KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Valdemar Blaževič

UML CASE įrankio išplėtimas duomenų vientisumo reikalavimų kodo generavimui

Magistro darbas

Darbo vadovas
prof. Lina Nemuraitė

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Valdemar Blaževič

UML CASE įrankio išplėtimas duomenų vientisumo reikalavimų kodo generavimui

Magistro darbas

Recenzentas
doc. dr. V. Pilkauskas
2008-05-

Vadovas
prof. L. Nemuraitė

Atliko
IFM-2/2 gr. studentas
Valdemar Blaževič
2008-05-

Kaunas, 2008

TURINYS

1. Įvadas 5
2. Esamos duomenų modelių vientisumo reikalavimų užtikrinimo situacijos analizė 8
   2.1. Vientisumo reikalavimų tipai 8
   2.2. Egzistuojančių vientisumo reikalavimų katalogas 9
   2.3. Vientisumo reikalavimų įgyvendinimas egzistuojančiuose CASE įrankiuose 12
      2.3.1. PowerDesigner 13
      2.3.2. USE 14
      2.3.3. MagicDraw 15
   2.4. Tyrimo uždavinio formuluotė 16
   2.5. Analizės išvados 17
3. Vientisumo reikalavimų realizavimo CASE įrankiuose metodas 18
3. Vientisumo reikalavimų realizavimo CASE įrankiuose metodas 18
   3.1. Vientisumo reikalavimų metamodelis 18
   3.2. UML duomenų modeliavimo profilio išplėtimas 19
   3.3. Šablonų specifikavimo kalba 21
      3.3.1. „e“ konstantos 21
      3.3.2 “tag” žymėtos reikšmės 21
      3.3.3. “foreach” ciklas 22
      3.3.4. Komentarų naudojimas 24
   3.4. Kodo generavimo algoritmas 24
4. Metodo realizacija UML CASE įrankyje MagicDraw 27
   4.1. Sistemos reikalavimų specifikacija 27
      4.1.1. Panaudojimo atvejų modelis 27
      4.1.2. Panaudojimo atvejų specifikacijos 30
      4.1.3. Funkciniai reikalavimai 32
      4.1.4. Nefunkciniai reikalavimai 37
   4.2. Sistemos architektūra 38
      4.2.1. Architektūros tikslai ir reikalavimai 38
      4.2.2. Loginė architektūra 39
      4.2.3. Klasių modelis 39
      4.2.4. Sistemos elgsenos modelis 45
      4.2.5. Realizacijos modelis 49
5. Sistemos realizacija 50
6. Sistemos testavimas 53
   6.1. Testavimo tikslai ir objektai 53
   6.2. Testavimo apimtis 53
   6.3. Pagrindiniai reikalavimai 53
   6.4. Testuojama programinė įranga 54
   6.5. Sąsajos 54
   6.6. Testavimo strategija 55
   6.7. Testavimo rezultatų dokumentacija 55
   6.8. Testavimo rezultatai ir išvados 55
7. Eksperimentinis tyrimas 57
   7.1. Vientisumo reikalavimų kodo generavimo pavyzdys 57
      7.1.1. Išvestinio atributo vientisumo reikalavimas 58
      7.1.2. Išorinio unikalumo vientisumo reikalavimas 59
      7.1.3. Ekvivalentiškumo vientisumo reikalavimas 61

7.2. Sistemos išplėtimais naujais vientisumo reikalavimais 64
7.3. Sistemos išplėtimas naujais SQL dialektais....65
7.4. DB fizinio modelio kūrimo greičio įvertinimas naudojant sistemą....66
8. Sistemos įvertinimas ir taikymo perspektyvų apibendrinimas 67
8.1. Įvertinimas pagal realizuotų vientisumo reikalavimų skaičių 67
8.2. Nefuncinių reikalavimų tenkinimo įvertinimas 68
8.3. DB fizinio modelio kūrimo greičio įvertinimas naudojant sistemą....69
9. Išvados 71
10. Literatūra....72
11. Santrauka užsienio (anglų) kalba....74
12. Terminų ir santrumpų žodynas 75
13. Priedai 77
13 .1. Priedas Nr.1. Straipsnis „Duomenų vientisumo reikalavimų įgyvendinimas naudojant SQL kodo generavimo šablonus“ 77
13.2. Priedas Nr.2. Vientisumo reikalavimų šablonai 81
13.3. Priedas Nr.3.plugin.xml 88
13.4. Priedas Nr.4. ctransformer.properties....89
13.5. Priendas Nr. 5. Junit ir Cobertura testavimo ataskaitų protokolai 90

1. Įvadas

Vientisumo reikalavimai yra neatskiriama konceptualiojo modelio dalis, apimanti dalį probleminės srities semantikos. Atlikus dažniausiai naudojamų konceptualiojo modeliavimo metodų analizę, galima teigti, kad nei vienas iš jų neapima visos reikalavimų aibės, kurios reikia semantiškai prasmingam konceptualiajam modeliui. Vientisumo reikalavimams vaizduoti naudojamos UML 2.0 versijoje patobulintos išplėtimo galimybės – stereotipai ir žymėtosios reikšmės. Reikalavimų įvertinimas konceptualiajame modelyje leidžia pilnai pavaizduoti dalykinę sritį, verifikuoti modelį ankstyvoje kūrimo stadijoje ir naudoti ne tik duomenų bazių, bet ir kitų tipų schemų bei programų kodui generuoti.

Augantys informacijos kiekiai ir šių duomenų automatizavimo galimybės e-Verslo, Data Warehousing, Data Mining ir semantinių Web servisų srityje reikalauja aukštesnio abstrakcijos ir tikslumo pavaizduojant ir modeliuojant informacinius srautus. OMG apibrėžė verslo taisyklių semantiką, leidžiančią formaliai pavaizduoti verslo terminus (verslo žodyną ir taisykles) nepriklausomai nuo kalbos. Koncepcinis modelis su vientisumo reikalavimais leidžia pavaizduoti informacinės sistemos verslo apibrėžimus koncepciniame lygmenyje. Kokia nors programavimo kalba užrašomi vientisumo reikalavimai (trigeriai, procedūros) nėra priimtini daugumai organizacijų dėl per daug žemo abstrakcijos lygmens ir priklausomybės nuo IS techninių sprendimų.

Magistrinio darbo metu sukurtas produktas turėtų užpildyti šiuo metu egzistuojančius informacijos srautus modeliuojančių programinių paketų trūkumus. Vientisumo reikalavimų stereotipų aibės papildymas ir jų transformacijų iš loginio į fizinį modelį galimybės turėtų sudominti IS kūrėjus. Šio produkto pagalba siekiama supaprastinti komunikacijos galimybes tarp skirtingų sričių specialistų. IS projektavimą ir programavimą galima būtų visiškai iškelti į aukštesnį abstrakcijos lygmenį, suteikiant IS didesnį lankstumą prisitaikant prie sparčiai besikeičiančios verslo aplinkos. Tai turėtų tiesioginę įtaką IS kūrimo ir modifikavimo sąnaudoms.

Vientisumo reikalavimus UML modeliuose galima užrašyti OCL kalba, tačiau OCL yra per sudėtinga daugeliui projektuotojų. Be to, SQL kodo generavimas iš OCL kol kas nėra efektyvus [2]. Magistriniame darbe pateikiamas alternatyvus vientisumo reikalavimus

realizuojančio kodo generavimo metodas, paremtas UML išplėtimo mechanizmais ir SQL šablonais.

SQL kodui generuoti sukurta dinaminių SQL išraiškų užrašymo kalba, paremta XML. Ja galima aprašyti loginio duomenų modelio vientisumo reikalavimus ir transformuoti į programos kodą. Transformavimo algoritmas realizuotas kaip UML CASE įrankio MagicDraw išplėtimas (įskiepis, angl. *plug-in*) *Ctransformer*. Įskiepio sukūrimo teorinis pagrindas yra disertacija [12] bei idėjos, aprašomos [9, 10, 13] straipsniuose. Šio įrankio sukūrimas suteikia galimybes kurti programinę įrangą ir geresnės kokybės duomenų bazes [11] naudojant UML kalbą bei tą patį UML CASE įrankį.

Šio magistrinio darbo tikslas yra padidinti UML CASE įrankių duomenų bazių projektavimo galimybes, sukuriant vientisumo reikalavimų vaizdavimo ir kodo generavimo priemones.

Darbo uždaviniai:

- ✓ Sukurti metamodelį ir papildomus duomenų modelio stereotipus vientisumo reikalavimams vaizduoti;
- ✓ Sukurti SQL šablonų aprašymo priemones, leidžiančias universaliai aprašyti pagal stereotipus generuojamą SQL kodą;
- ✓ Sudaryti kodo generavimo algoritmus;
- ✓ Suprojektuoti pasirinkto MagicDraw įrankio išplėtimą;
- ✓ Realizuoti ir ištestuoti;
- ✓ Atlikti eksperimentinį sukurto produkto išbandymą bei įvertinimą;
- ✓ Vientisumo apribojimų panaudojimas modeliuojant ir kuriant duomenų bazės loginį ir fizinį modelius.

Šiame darbe sprendžiamos grafinio vientisumo reikalavimų vaizdavimo loginiuose duomenų modeliuose bei jų kodo realizavimo problemos. Sprendžiant šiuos uždavinius, buvo analizuojami vientisumo reikalavimų tipai, UML plėtimo mechanizmai bei esamos vientisumo reikalavimų įgyvendinimo CASE įrankiuose galimybės.

Darbo struktūra:

- ✓ Darbo analizės dalyje pateikta egzistuojančių vientisumo reikalavimų tipai, klasifikacija ir aprašymai. Apžvelgiami šiuo metu egzistuojantys CASE įrankiai. Detaliai aprašomi trys CASE įrankiai: Sybase PowerDesigner, USE, NoMagic MagicDraw funkcionalumo analizė ir išplėtimo glimybės. Suformuluojamas tyrimo uždavinys. Šios dalies pabaigoje pateiktas analizės rezultatų apibendrinimas ir išvados.

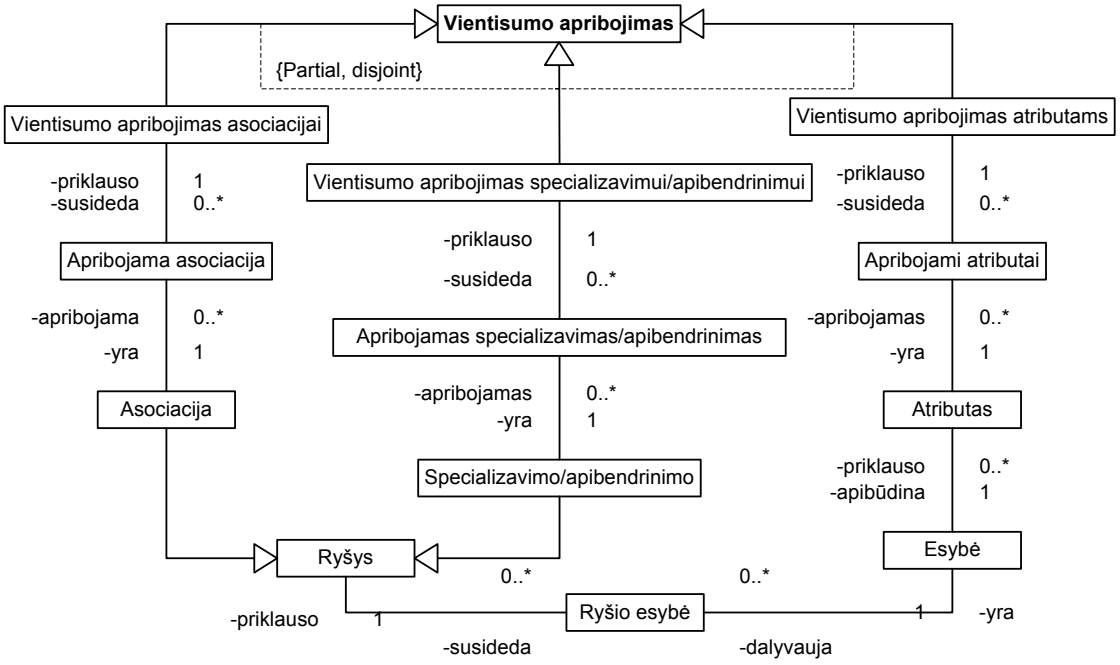
- ✓ Projektavimo dalyje aprašyti vientisumo reikalavimų atvaizdavimo ir transformavimo į programinį SQL kodą įrankio reikalavimai ir projektiniai sprendimai, kurie buvo pritaikyti įrankio kūrimo proceso metu. Taip pat šioje dalyje aprašomas sukurto vientisumo reikalavimų transformavimo įrankio architektūros modelis, integravimo su MagicDraw UML CASE įrankių sąsajos realizavimas.
- ✓ Eksperimentinėje dalyje aprašytas sukurto MagicDraw įskiepio naudingumo tyrimas, pateikti tyrimo rezultatai. Tyrimo tikslas - nustatyti automatinio vientisumo reikalavimų transformavimo į programinį SQL kodą pranašumus, prieš rankinį fizinio duomenų bazės modelio vientisumo reikalavimų kūrimo būdą. Pranašumas vertinamas nefunkcinių modelio kūrimui reikalingo laiko sąnaudų palyginimu. Taip pat nustatyti sukurto įskiepio realizacijos kokybę įvertinant nefunkcinių reikalavimų tenkinimo pilnumą ir realizuotų vientisumo reikalavimų skaičių lyginant su kitais CASE ir DBVS įrankiais.
- ✓ Pabaigoje pateikiamos atlikto darbo pagrindinės išvados ir rezultatai.
- ✓ Prieduose pateikiami vientisumo reikalavimų realizacijos dinaminių SQL šablonų kalba, sukurto įskiepio testavimo ataskaitos ir kita su darbu susijusi naudinga medžiaga.

