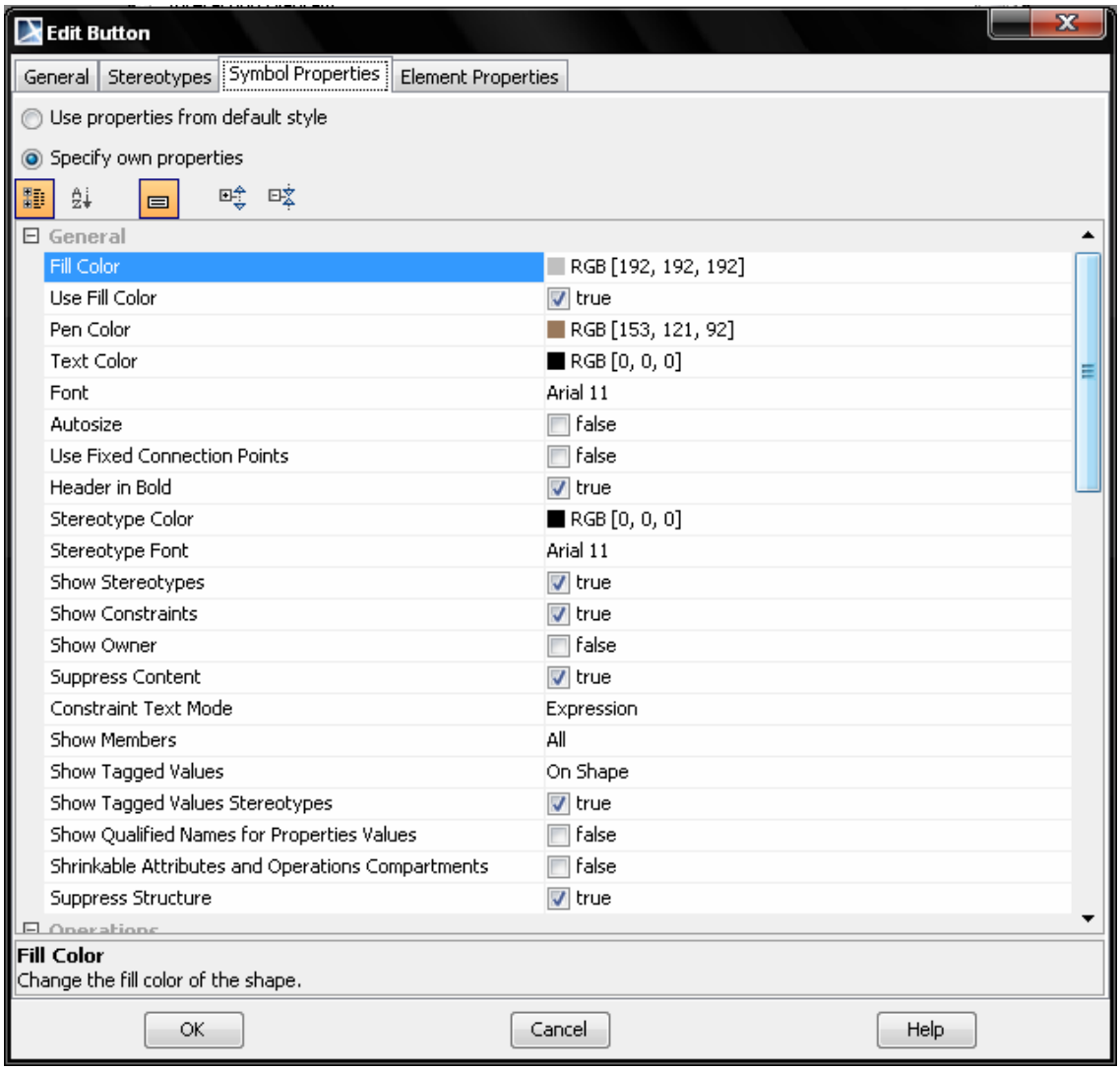
60 pav. Komponento redagavimo langas

Kortelėje „Stereotypes“ pasirenkamas komponentą apibūdinantis stereotipas. Spaudžiama OK.