2. Esamos duomenų modelių vientisumo reikalavimų užtikrinimo situacijos analizė

Šiame skyriuje pateikiama išsami vientisumo reikalavimų analizė. Pateikiami vientisumo reikalavimų klasifikacijos principai, apžvelgiami egzistuojantys vientisumo reikalavimai. Pateikiama šiuo metu egzistuojančių CASE įrankių apžvalga. Plačiau aprašomi MagicDraw, Sybase PowerDesigner, USE CASE įrankiai. Skyriaus pabaigoje pateikiamos analizės išvados.

2.1. Vientisumo reikalavimų tipai

“Konceptualiojo modeliavimo metodai (esybių-ryšių, išplėstasis esybių-ryšių, aukštesnės eilės esybių-ryšių, objektų rolių modelis ir unifikuota modeliavimo kalba) turi savų privalumų ir trūkumų, tačiau visi jie gali būti laikomi konceptualiojo modeliavimo „tėvo“ ER modelio išplėtimais.” [9]. Vientisumo reikalavimus ir ryšius tarp jų galima būtų pavaizduoti sekančia schema (1 pav.):

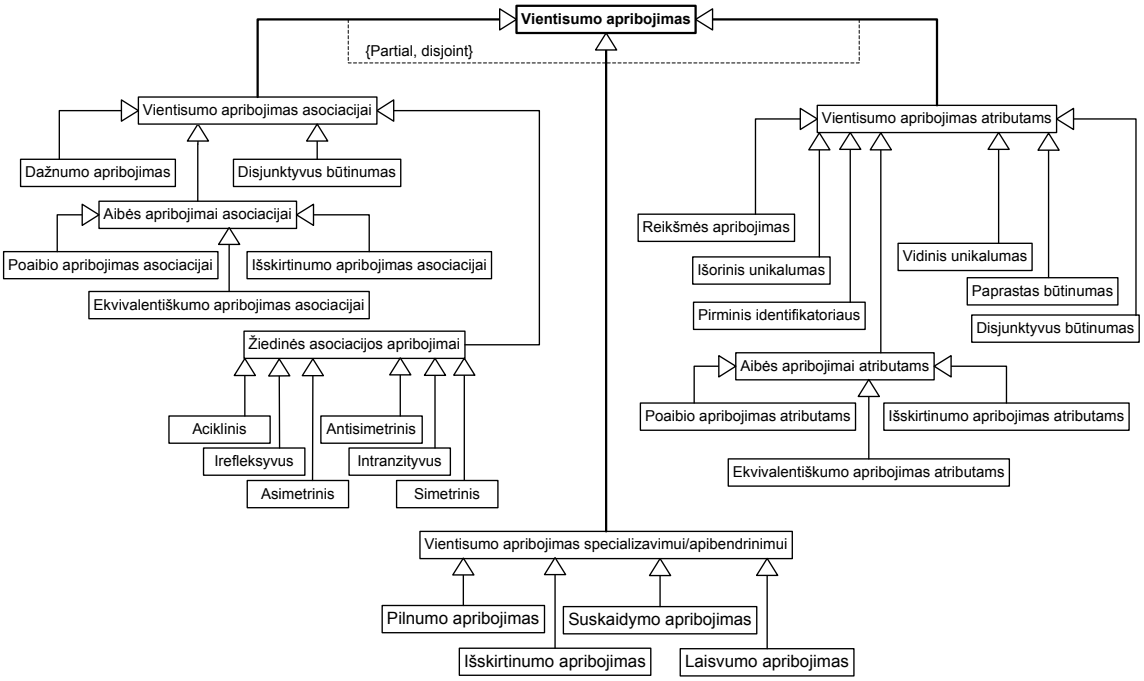


1 pav. *Konceptualiojo modelio pagrindiniai elementai ir ryšiai tarp jų [9].*

Vientisumo reikalavimus galima būtų suklasifikuoti, remiantis jų pritaikymu konceptualiojo modelio elementams:

- ✓ Vientisumo reikalavimai esybių atributams.
- ✓ Vientisumo reikalavimai asociacijų ryšiams tarp esybių.
- ✓ Vientisumo reikalavimai specializacijos/apibendrinimo ryšio tipui.

Remiantis šia klasifikacija galima išskirti sekančius dažniausiai naudojamus reikalavimų potipius (2 pav.):



2 pav. *Vientisumo reikalavimų klasifikacija [9].*

2.2. Egzistuojančių vientisumo reikalavimų katalogas

Vientisumo reikalavimų aprašymas pateikiamas 2.2 lentelėje:

2.2 lentelė*. Vientisumo tipų reikalavimų aprašymas.*

Pavadinimas	Aprašymas
Nulinės reikšmės (angl. *null*)	Atributo reikšmė yra neprivaloma.
Nenulinės reikšmės (angl. *not null*)	Atributo reikšmė yra privaloma. Atributui negali būti priskiriamos neapibrėžtos reikšmės.
Unikalumo	Atributo reikšmės turi būti unikalios lentelėje, t.y. ta pati reikšmė

(angl. *uniqueness*)	stulpelyje nesikartoja. Null reikšmė leistina.
Pirminio rakto (angl. *primary key*)	Atributo reikšmė yra unikali. Stulpelio reikšmė vienareikšmiškai identifikuoja lentelės įrašą (eilutę). Null reikšmė neleistina.
Išorinio rakto (angl. *foreign key*)	Reikalavimas ryšiui tarp dviejų lentelių. Suriša vienos lentelės atributą/-us su kitos lentelės atributu/-ais.
Tikrinimo (angl. *check*)	Apibrėžia loginę operaciją, kurią privalo tenkinti atributui suteikiama reikšmė. Pavyzdžiui: *amzius INTEGER check (amzius > 0)*
Disjunktyvaus būtinumo (angl. *disjunctive mandatory*)	Reikalauja, kad visose leistinose IS būsenose bent vienas iš esybės atributų, kuriems nurodytas šis apribojimas, turėtų reikšmę. Reikalavimas neuždraudžia visiems apribotiems atributams įgyti reikšmes [9].
Išorinio unikalumo (angl. *external uniqueness*)	Šis reikalavimas apibrėžiamas, kaip ir unikalumo apribojimas, tačiau jis iškelia unikalumo reikalavimą tarp dviejų esybių atributų reikšmių [9].
Išvestinio atributo (angl. *precalculated derived attribute*)	Reikalavimas specifikuoja taisyklę, pagal kurią apskaičiuojama atributo reikšmė. Atributo reikšmė apskaičiuojama prieš įterpimą.
Išvestinio atributo (angl. *postcalculated derived attribute*)	Reikalavimas specifikuoja taisyklę, pagal kurią apskaičiuojama atributo reikšmė. Atributo reikšmė apskaičiuojama po įterpimo, atnaujinimo ir ištrynimo operacijos.
Kardinalumo (angl. *cardinality higher and lower bound*)	Reikalavimas nusako apatines ir viršutines ryšyje galinčių dalyvauti egzempliorių aibių kardinalumo ribas [9].
Kardinalumo (angl. *cardinality higher bound*)	Reikalavimas nusako viršutines ryšyje galinčių dalyvauti egzempliorių aibių kardinalumo ribas.
Apibendrinimo (angl. *disjoint specialization*)	Reikalavimas naudojamas, jeigu potipių esybių egzemplioriai tarpusavyje nepersidengia, tačiau pilnai nepadengia supertipo esybės egzempliorių aibės [9].
Apibendrinimo (angl. *complete specification*)	Reikalavimas naudojamas, jeigu potipių esybių egzemplioriai tarpusavyje persidengia ir pilnai padengia supertipo esybės egzempliorių aibę [9].
Apibendrinimo (angl. *combined specification*)	Reikalavimas naudojamas, jeigu potipių esybių egzemplioriai tarpusavyje nepersidengia ir pilnai padengia supertipo esybės egzempliorių aibės [9].
Nerefleksyvaus ryšio (angl. *irreflexive association*)	Reikalauja, kad tas pats esybės egzempliorius nedalyvautų abiejose ryšio rolėse tuo pačiu metu, t.y., esybės tipo egzempliorius A negali turėti ryšio pats su savim [9].
Asimetrinio ryšio (angl. *asymmetric association*)	Apriboja priešingos rolės egzistavimą tarp dviejų skirtingų esybės egzempliorių. Jei esybės egzempliorius A turi ryšį su tos pačios esybės egzemplioriumi B, egzempliorius B negali turėti ryšio su

	tokia pat role su tuo pačiu esybės A egzemplioriumi (jei A yra tėvas, o B sūnus, tai A negali būti B sūnus) [9].
Nesimetrinio ryšio (angl. *antisymmetric association*)	Neleidžia egzistuoti priešingai rolei tarp skirtingų esybės egzempliorių, tačiau priešingai negu „Nerefleksyvus“ apribojimas leidžia, kad ta pati esybė dalyvautų abiejose ryšio rolėse. Šis apribojimas leidžia esybės egzemplioriui A turėti ryšį su tos pačios esybės A egzemplioriumi ir leidžia egzemplioriui A turėti ryšį su egzemplioriumi B, tačiau draudžia egzemplioriui B turėti ryšį su egzemplioriumi A. [9].
Aciklinio ryšio (angl. *acyclic association*)	nurodo, kad negali egzistuoti ciklų tarp esybės egzempliorių, susietų refleksyviuoju ryšiu. Jeigu esybės egzempliorius A turi ryšį su tos pačios esybės B egzemplioriumi ir egzempliorius B turi ryšį su tos pačios esybės C egzemplioriumi, egzemplioriaus C ryšys su A egzemplioriumi yra negalimas. [9].
Netranzityvaus (angl. *intransitive association*)	užtikrina, kad egzistuoja tik tiesioginis „tėvo“ – „vaiko“ ryšys. Jeigu esybės egzempliorius A turi ryšį su tos pačios esybės B egzemplioriumi ir egzempliorius B turi ryšį su tos pačios esybės C egzemplioriumi, egzemplioriaus A ryšys su C egzemplioriumi yra negalimas [9].
Ekvivalentiškumo (angl. *equal set*)	„Ekvivalentiškumas“ pasireiškia tuo, kad egzempliorių aibės, turinčios vieno atributo ar ryšio (ar jų grupės) reikšmes, sutampa su aibėmis, turinčiomis kito atributo ar ryšio (ar jų grupės) reikšmes [9].
Poaibio (angl. *sub set*)	Šis apribojimo tipas rodo, kad ne visi esybės egzemplioriai gali dalyvauti ryšyje, o tik tie, kuriuos atrenka nurodytas ryšių kelias Poaibio apribojimas sako, kad vieno ryšių kelio atrinkta egzempliorių aibė yra kito ryšių kelio atrinktų egzempliorių poaibis [9].

2.3. Vientisumo reikalavimų įgyvendinimas egzistuojančiuose CASE įrankiuose

Šiuo metu rinkoje yra sekantys komerciniai ir nekomerciniai UML, OCL ir duomenų bazių projektavimo programiniai įrankiai (2.3. lentelė) [1, 3, 5, 6, 8, 14]:

***2.3 lentelė**. Egzistuojančių CASE įrankių apžvalga.*

Pavadinimas	**Gamintojas**	**Kaina**	**Funkcionalumas**			
			UML[1]	DBP[2]	VASUML[3]	VASPP[4]
PowerDesigner	Sybase *www.sybase.com*	2500 EUR	Taip	Taip	Ne	Taip
Oracle Designer	Oracle *www.oracle.com*	Nėra[5]	Ne	Taip	Ne	Taip
Enterprise Architect	Sparx Systems *www.sparxsystems.com*	199 USD	Taip	Ne	Ne	Ne
myEclipse	Genuitec LLC *www.myeclipseide.com*	30 USD per metus	Taip	Ne	Ne	Ne
MagicDraw	No Magic Inc. *www.nomagic.com*	1355 EUR	Taip	Taip	Ne	Taip
Eclipse UML2 Plug-In	www.eclipse.org	Nemokamai. Žiūrėti CPL licenziją.	Taip	Ne	Ne	Ne
ArgoUML	http://argouml.tigris.org/	Nemokamai. Žiūrėti BSD licenziją .	Taip	Ne	Ne	Ne
USE	Bremeno universitetas *http://www.db.informatik.uni-bremen.de/*	Nemokamai	Taip	Ne	Taip	Ne

[1] - UML ir OCL diagramos.

[2] - duomenų bazių projektavimo grafinės ir programinės priemonės.

[3] - vientisumo reikalavimų projektavimas/apibrėžimas UML arba OCL notacija.

[4] - vientisumo reikalavimų programavimo priemonės.

[5] – ši informacija nėra pateikta Oracle žinialapyje.

Iš aukščiau pateiktų duomenų matyti, kad šiuo metu rinkoje egzistuoja nemažai programinių sprendimų. Šie sprendimai skiriasi kaina ir funkcionalumu. Žemiau bus išnagrinėti trys programiniai sprendimai, labiausiai atitinkantys funkcinius ir techninius projekto reikalavimus.

2.3.1. PowerDesigner

PowerDesigner yra Sybase firmos produktas skirtas duomenų bazių analizei ir verslo modelių projektavimui, panaudojant UML objektų modelius. Remiantis Gartner Dataquest pateiktais duomenimis [7], Sybase PowerDesigner užima pirmaujančias pozicijas duomenų bazių programinės įrangos kūrimo rinkoje, remiantis 2004 metų duomenimis apie naujų licenzijų pardavimo pelną, siekė 33.7 procentus rinkos dalies. Remiantis Sybase žinialapyje pateikta informacija apie šį produktą [14], galima būtų išskirti sekančias svarbiausias savybes:
Bendros savybės:

- ✓ Reikalavimų analizės priemonės.
- ✓ Veikimo/Įtakos analizės priemonės, prieš įdiegiant DB pakeitimus produkcinėje aplinkoje.
- ✓ Dokumentacijos generatorius.
- ✓ Patogi vartotojo sąsaja.