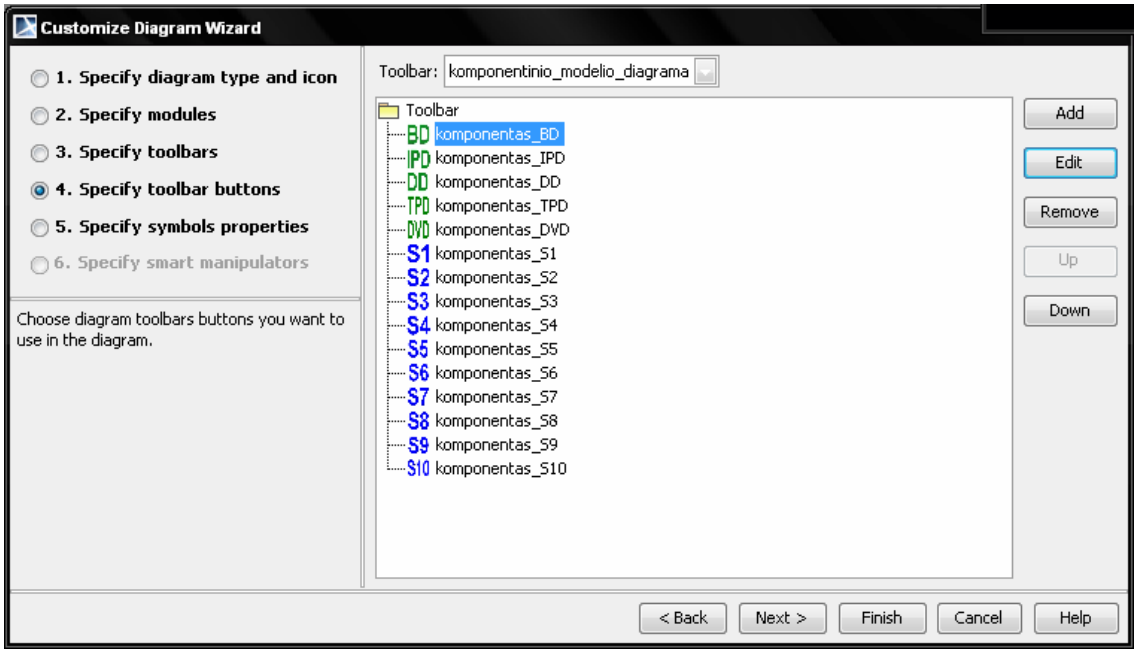
61 pav. Komponento kūrimo langas priskiriant stereotipus

Kortelėje „Symbol Properties“ galima parinkti įvairius komponento parametrus (teksto, fono spalvą ir pan.). Spaudžiama OK.



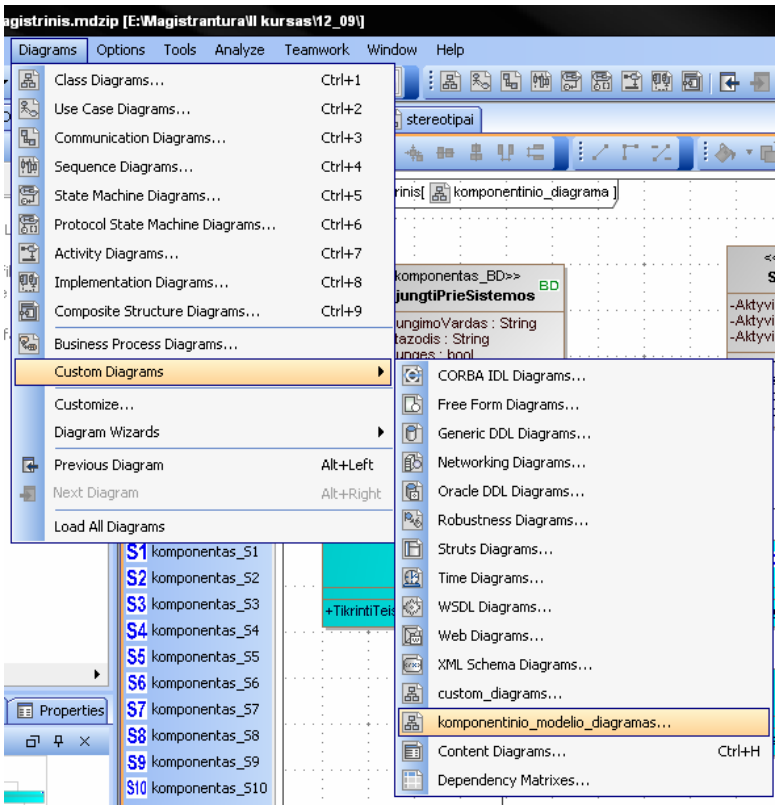
62 pav. Komponento parametrų pasirinkimo langas

Analogiškai sukeliami likę komponentai (pvz. komponentas_IPD, komponentas_DD, komponentas_S1 ir t.t.).



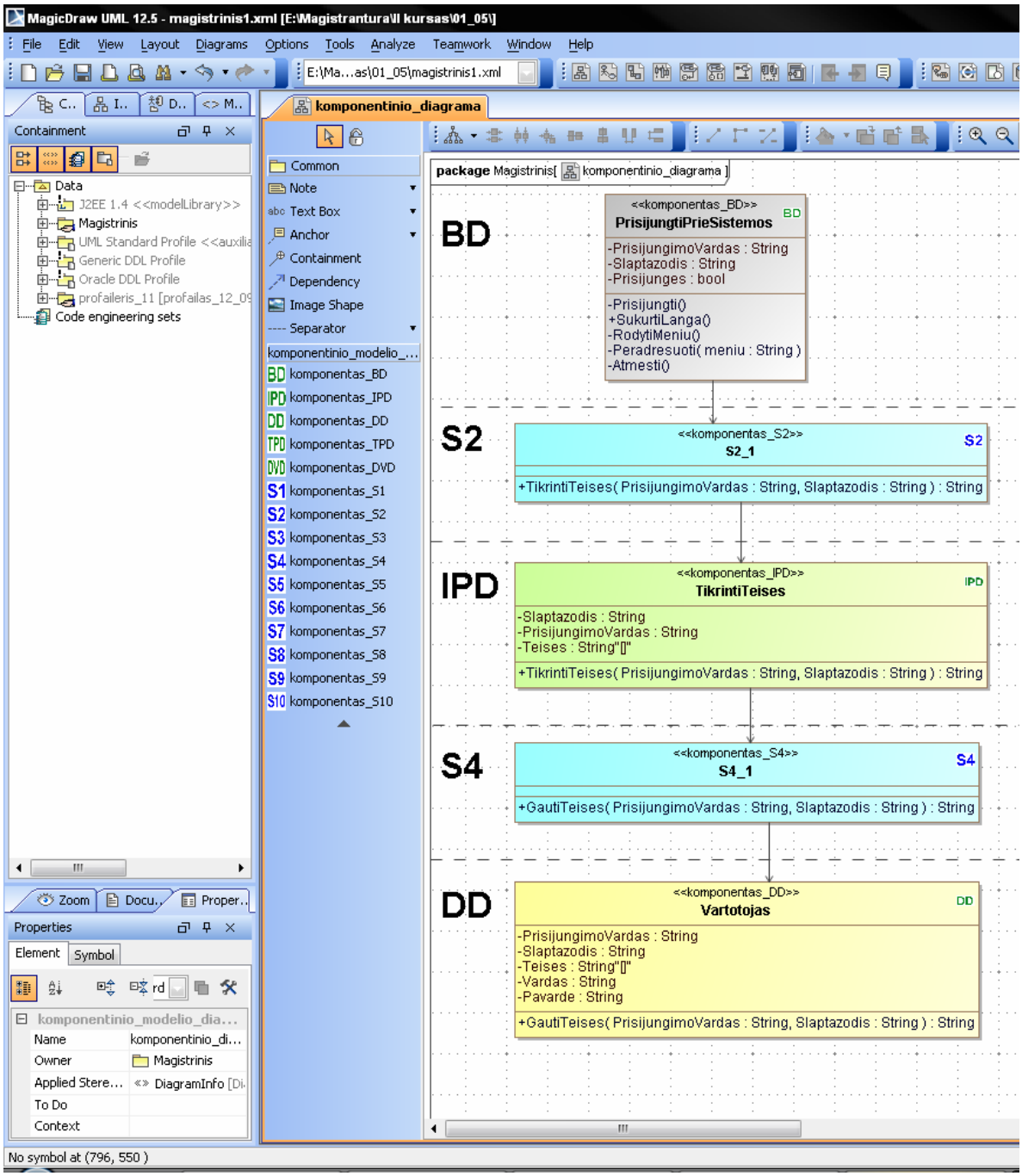
63 pav. Sukurtų komponentų langas

Spaudžiama Finish. „Magic Draw UML 12.5“ paketas reikalauja perkrauti projektą. Perkrovus galima naudotis sukurta diagrama (64 pav.).