Modeliavimo technologijos:

- ✓ Verslo procesų modeliavimo įrankiai.
- ✓ Koncepcinio, loginio ir fizinis duomenų modeliavimo įrankiai.
- ✓ Objektų modeliavimas UML 1.x ir 2.0 priemonėm.
- ✓ XML modeliavimo priemonės.

Palaikomos platformas:

- ✓ Procesų vykdymo – ebXML, BPEL4WS
- ✓ RDBVS – arti 60 RDBVS, įskaitant Oracle, IBM DB/2, Microsoft SQL Server, Sybase, MySQL, NCR Teradata ir daug kitų.
- ✓ Objektinės programavimo kalbos - tokios kaip Java J2EE, C#, VB.NET, PowerBuilder, XML, C++, Web Servisai ir kitos.

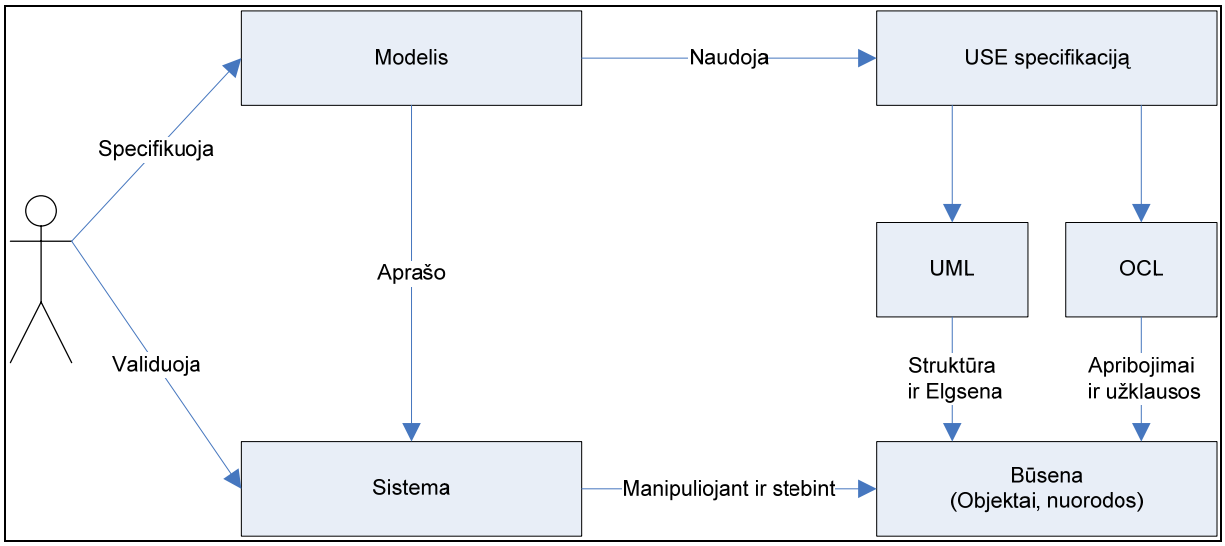
- ✓ Programinės įrangos kūrimo platformos - Eclipse, PowerBuilder ir Visual Studio.

Trūkumai:

- ✓ Didelė produkto kaina.
- ✓ Mažas paplitimas Lietuvoje lyginant su konkurentais.
- ✓ Šiame produkte nėra realizuotos grafinės vientisumo apribojimo projektavimo galimybės naudojant UML priemones. Duomenų bazės modelis yra išplečiamas vientisumo reikalavimais rankinių būdų, t.y. rankinių būdų papildant sugeneruotus DB skriptus.

2.3.2. USE

USE yra informacinių sistemų specifikavimo sistema. Šis programinis produktas buvo sukurtas ir toliau vystomas Bremeno universiteto Informatikos fakultete [3]. Produktas paremtas UML kalba. USE specifikacija turi savyje tekstinį modelio aprašą panaudojant UML klasių diagramų savybes (klasės, asociacijos ir t.t.). Modelio integralumo reikalavimai yra papildomai užrašomi naudojant OCL kalbą. Sistema leidžia atlikti projektuojamos sistemos simuliacijas neformalių reikalavimų specifikacijos patikrinimui. Simuliacijų metu gali būti kuriamos ir manipuliuojamos atskiros sistemos būsenos. Kiekvienai sistemos būsenai automatiškai yra tikrinami OCL reikalavimai. Detali informacija apie sistemos būseną yra pateikiama grafiniu ir tekstiniu būdais. Sekantis modelis pavaizduoja bendrą USE metodiką (3 pav.):



***3 pav**. USE metodikos modelis [3]*

Sistemos privalumai:

- ✓ Modelio vientisumo reikalavimų aprašymo galimybė naudojant OCL kalbą.
- ✓ Sistemos būsenų validavimo ir simuliavimo galimybė.
- ✓ USE sistemos programinis kodas yra laisvai prieinamas.
- ✓ Programa yra nemokama.

Sistemos trūkumai:

- ✓ Sistema nėra pritaikyta duomenų bazių projektavimui.
- ✓ Sistemoje nenumatytos integracijos su DBVS priemonės.
- ✓ Sistemoje nenumatytos modelio transformacijos į programinį kodą galimybės.
- ✓ Sistema nepalaiko modelio išplėtimo galimybės programiniu kodu.

2.3.3. MagicDraw

MagicDraw yra Nomagic firmos vizualus UML ir CASE modeliavimo įrankis pritaikytas komandiniam darbui. Produktas yra skirtas verslo, programinės įrangos analitikams, programuotojams, kokybės užtikrinimo inžinieriams, ir dokumentacijos rašytojams. Produktas yra pritaikytas objektiškai orientuotų sistemų ir duomenų bazių analizei ir dizainui.

Bendros MagicDraw savybės [6]:

- ✓ Patogi ir lanksti vartotojo sąsaja.
- ✓ Automatizuotos UML semantikos klaidų tikrinimo ir paieškos priemonės.
- ✓ UML modelių generavimas iš programinio kodo ir į programinį kodą parašytą sekančiomis programinio kalbomis:. Java, C#, C++, CORBA IDL, EJB 2.0, DDL, CIL (MSIL), WSDL, ir XML schema.
- ✓ MagicDraw Teamwork serveris leidžia keliems vartotojams paraleliai dirbti su tuo pačiu modeliu.
- ✓ MagicDraw funkcionalumas gali būti integruojamas su sekančiomis programavimo platformomis: Eclipse, IBM WSAD ir RAD, Borland JBuilder, IntelliJ IDEA, NetBeans, Sun Java Studio.
- ✓ Automatizuotas modelių dokumentacijos, ataskaitų kūrimas tokiais formatais: HTML, PDF, ir RTF.
- ✓ Modeliai gali būti transformuojami iš nepriklausomų nuo platformos (*angl*. PIM- Platform Independent Model) modelių į platformai specifinius modelius (*angl*. PSM- Platform Specific Model).

- ✓ MagicDraw yra pritaikytas naudojimui daugelyje operacinių sistemų, tokiu kaip Windows 98/ME/NT/2000/XP, Solaris, OS/2, Linux, HP-UX, AIX, MacOS (X) ir kitose, kur palaikomas Java 1.4 arba 1.5 versijos.
- ✓ Suteikia UML standarto išplėtimo galimybes be papildomo programinio kodo generavimo.
- ✓ Produkto išplėtimo galimybės panaudojant MagicDraw OpenApi biblioteką.

Sistemos trūkumai:

- ✓ Ribotos duomenų bazių projektavimo ir analizės galimybės/priemonės.
- ✓ Didelė produkto kaina.

2.4. Tyrimo uždavinio formuluotė

Šiuo metu egzistuojantys UML CASE įrankiai gali sugeneruoti SQL kodą, atsižvelgdami tik į pagrindinius vientisumo reikalavimus, realizuojamus DBVS priemonėmis: pirminius ir išorinius raktus bei unikalius indeksus. Praktikoje vientisumo reikalavimų yra daug, ir ne pagrindinių reikalavimų išpildymas reikalauja didelių pastangų [9, 10]. Dažniausiai šie reikalavimai įgyvendinami fizinės realizacijos metu, o tai prieštarauja modeliais grindžiamos architektūros MDA (*angl*. Model Driven Architecture) principams, pagal kuriuos pagrindinis kūrimas vyksta aukštesnės abstrakcijos (t.y. modelių) lygyje. Tai leidžia padidinti kūrimo efektyvumą (padidinti greitį, pakartotinį naudojimą, plečiamumą) ir išvengti daugelio klaidų, kadangi visi formalizuojami darbai perduodami kompiuteriui.

Įvertinus jau esamą produktų funkcionalumą, išplėtimo galimybes ir pritaikymo galimybes, buvo pasirinktas firmos NoMagic MagicDraw programinis paketas. Vieni iš svarbiausių programinio paketo pranašumų lyginant su kitais produktais:

- ✓ Į MagicDraw programinį paketą įeina JAVA klasių biblioteka, pavadinimu OpenApi, kuri žymiai palengvina produkto išplėtimo galimybes.
- ✓ Produktas parašytas JAVA programavimo kalba, kas leidžia produktą naudoti skirtingose operacinėse sistemose, tokiose kaip Windows ir Linux.
- ✓ KTU yra jau įsigiję MagicDraw licenziją, kas žymiai sumažino projekto sąnaudas.

Sujungiant:

- ✓ UML 2 išplėtimo galimybes: profilius, stereotipus ir žymėtąsias reikšmes;
- ✓ Egzistuojančio CASE įrankio galimybes;
- ✓ SQL šablonus

galima sukurti priemonę, kuri galėtų išspręsti vientisumo reikalavimų atvaizdavimo,duomenų bazių modelio nepilnumo ir generuojamo kodo efektyvumo problemas. Toks sprendimas padidintų esamų UML CASE įrankių efektyvumą, programinės įrangos kūrimo greitį ir kokybę, griežčiau atskirtų sistemos kūrėjų užduotis.

Taigi šio **darbo tikslas** yra padidinti UML CASE įrankių duomenų bazių projektavimo galimybes, sukuriant vientisumo reikalavimų vaizdavimo ir kodo generavimo priemones.

Darbo uždaviniai:

√ Sukurti metamodelį ir papildomus duomenų modelio stereotipus vientisumo reikalavimams vaizduoti;

√ Sukurti SQL šablonų aprašymo priemones, leidžiančias universaliai aprašyti pagal stereotipus generuojamą SQL kodą;

√ Sudaryti kodo generavimo algoritmus;

√ Suprojektuoti pasirinkto MagicDraw įrankio išplėtimą;

√ Realizuoti ir ištestuoti;

√ Atlikti eksperimentinį sukurto produkto išbandymą bei įvertinimą.

2.5. Analizės išvados

1. Šiuo metu egzistuojantys UML CASE įrankiai gali sugeneruoti SQL kodą, atsižvelgdami tik į pagrindinius vientisumo reikalavimus, realizuojamus DBVS priemonėmis: pirminius ir išorinius raktus bei unikalius indeksus.

2. Vientisumo reikalavimų analizė rodo, kad galima išskirti dar 18 apribojimų tipų, kurie dažnai sutinkami projektuojant duomenų bazes ir kuriais tikslinga papildyti duomenų loginius modelius.

3. UML išplėtimo mechanizmai ir CASE įrankių realizavimo principai leidžia plėsti esamus UML modelius bei CASE įrankių galimybes.

4. Tolesniam tyrime nustatyta vientisumo apribojimų aibė bus realizuojama pasirinktu CASE įrankiu MagicDraw, taikant UML plėtimo mechanizmus ir papildant jau esamą CASE įrankio funkcionalumą.

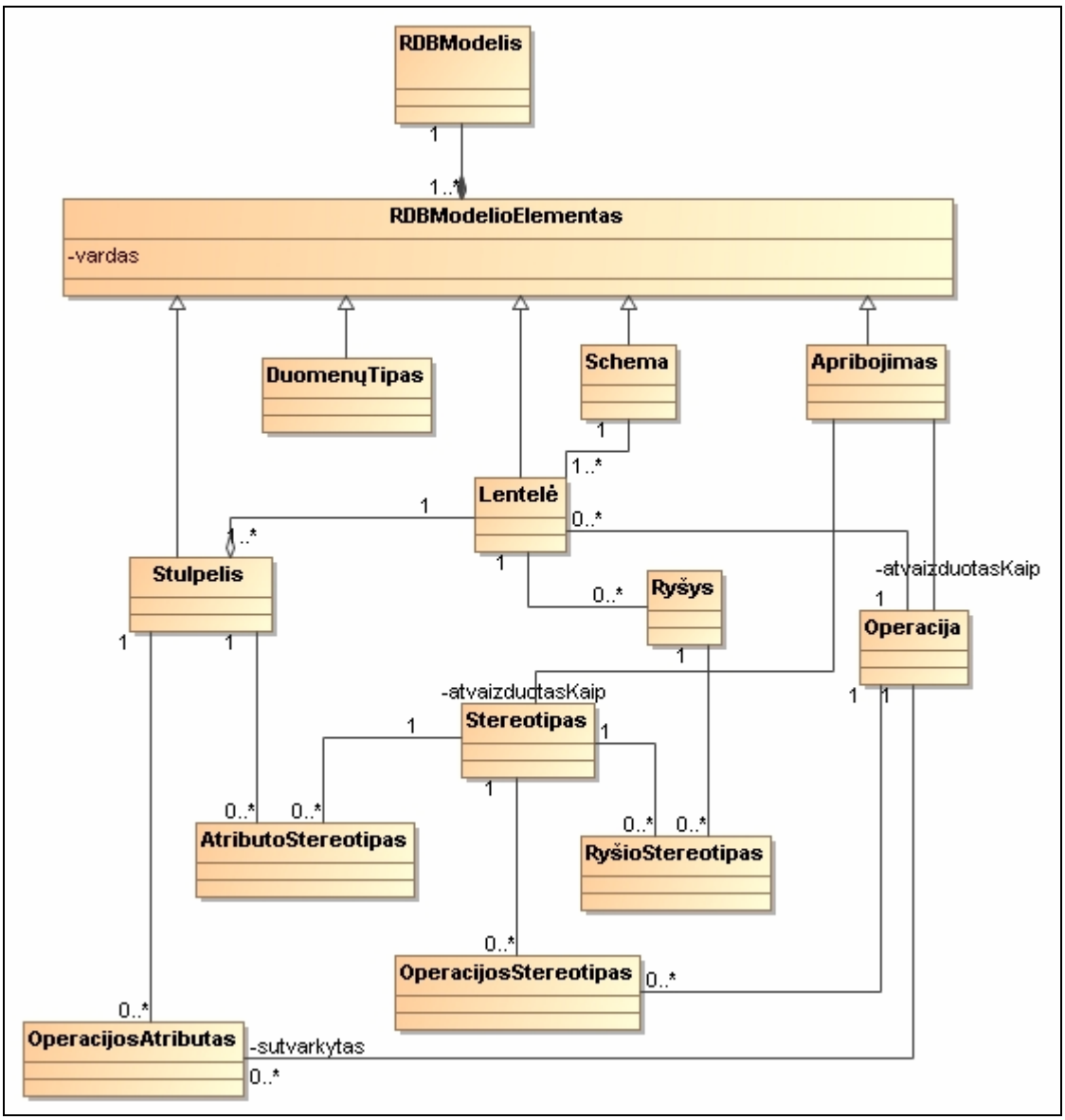
3. Vientisumo reikalavimų realizavimo CASE įrankiuose metodas

Šiame skyriuje aprašomas sudarytas vientisumo reikalavimų realizavimo metodas, kuris apima:

- vientisumo apribojimų stereotipus, kuriais apribojimai vaizduojami duomenų modelyje;
- vientisumo apribojimų metamodelį, kuris naudojamas realizuojant vientisumo apribojimų kodo generavimą;
- SQL šablonų kalbą, kuri leidžia užrašyti SQL sakinius vientisumo apribojimų stereotipams ir taikyti juos bet kokiai dalykinei sričiai;
- Vientisumo apribojimų kodo generavimo algoritmą.

3.1. Vientisumo reikalavimų metamodelis

Duomenų bazės metamodelį sudaro modelio elementai, kurių kiekvienas priklauso duomenų bazės schemai. Schema apima lenteles, kurias sudaro atributai ir operacijos. Lentelės tarpusavyje yra surištos ryšiais. Vientisumo reikalavimus reliaciniame duomenų bazės modelyje galima atvaizduoti kaip stereotipą lentelės stulpeliui, stulpelių grupei, ryšiui tarp lentelių. Kai tas pats vientisumo reikalavimas galioja daugiau nei vienam lentelės stulpeliui t.y. stulpelių grupei, tai atvaizduojama kaip stereotipas lentelės operacijai. Operacijos parametrais tampa lentelės stulpeliai, kuriems galioja duotas vientisumo reikalavimas. Metamodelis pavaizduotas 4 paveiksle:



4 pav. *Duomenų bazės loginio modelio UML metamodelis*

Remiantis šiuo metamodeliu galima sukurti profilį, kuriame kiekvienas vientisumo reikalavimas būtų realizuotas kaip stereotipas su žymėtom reikšmėmis. Kiekvienam stereotipui galima specifikuoti apribojimą, kuris užrašytų šio reikalavimo realizavimą reikiama kalba.

3.2. UML duomenų modeliavimo profilio išplėtimas

Pasinaudojant UML 2.0 išplėtimo galimybėmis, tokiomis kaip profiliai, stereotipai ir žymėtos reikšmės bei remiantis sudarytu metamodeliu, buvo sukurtas profilis, kuriame kiekvienas vientisumo reikalavimas realizuotas kaip stereotipas su žymėtosiomis reikšmėmis. Stereotipai sukurti vientisumo reikalavimams, kurių sąrašas buvo sudarytas remiantis atliktais

tyrimais vientisumo reikalavimų analizės srityje [2, 9, 10, 11, 12, 13]. Šie nauji stereotipai papildo MagicDraw programiniame pakete esančius stereotipus.

***3.1 lentelė**. Vientisumo apribojimų stereotipai.*

Apribojimo pavadinimas	**Stereotipo žymėjimas**	**Tipas**		
		Atributui	**Atributų grupei**	**Ryšiui**
Disjunktyvas būtinumo	DisjunctiveMandatory	Taip	Taip	Ne
Išorinio unikalumo	ExternalUniqueness	Ne	Ne	Taip
Išvestinio atributo[1]	PreCalcDerivedAttribute	Taip	Ne	Ne
Išvestinio atributo[2]	PostCalcDerivedAttribute	Taip	Taip	Ne
Kardinalumo[3]	CardinalityHLB	Ne	Ne	Taip
Kardinalumo[4]	CardinalityHB	Ne	Ne	Taip
Apibendrinimo[5]	DisjoinedSpecification	Ne	Ne	Taip
Apibendrinimo[6]	CompleteSpecification	Ne	Ne	Taip
Apibendrinimo[7]	CombinedSpecification	Ne	Ne	Taip
Nerefleksyvaus ryšio	IrreflexiveAssociation	Ne	Ne	Taip
Asimetrinio ryšio	AsymmetricAssociation	Ne	Ne	Taip
Nesimetrinio ryšio	AntisymmetricAssociation	Ne	Ne	Taip
Aciklinio ryšio	AcyclicAssociation	Ne	Ne	Taip
Netranzityvaus	IntransitiveAssociation	Ne	Ne	Taip
Ekvivalentiškumo	EqualSet	Ne	Taip	Ne
Poaibio	SubSet	Ne	Taip	Ne

1- Atributo reikšmė skaičiuojama PRIEŠ įterpimą.

2 – Atributo reikšmė skaičiuojama PO įterpimo.

3- Apribojimas viršutiniam ir apatiniam rėžiams.

4 - Apribojimas viršutiniam rėžiui.

5- Išskirstymo specifikacija.

6 - Pilno apibendrinimo specifikacija.

7 – Kombinuotas išskirstymo ir pilno apibendrinimo apribojimas.

Visi šie vientisumo reikalavimai buvo realizuoti viename profilyje stereotipų pavidalu su žymėtomis reikmėmis. Panaudojant dinaminių SQL užrašymo kalbą buvo užrašyti SQL šablonai, kurie naudojami vientisumo reikalavimų transformacijai į programinį kodą. Eksperimentui buvo pasirinktas firmos NoMagic CASE įrankis MagicDraw. Panaudojus

OpenAPI, t.y. Java klasių biblioteką produkto išplėtimui, buvo sukurtas įskiepis, kuris leidžia loginį duomenų bazės modelį išplėsti vientisumo reikalavimais ir transformuoti juos į programinį kodą.

Kiekvienam stereotipui specifikuotas apribojimas, kuris užrašo vientisumo reikalavimo realizavimą standartine SQL kalba.. Apribojimų skaičius stereotipui yra neribotas, todėl kiekvienas apribojimas gali saugoti SQL šabloną skirtingiems SQL dialektams. Toliau magistriniame darbe pateikta dinaminių SQL šablonų specifikavimo kalba paremta XML žymėtų reikšmių užrašymo taisyklėmis. Panaudojant SQL šablonų kalbą, galima aprašyti vientisumo reikalavimų realizacijas pasirinktam SQL dialektui, tuo užtikrinant generuojamo SQL kodo panaudojamumą skirtingoms reliacinių duomenų bazių valdymo sistemoms.

3.3. Šablonų specifikavimo kalba

Dinaminių SQL išraiškų užrašymui buvo sukurta šablonų kalba, paremta XML žymėtų reikšmių užrašymo taisyklėm. Šiame skyriuje bus pateiktos šios kalbos sintaksės taisyklės ir panaudojimo pavyzdžiai.

3.3.1. „e“ konstantos

<e:table_1/> - įrašomas lentelės/vaizdinio pavadinimas, kuriam priskirtas duotas apribojimo stereotipas.
<e:table_2/> - įrašomas lentelės/vaizdinio pavadinimas, su kuriuo sujungtas <e:table_1/> apribojimu.
<e:constraint_owner/> - įrašomas stulpelio/metodo/lentelės/vaizdinio pavadinimas, kuriam priskirtas apribojimo stereotipas.
e:element - naudojamas cikle „foreach“ iteracijos elementui pažymėti.

3.3.2 “tag” žymėtos reikšmės

tag:“žymėtos reikšmės vardas“ - įrašoma stereotipo žymėtos reikšmės specifikuota reikšmė transformacijos metu. Pavyzdžiui, jei stereotipas turi specifikuotą *tag1* žymėtą reikšmę, norint užrašyti jos reikšmę SQLe, tai užrašoma taip:

```
<tag:tag1/>
```

3.3.3. "foreach" ciklas

Ciklo užrašymo šablonas atrodo taip:

```
<foreach:[constraint_arg|column|value] [name="[table_1|table_2]|[tag'o pavadinimas]"]
        [separator="skirtukas"],
[separated="[skirtukas, kurio išskirtos tago reikšmių sąrašas]"]
[including_self="true|false" (default=true)]>
        e:element
</foreach:[constraint_arg|column|value]>
```

Sutrumpinta ciklo užrašymo forma atrodo taip:

```
<foreach:[constraint_arg|column|value] [name="[table_1|table_2]|[tag'o pavadinimas]"]
[separator="skirtukas"] [separated="[skirtukas, kurio išskirtos tag'o reikšmių sąrašas]"]
[including_self="true|false"(default="true")]/>
```

Ciklų specifikacijai galioja tokie apribojimai:

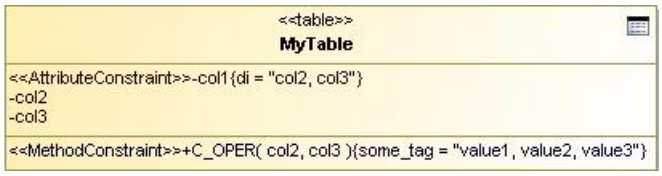
- ✓ Nepalaikomas ciklas cikle
- ✓ including_self naudojamas/įvertinamas tik tada, kai stereotipas priskirtas atributui. Visais kitais atvejais ignoruojamas.

Ciklo panaudojimą galima prailiustruoti sekančiu pavyzdžiu:

Turime lentelę 'MyTable', kurioje specifikuoti sekantys stulpeliai:

col1, col2, col3

Papildomai specifikuota operacija 'C_OPER(col2, col3)'. Šiai operacija priskiriamas stereotipas <<MethodConstraint>>. << MethodConstraint>> specifikuotas 'some_tag' žymėta reikšmė. 'some_tag' suteikiama reikšmė 'value1, value2, value3'. Papildomai specifikuojamas <<AttributeConstraint>> apribojimas stulpeliui col1, su reikšme ‚col2, col3':



1. Norint atspausdinti MyTable stulpelių pavadinimus, juos atskiriant kabliataškiu, SQL šablonas užrašomas taip:

```
<foreach:column name="table_1" separator=";"">
  e:element
</foreach:column>
```

arba

```
<foreach:column name="table_1" separator=";"/>
```

rezultate gautume:

col1;col2;col3

Pastaba: *name* atributas, pagal nutylėjimą inicializuojamas "*table_1*" reikšme. Todėl užtenka užrašyti *<foreach:column, separator=";"/>*

2. Norint atspausdinti C_OPER operacijos argumentus, juos atskiriant kableliu, SQL šablonas užrašomas taip::

```
<foreach:constraint_arg separator=",">
        e:element
</foreach:constraint_arg>
```

arba

<foreach:constraint_arg separator=","/>

rezultate gautume:

col2,col3

Pastaba 1: Jei stereotipas priskirtas atributui, išvedamas atributo pavadinimas.
Pastaba 2: Galima naudoti *name* atributą, tokiu atveju, jei stereotipo žymėta reikšmė su tokiu pavadinimu yra inicializuota, ciklas tampa ekvivalentišku užrašui:

```
<foreach:value name="[tag name]" separator="[separator]">
        e:element
</foreach:value>
```

Metodo argumentai yra ignoruojami. Jei šis stereotipas priskiriamas atributui, tada ignoruojamos žymėtos reikšmės ir galioja pirma pastaba ‚Pastaba 1'.