64 pav. Sukurtos diagramos paleidimas

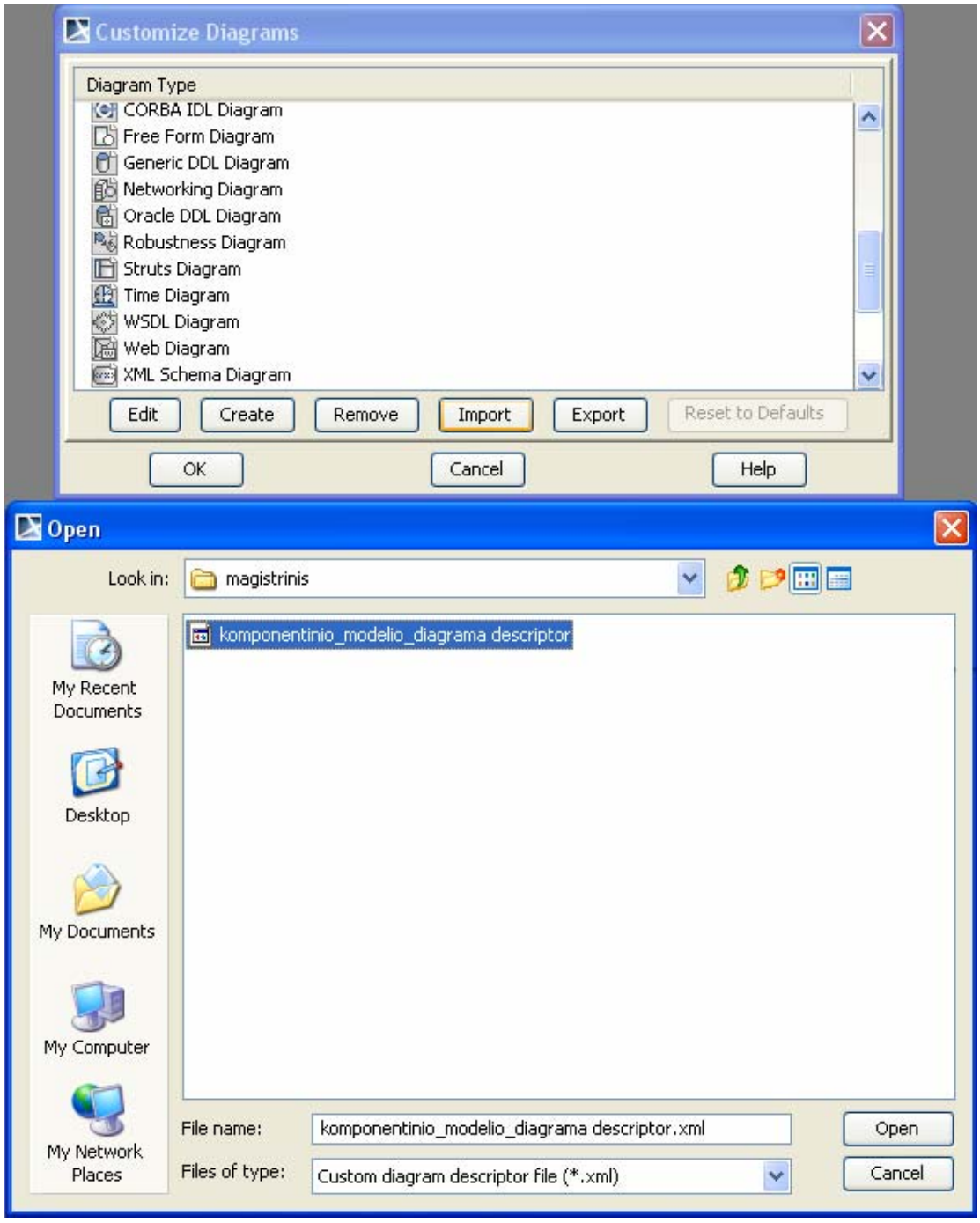
Iš vartotojo meniu keliami komponentai. Įvedamas komponento pavadinimas, atributai bei metodai. Komponentai tarpusavyje sujungiami „Association“ ryšiais (65 pav.) būtinai nurodant koks komponentas į kokį nukreipiamas.