3. Norint atspausdinti ***some_tag*** reikšmių sąrašą ir jas atskirti ' OR' žodžiu:

```
<foreach:value name="some_tag" separator=" OR" separated=",">
        e:element
</foreach:value>
```

arba

<foreach:value name="some_tag" separator=" OR" separated=","/>

rezultate gautume:

value1 OR
value2 OR
value3

4. Užrašius:

<foreach:value name="some_tag" separator=" or" separated=",">
e:element IS NOT NULL
</foreach:value>

rezultate gautume:

value1 IS NOT NULL or
value2 IS NOT NULL or
value3 IS NOT NULL

5. Norėdami užrašyti col1 <<AttributeConstraint>> apribojimo *di* žymėtos reikšmės turinį:

5.1. be col1, tada turime naudoti *including_self='false*' atributą:

<foreach:value name="di" separator=" or" including_self="false">
e:element IS NOT NULL
</foreach:value>

rezultate gautume:

col2 IS NOT NULL or
col3 IS NOT NULL

5.2. su col1,

<foreach:value name="di" separator=" or">
e:element IS NOT NULL
</foreach:value>

rezultate gautume:

col1 IS NOT NULL or
col2 IS NOT NULL or
col3 IS NOT NULL

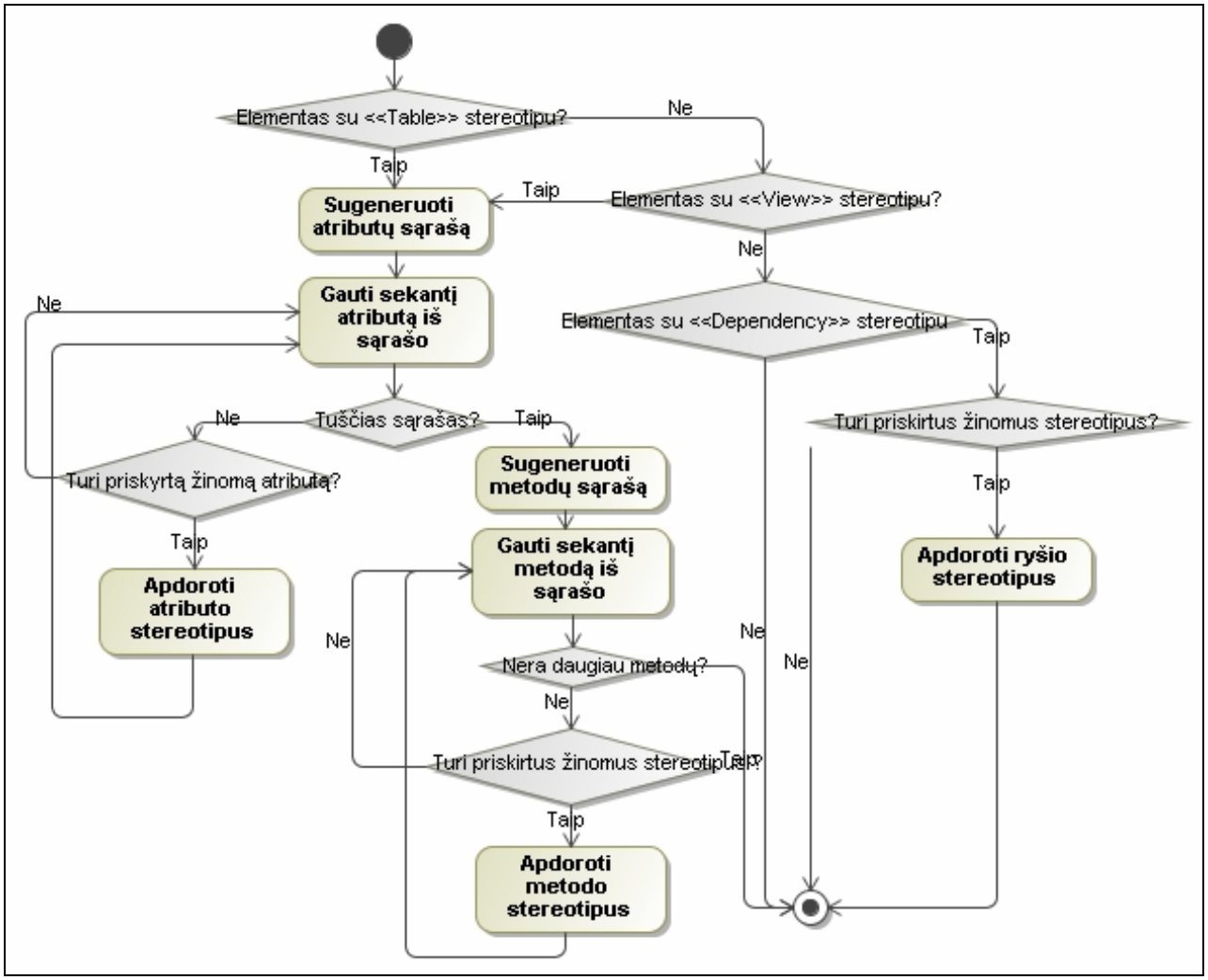
3.3.4. Komentarų naudojimas

Šablonų specifikavimo kalboje numatytos komentarų rašymo galimybės. Visos eilutės komentaro pradžios simbolis yra '#'. Toks komentaras bus išvedamas SQL skripte transformavus duomenų bazės loginį modelį. Visos ir dalies eilutės komentaro pradžios simbolis yra '//'. Toks komentaras neišvedamas SQL skripte transformavus loginį duomenų bazės modelį.

3.4. Kodo generavimo algoritmas

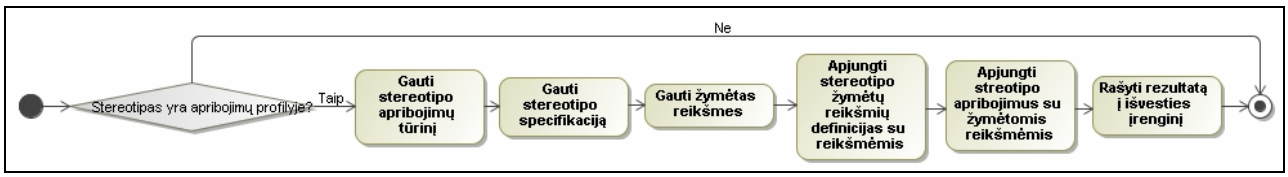
MagicDraw programiniame pakete duomenų bazės loginio modelio atvaizdavimui naudojama UML klasių diagrama. Duomenų modelio klasių (DDL) diagramą sudaro

elementai, kurie pagal priskirtus stereotipus skirstomi į lenteles (klasės su *<<Table>>* stereotipu), vaizdinius (klasės su *<<View>>* stereotipu), ir ryšius – (klasių priklausomybes su *<<Dependency>>* stereotipu). Naudojant *MagicDraw* išplėtimo Java klasių biblioteką *OpenAPI*, galima nuskaityti norimą duomenų modelį vaizduojančių elementų sąrašą ir kiekvieną iš jų apdoroti pagal 5 paveiksle pateikiamą algoritmą.



5 pav. Duomenų modelio diagramos vaizdavimo elemento apdorojimas[4]

Duomenų bazės modelio transformavimui patogiausia naudoti lankytojo (*angl*. Visitor) programavimo šabloną. Tokiu būdų užtikrinamas visų diagramos elementų dinaminis apdorojimas. Transformuojant duomenų modelio elementus, iš vientisumo reikalavimų profilio nuskaitomos apdorojamo stereotipo žymėtų reikšmių specifikacijos (vardai, tipai, pradinės reikšmės) ir reikalavimai – SQL kodo generavimo šablonai. Kitame stereotipo apdorojimo žingsnyje iš loginio duomenų modelio nuskaitomos stereotipų žymių reikšmės. Žymėtosios reikšmės SQL kodo šablone pakeičiamos reikšmėmis iš loginio modelio ir vientisumo reikalavimų profilio. Gautas SQL kodas išvedamas į išvesties įrenginį. Šis algoritmas pavaizduotas 6 paveiksle.



6 pav. Duomenų modelio elemento stereotipo apdorojimas[4]

Pagal pateiktą algoritmą apdorojus visus diagramos stereotipus, sukuriamas SQL kodas, kuriuo papildomas egzistuojančio *MagicDraw* SQL generavimo įrankio sugeneruojamas kodas.

4. Metodo realizacija UML CASE įrankyje MagicDraw

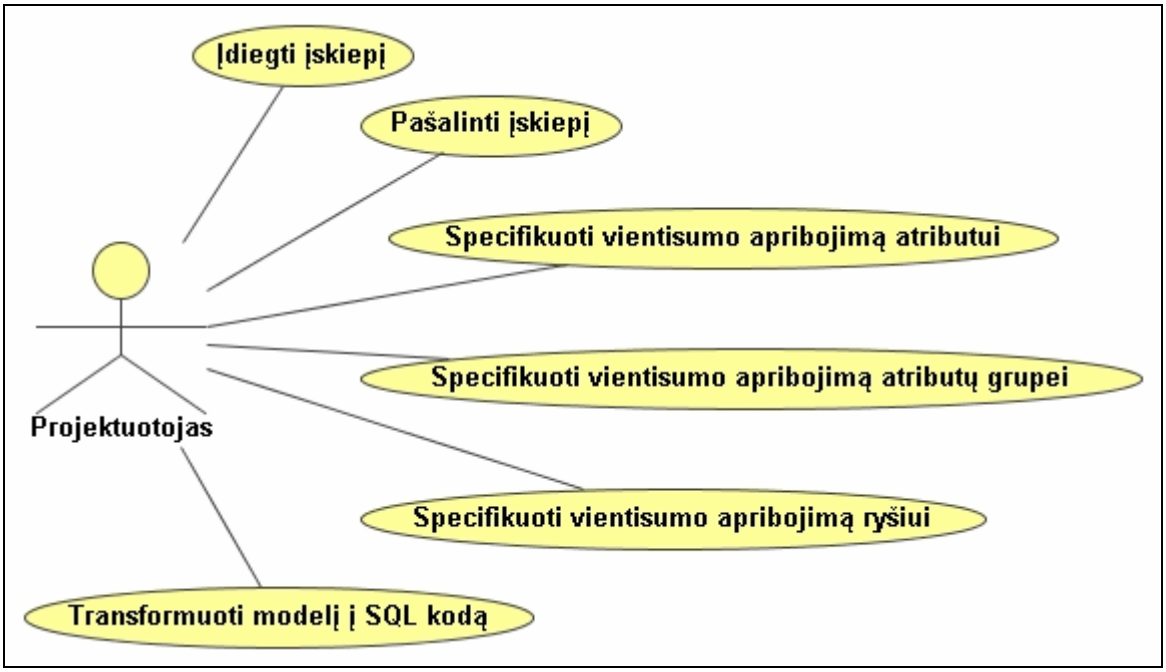
Šiame skyriuje aprašoma sistemos realizacija firmos NoMagic programiniame pakete MagicDraw. Pateikiamas sistemos dinaminis ir statinis vaizdas. Specifikuojami funkciniai ir nefunkciniai sistemos reikalavimai. Papildomai pateikiami sistemos kokybės užtikrinimo – testavimo metodai. Supažindinama su realizacijos vartotojo sąsaja.

4.1. Sistemos reikalavimų specifikacija

Šiame poskyryje detaliai aprašomi sistemoje realizuoti panaudojimo atvejai ir nefunkciniai reikalavimai. Funkciniai reikalavimai pateikiami panaudojant Volere šablonus.

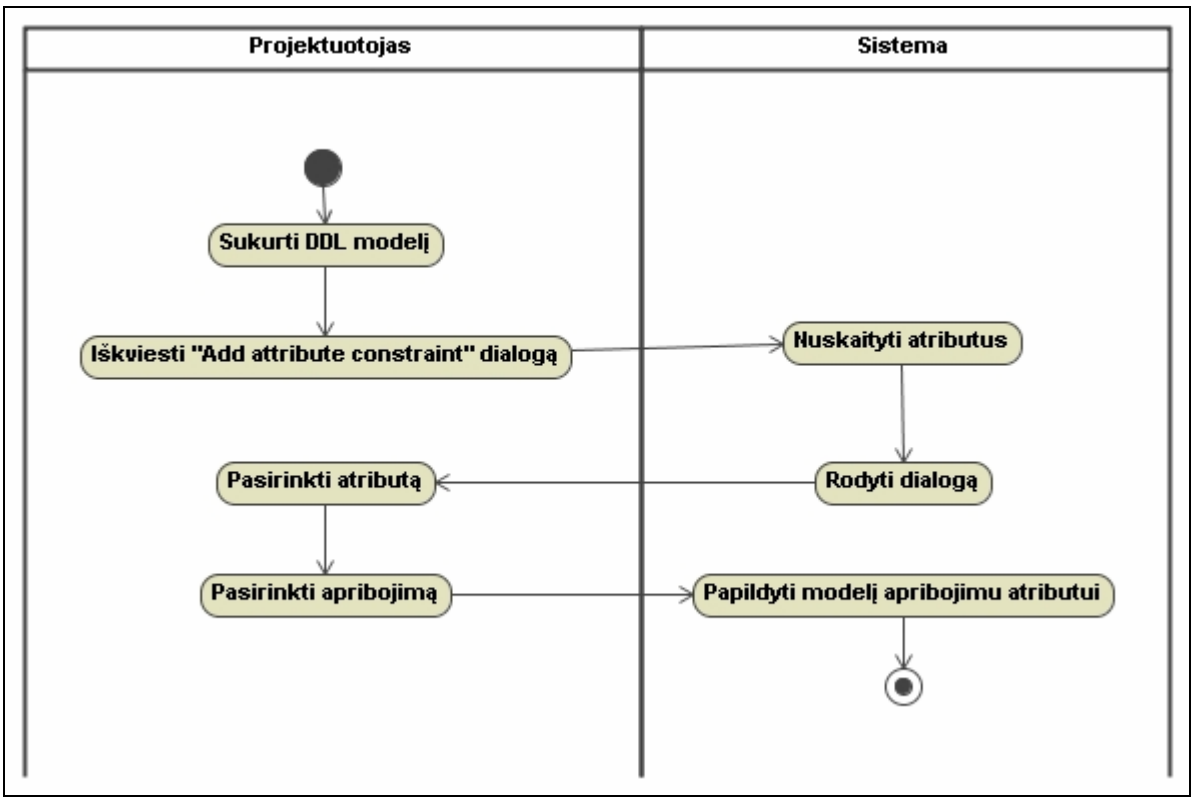
4.1.1. Panaudojimo atvejų modelis

Sistema realizuoja panaudojimo atvejus pateiktus 7 pav.:

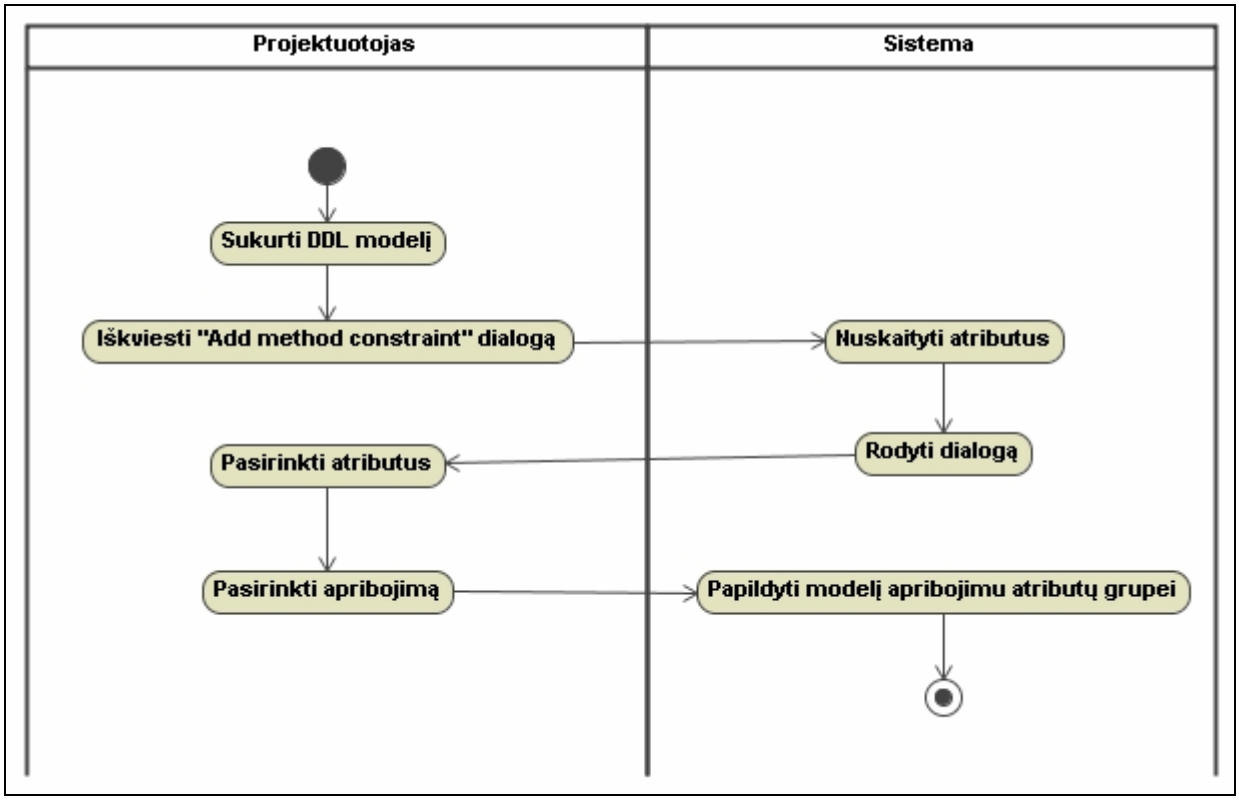


7 pav. *Kontekstinė diagrama*

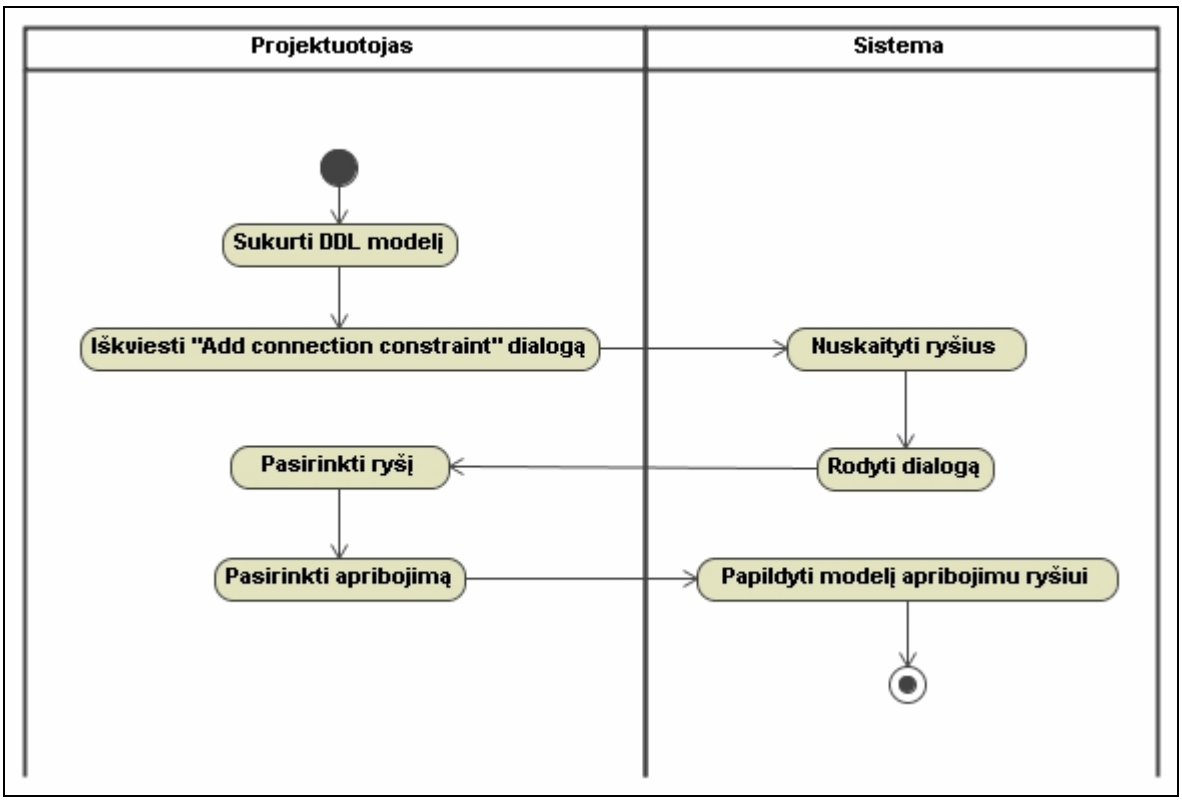
Panaudojimo atvejus aprašančios veiklos diagramos pateikiamos 8-11 paveiksluose, būsenų diagrama – 12 paveiksle.



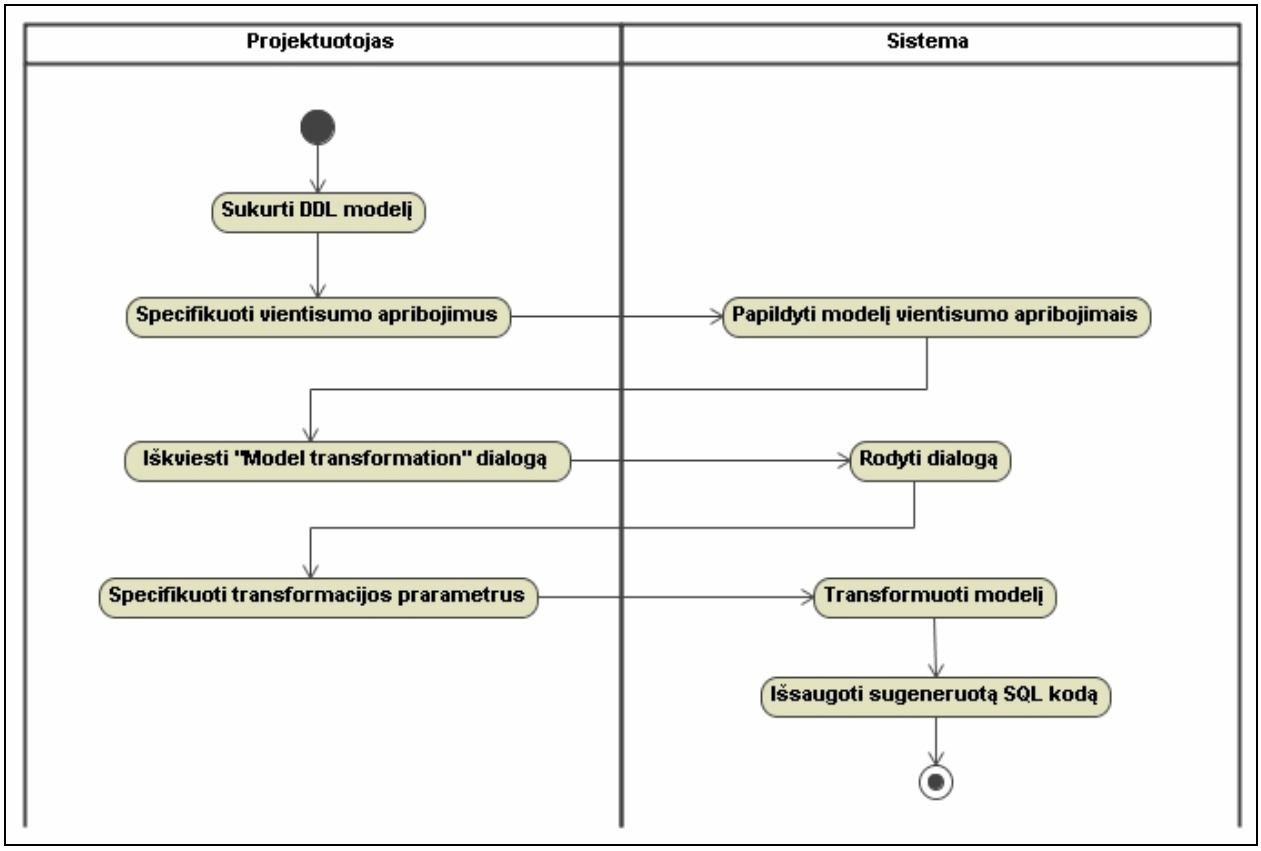
8 pav. *PA "Specifikuoti VA atributui" veiklos diagrama.*



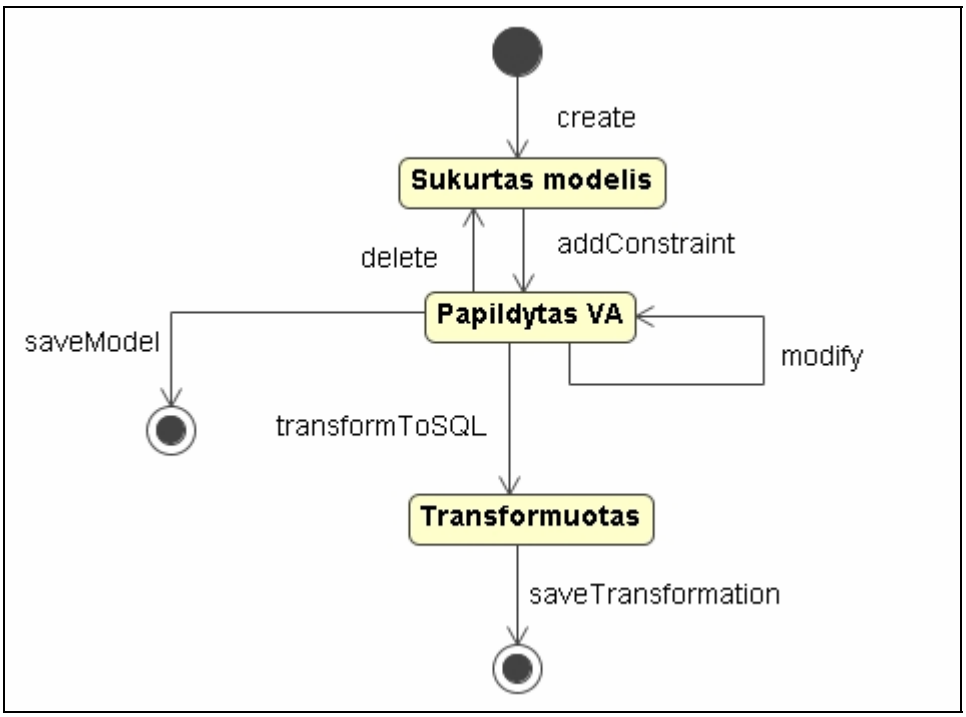
9 pav. *PA "Specifikuoti VA atributų grupei" veiklos diagrama.*



10 pav. *PA "Specifikuoti VA ryšiui" veiklos diagrama.*



11 pav. *PA "Specifikuoti VA ryšiui" veiklos diagrama.*



12 pav. *Modelio būsenų diagrama.*

4.1.2. Panaudojimo atvejų specifikacijos

1. PANAUDOJIMO ATVEJIS: Įskiepio diegimas **Vartotojas/Aktorius:** Vartotojas **Aprašas:** Vientisumo reikalavimų aprašymo ir kodo generavimo įskiepio diegimas į MagicDraw programinį paketą. **Prieš sąlyga:** Įdiegtas MagicDraw programinis paketas. **Sužadinimo sąlyga:** Aktorius nori įdiegti įskiepį. **Po-sąlyga:** Įskiepis įdiegtas.

2. PANAUDOJIMO ATVEJIS: Įskiepio šalinimas **Vartotojas/Aktorius:** Vartotojas **Aprašas:** Vientisumo reikalavimų aprašymo ir kodo generavimo įskiepio šalinimas iš MagicDraw programinio paketo. **Prieš sąlyga:** Įskiepis yra įdiegtas. **Sužadinimo sąlyga:** Aktorius nori pašalinti įskiepį. **Po-sąlyga:** Įskiepis pašalintas iš MagicDraw programinio paketo.

3. PANAUDOJIMO ATVEJIS: Vientisumo apribojimo atributui specifikavimas

Vartotojas/Aktorius: Sistema, vartotojas

Aprašas: Vartotojas papildo duomenų bazės loginį modelį vientisumo apribojimu atributui.

Prieš sąlygos: Veikianti kuriamos programinės įrangos instaliacija.

Duomenų bazės modelyje egzistuoja bent viena esybė su bent vienu atributu.

Sužadinimo sąlyga: Vartotojas nori papildyti duomenų bazės modelį vientisumo apribojimu atributui.

Po-sąlyga: Duomenų bazės modelis papildytas vientisumo apribojimu atributui.

4. PANAUDOJIMO ATVEJIS: Vientisumo apribojimo specifikavimas atributų grupei

Vartotojas/Aktorius: Sistema, vartotojas

Aprašas: Vartotojas papildo duomenų bazės loginį modelį vientisumo apribojimu atributų grupei.

Prieš sąlyga: Veikianti kuriamos programinės įrangos instaliacija.

Duomenų bazės modelyje egzistuoja bent viena esybė su ne mažiau kaip dviem atributais.

Sužadinimo sąlyga: Vartotojas nori papildyti duomenų bazės modelį vientisumo apribojimu atributų grupei.

Po-sąlyga: Duomenų bazės modelis papildytas vientisumo apribojimu atributų grupei.