65 pav. Komponentų įkėlimo iš vartotojo meniu fragmentas

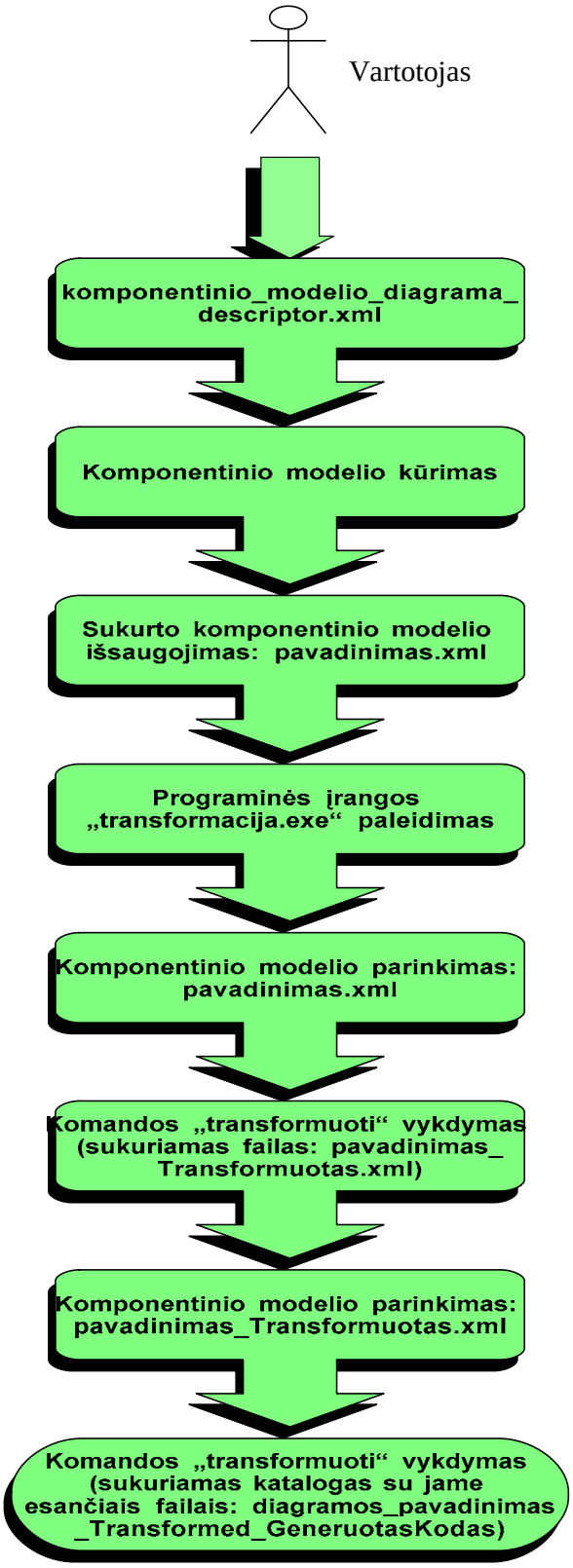
9.2. Priedas Nr.2 Komponentinio modelio diagramos importavimas

Prieš tai sukurtą komponentinio modelio diagramą galima eksportuoti „xml“ formatu. Gaunamas failas „komponentinio_modelio_diagrama descriptor.xml“. Taip pat ją galima importuoti naujoje darbo vietoje (66 paveikslėlis), kad nereikėtų kurti naujo, kaip parodyta pirmame priede.



66 pav. Komponentinio modelio diagramos importavimoas

67 paveiksle pateikiami pagrindiniai žingsniai, kuriuos turi atlikti vartotojas projekto eigoje.



67 pav. Projekto eigos žingsniai

9.3. Priedas Nr.3 Kompaktinis diskas (CD)