5. PANAUDOJIMO ATVEJIS: Vientisumo specifikavimas ryšiams tarp esybių

Vartotojas/Aktorius: Sistema, vartotojas

Aprašas: Vartotojas papildo duomenų bazės loginį modelį vientisumo apribojimu ryšiams tarp esybių.

Prieš sąlyga: Veikianti kuriamos programinės įrangos instaliacija. Duomenų bazės modelyje egzistuoja ne mažiau kaip viena esybė ir ne mažiau kaip vienas ryšys.

Sužadinimo sąlyga: Vartotojas nori papildyti duomenų bazės modelį vientisumo apribojimu ryšiams tarp esybių.

Po-sąlyga: Duomenų bazės modelis papildytas vientisumo apribojimu ryšiams tarp esybių.

6. PANAUDOJIMO ATVEJIS: Modelio transformacija į SQL kodą

Vartotojas/Aktorius: Sistema, vartotojas

Aprašas: MagicDraw DB modelio su vientisumo reikalavimais transformavimas į SQL programinį kodą.

Prieš sąlygos:

Egzistuoja duomenų bazės projektas sukurtas MagicDraw programinėm priemonėm.

Transformuojamas modelis neturi sintaksės klaidų.

Sužadinimo sąlyga: Aktorius nori transformuoti sukurtą duomenų bazės modelį į SQL programinį kodą.

Po-sąlyga: Sugeneruotas modelio SQL programinis kodas gali būti panaudotas duomenų bazės sukūrimui.

4.1.3. Funkciniai reikalavimai

Reikalavimas #:	1	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	1
Aprašymas:	Įskiepio diegimo procedūra neturi skirtis nuo standartinės MagicDraw įskiepių diegimo procedūros				
Pagrindimas:	Apsunkinamas vartotojo darbas				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	Įskiepio diegimo procedūros palyginimas su diegimo procedūra aprašyta MagicDraw dokumentacijoje				
Užsakovo tenkinimas:	4			**Užsakovo netenkinimas:**	2
Priklausomybės	Nėra			**Konfliktai:**	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2006.11.10				

Reikalavimas #:	2	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	2
Aprašymas:	Įskiepio šalinimo procedūra neturi skirtis nuo standartinės MagicDraw įskiepių šalinimo procedūros				

Pagrindimas:	Apsunkinamas vartotojo darbas		
Šaltinis:	Užsakovas		
Tikimo kriterijus:	Įskiepio šalinimo procedūros palyginimas su šalinimo procedūra aprašyta MagicDraw dokumentacijoje		
Užsakovo tenkinimas:	4	**Užsakovo netenkinimas:**	3
Priklausomybės	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2006.11.10		

Reikalavimas #:	3	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	3
Aprašymas:	Sistema grafiškai pavaizduoja vientisumo atribojimus atributams kaip atributų stereotipus				
Pagrindimas:	Standartinis vientisumo reikalavimų pavaizdavimo būdas MagicDraw DB modeliuose. Reikalavimas atvaizduojamas kaip klasės atributo savybė-stereotipas				
Šaltinis:	Analitikas				
Tikimo kriterijus:	MagicDraw ir UML dokumentacija.				
Užsakovo tenkinimas:	5			**Užsakovo netenkinimas:**	5
Priklausomybės	Nėra			**Konfliktai:**	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2006.11.10				

Reikalavimas #:	4	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	4
Aprašymas:	Sistema privalo grafiškai pavaizduoti vientisumo atribojimus atributų grupei kaip klasės metodą				
Pagrindimas:	Standartinis vientisumo reikalavimų pavaizdavimo būdas MagicDraw DB modeliuose. Reikalavimas pavaizduojamas kaip klasės metodo savybė-stereotipas				
Šaltinis:	Analitikas				

Tikimo kriterijus:	MagicDraw ir UML dokumentacija		
Užsakovo tenkinimas:	5	**Užsakovo netenkinimas:**	5
Priklausomybės	Nėra	**Konfliktai:**	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2006.11.10		

Reikalavimas #:	5	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	5
Aprašymas:	Sistema privalo grafiškai pavaizduoti vientisumo atribojimus ryšiams tarp esybių				
Pagrindimas:	Standartinis vientisumo pavaizdavimo būdas MagicDraw DB modeliuose. Reikalavimas atvaizduojamas kaip klasės ryšio savybė-stereotipas				
Šaltinis:	Analitikas				
Tikimo kriterijus:	MagicDraw ir UML dokumentacija.				
Užsakovo tenkinimas:	3			**Užsakovo netenkinimas:**	3
Priklausomybės	Nėra			**Konfliktai:**	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2006.11.10				

Reikalavimas #:	6	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	6
Aprašymas:	Galimybė vientisumo apribojimus transformuoti į SQL kodą				
Pagrindimas:	SQL kodas reikalingas fizinio duomenų bazės modelio generavimui				
Šaltinis:	Analitikas				
Tikimo kriterijus:	Fizinio duomenų bazės modelio atitikimas loginiam duomenų bazės modeliui				
Užsakovo tenkinimas:	4			**Užsakovo netenkinimas:**	3
Priklausomybės	7, 8, 9			**Konfliktai:**	Nėra
Papildoma					

medžiaga:	
Istorija:	Užregistruotas 2006.11.10

Reikalavimas #:	7	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	3
Aprašymas:	Galimybė specifikuoti sekančius vientisumo apribojimus atributui: ▪ disjunktyvaus būtinumo reikalavimas ▪ išorinio unikalumo reikalavimas ▪ atributo reikšmės reikalavimas				
Pagrindimas:	Loginio duomenų bazės modelio pilnumas negali būti užtikrintas be vientisumo reikalavimų atributui				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	Loginio duomenų bazės modelio kūrimas panaudojant apribojimus atributui				
Užsakovo tenkinimas:	5			**Užsakovo netenkinimas:**	5
Priklausomybės	3			**Konfliktai:**	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2006.11.10				

Reikalavimas #:	8	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	4
Aprašymas:	Galimybė specifikuoti sekančius vientisumo apribojimus atributų grupei: ▪ disjunktyvaus būtinumo reikalavimas ▪ ekvivalentiškumo reikalavimas ▪ poaibio reikalavimas ▪ išrinktinumo reikalavimas				
Pagrindimas:	Loginio duomenų bazės modelio pilnumas negali būti užtikrintas be vientisumo reikalavimų atributų grupei				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	Loginio duomenų bazės modelio kūrimas panaudojant apribojimus atributų grupei				
Užsakovo	5			**Užsakovo netenkinimas:**	4

tenkinimas:			
Priklausomybės	4	**Konfliktai:**	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2006.11.10		

Reikalavimas #:	9	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	5
Aprašymas:	Galimybė specifikuoti sekančius vientisumo reikalavimus ryšiams tarp esybių: ▪ ekvivalentiškumo reikalavimai ▪ poaibio reikalavimas ▪ išskirtinumo reikalavimas ▪ refleksyviojo ryšio reikalavimas ▪ egzempliorių susiejimo ryšio reikalavimas				
Pagrindimas:	Loginio duomenų bazės modelio pilnumas negali būti užtikrintas be vientisumo reikalavimų ryšiams tarp esybių				
Šaltinis:	Užsakovas				
Tikimo kriterijus:	Loginio duomenų bazės modelio kūrimas panaudojant reikalavimus ryšiams tarp esybių				
Užsakovo tenkinimas:	5			**Užsakovo netenkinimas:**	2
Priklausomybės	5			**Konfliktai:**	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2006.11.10				

Reikalavimas #:	10	**Reikalavimo tipas:**	F	**Įvykis/panaudojimo atvejis #:**	6
Aprašymas:	Vientisumo reikalavimų sintaksės patikrinimas				
Pagrindimas:	Noras sumažinti sugeneruoto SQL kodo redagavimo poreikį				
Šaltinis:	Analitikas				
Tikimo kriterijus:	Klaidų skaičius transformuojant loginį duomenų bazės modelį į SQL kodą				
Užsakovo	3			**Užsakovo netenkinimas:**	3

tenkinimas:			
Priklausomybės	6	**Konfliktai:**	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2006.11.10		

4.1.4. Nefunkciniai reikalavimai

- **Reikalavimai sistemos išvaizdai:**
  - ✓ Įskiepis naudoja MagicDraw programinio paketo grafinę vartotojo sąsają. Kuo mažiau pakeisti esamą varotojo sąsają. Maksimaliai išnaudoti esamus vartotojo sąsajos elementus ir funkcionalumą;
  - ✓ Pakeitimai negali keisti bendros MagicDraw išvaizdos. Bet kokie sistemos pakeitimai turi būti vartotojui intuityviai suprantami, logiški ir aiškus.

- **Reikalavimai panaudojamumui:**
  - ✓ Produktas turi būti intuityviai suprantamas kaip patyrusiems taip ir naujiems MagicDraw vartotojams;
  - ✓ Produkto panaudojamumas neturi išsiskirti iš esamo MagicDraw panaudojamumo koncepcijos.

- **Reikalavimai išplečiamumui:**
  - ✓ Produktas turi būti lengvai išplečiamas naujais vientisumo reikalavimais, nekeičiant programos išeities kodo;
  - ✓ Produktas turi būti lengvai išplečiamas naujais SQL dialektais, nekeičiant programos išeities kodo.

- **Reikalavimai sistemos suderinamumui ir realizacijai:**
  - ✓ Įskiepis turi būti realizuotas naudojant JAVA5 programavimo kalbą;
  - ✓ Naudojama notacija turi atitikti UML standartą;
  - ✓ Generuojamas kodas turi atitikti SQL3 standartą;
  - ✓ Įskiepis turi palaikyti JMI ir XMI standartus.
  - ✓ Sistema suderinta su MagicDraw 12.X, 14.0 programiniu paketu;

- **Reikalavimai vykdymo charakteristikoms:**

- ✓ Dėl įskiepio gali sulėtėti programos startavimo greitis, tačiau ne daugiau kaip 25%;
- ✓ Modelio transformacijos greitis į SQL programinį kodą negali padidėti daugiau kaip 50%;
- ✓ Bendra MagicDraw sparta dirbant su kitais modeliais nei duomenų bazės, po įskiepio įdiegimo, negali pablogėti.

- **Sistemos veikimo sąlygos:**
  - ✓ Sistema neturi kelti reikalavimų MagicDraw programos veikimui;
  - ✓ Visos sistemos veikimo sąlygos keliamos MagicDraw programiniam paketui turi galioti ir kuriamai sistemai.

- **Reikalavimai sistemos priežiūrai:**
  - ✓ Visos problemos ir sistemos statusas turi būti dokumentuojami MagicDraw programinio paketo žurnalinėse bylose (*angl*. log file);
  - ✓ Sistemos priežiūros instrukcijos, galimos klaidos ir jų sprendimo būdai turi būti aprašyti atskirame dokumente arba pateikti kaip vartotojo vadovo dokumentacijos dalis.

4.2. Sistemos architektūra

Šiame poskyryje pateikiama magistrinio darbo realizacijos architektūra. Detaliai aprašomi sistemos dinaminis ir statinis modeliai.

4.2.1. Architektūros tikslai ir reikalavimai

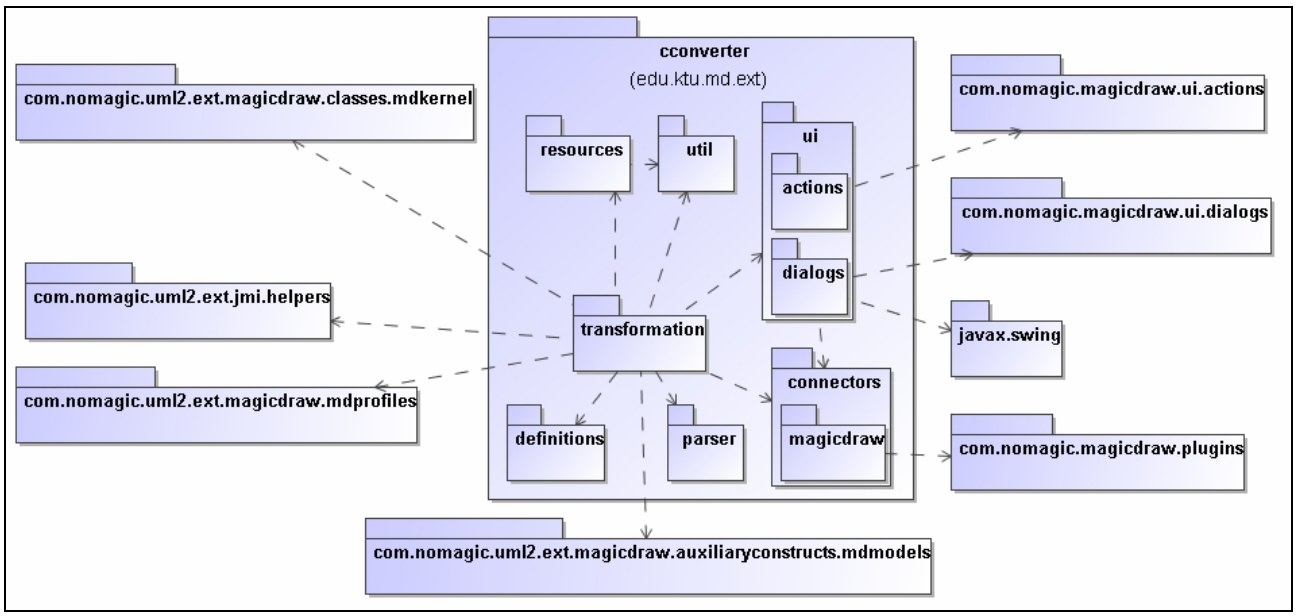
- **Kuriamos sistemos architektūros tikslai:**
  - ✓ Maksimaliai galima integracija į MagicDraw modeliavimo įrankį;
  - ✓ Minimalus pokyčiai MagicDraw vartotojo sąsajoje;
  - ✓ Nepriklausomumas nuo platformos - įrankiu galima naudotis visose operacinėse sistemose, kuriose yra įdiegta Java virtuali mašina.

- **Kuriamos sistemos reikalavimai:**
  - ✓ Kuriama sistema turi būti integruota į MagicDraw modeliavimo įrankį.
  - ✓ Naudojama notacija turi atitikti UML standartą.
  - ✓ Generuojamas kodas turi atitikti SQL3 standartą.
  - ✓ Įskiepis turi palaikyti JMI ir XMI standartus.

- ✓ Sistema turi būti suderinama su MagicDraw 12 programiniu paketu.
- ✓ Sistema neturi kelti papildomų reikalavimų programinei ir techninei įrangai, negu nurodyta MagicDraw techniniuose reikalavimuose.

4.2.2. Loginė architektūra

Programinė įranga kuriama kaip MagicDraw įskiepis. Kuriamos sistemos realizacijai reikalinga paketų struktūra atrodo taip (13 pav.):



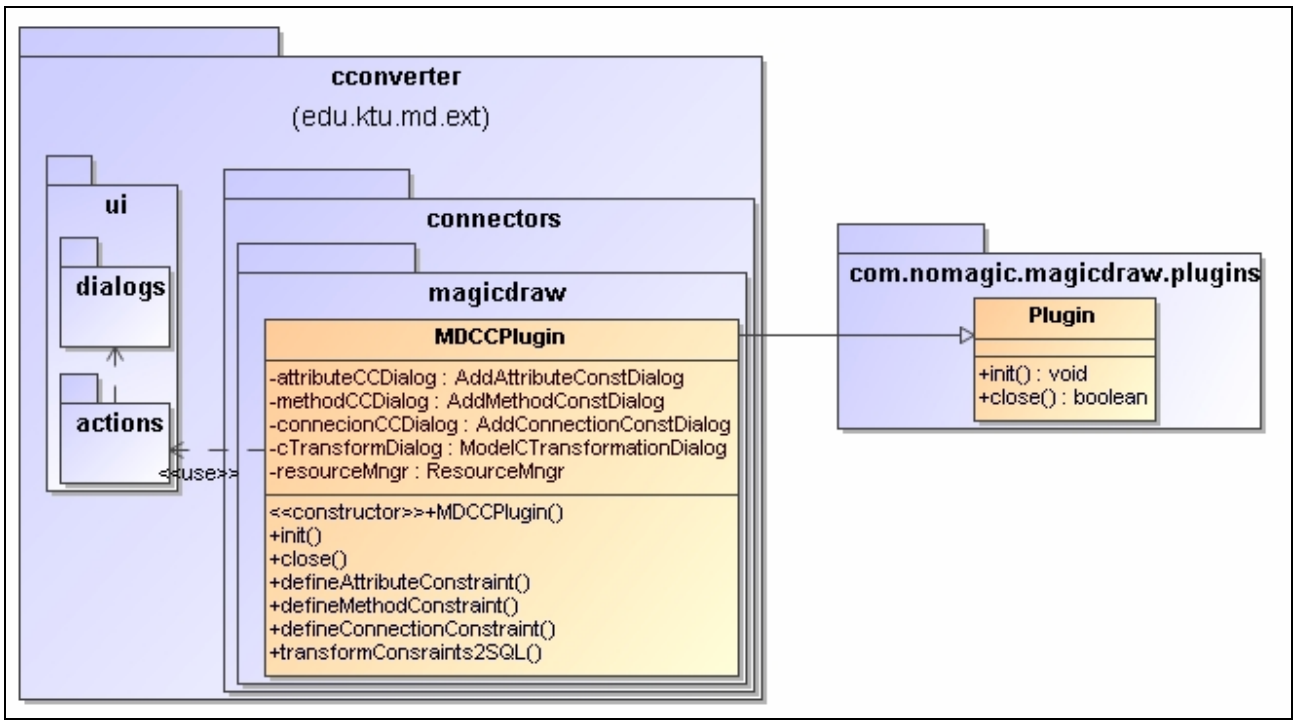
13 pav. *Sistemos paketų struktūros diagrama*

4.2.3. Klasių modelis

edu.ktu.md.ext.cconverter – sąsajos su MagicDraw.

edu.ktu.md.ext.cconverter.connectors – išorinių programų bendravimo sąsajų paketas.

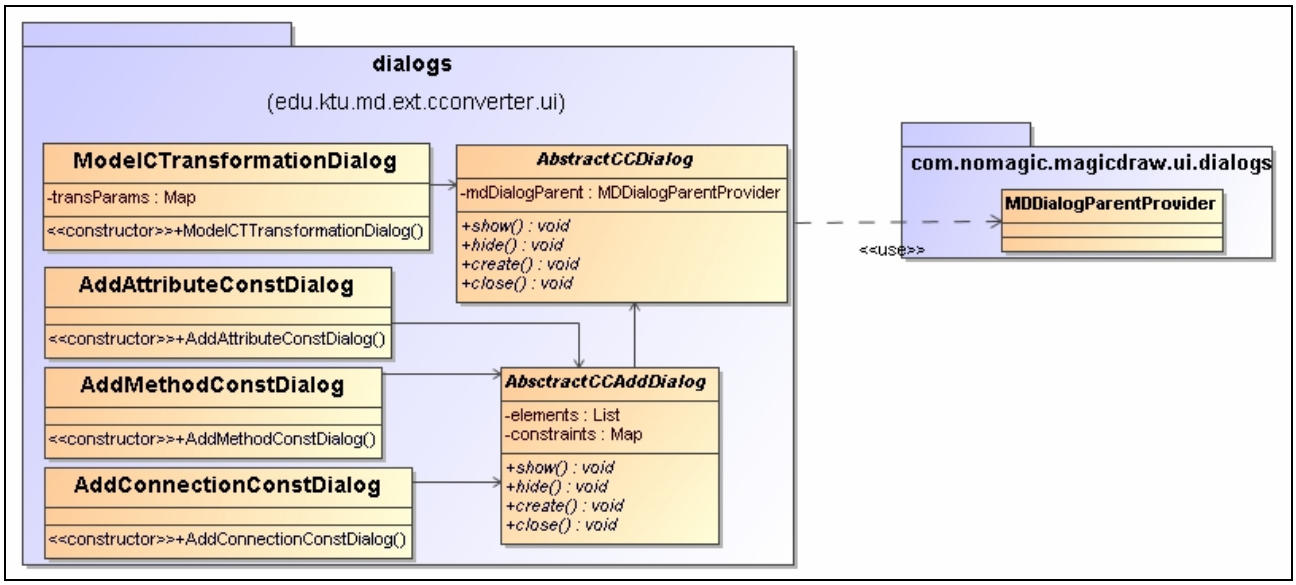
edu.ktu.md.ext.cconverter.connectors.magicdraw - sąsaja skirta sistemos bendravimui su MagicDraw modeliavimo įrankiu.



14 pav. *cconverter paketo struktūros diagrama*

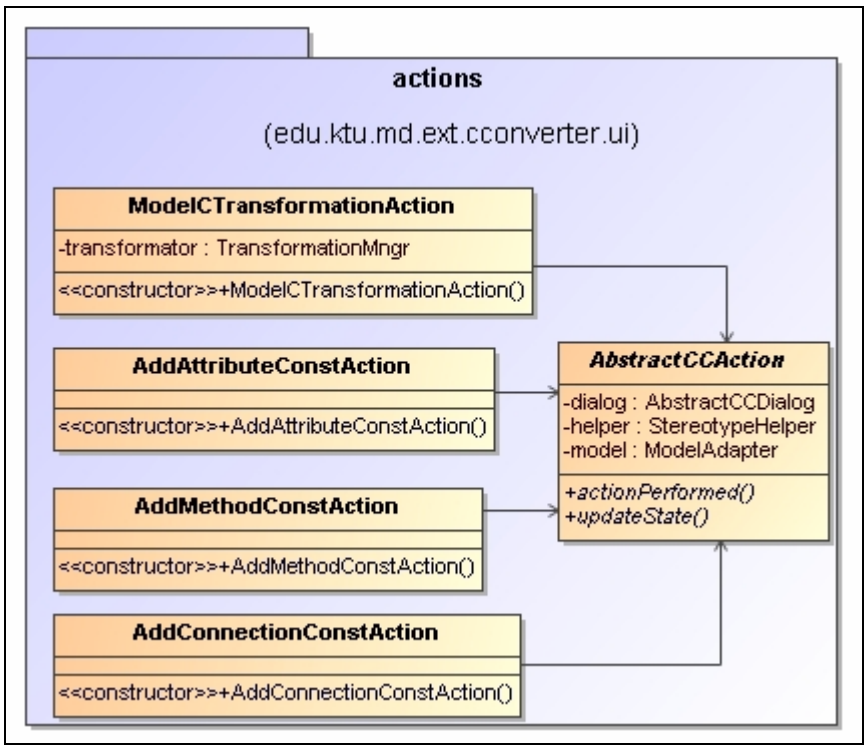
edu.ktu.md.ext.cconverter.ui – grafinio vartotojo sąsajos klasių paketas.

edu.ktu.md.ext.cconverter.ui.dialogs – grafinių dialogų paketas.



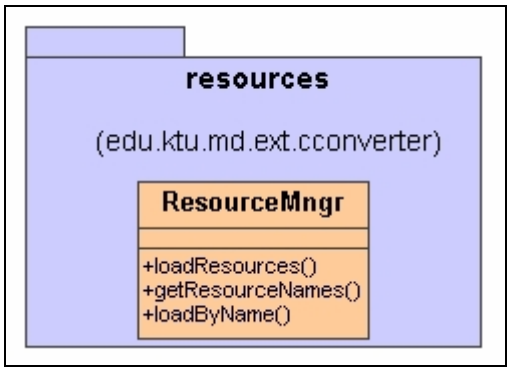
15 pav. *dialogs paketo struktūros diagrama*

edu.ktu.md.ext.cconverter.ui.actions – klasių realizuojančių sistemos panaudojimo atvejus paketas.



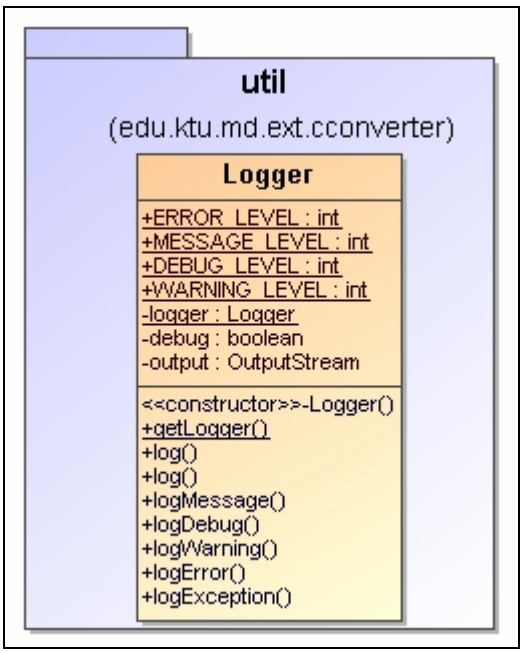
16 pav. *actions paketo struktūros diagrama*

edu.ktu.md.ext.cconverter.resources – įskiepio resursų ir resursų tvarkymo klasių paketas.



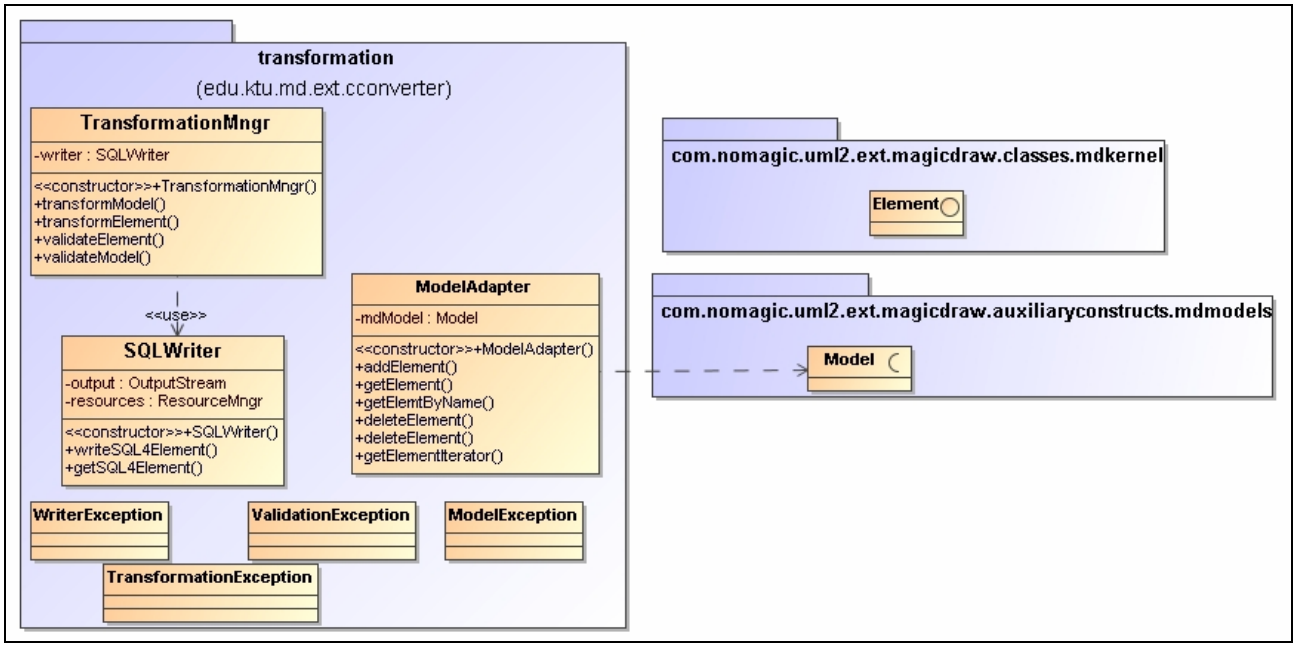
17 pav. *resources paketo struktūros diagrama*

edu.ktu.md.ext.cconverter.util – pagalbinių klasių paketas.



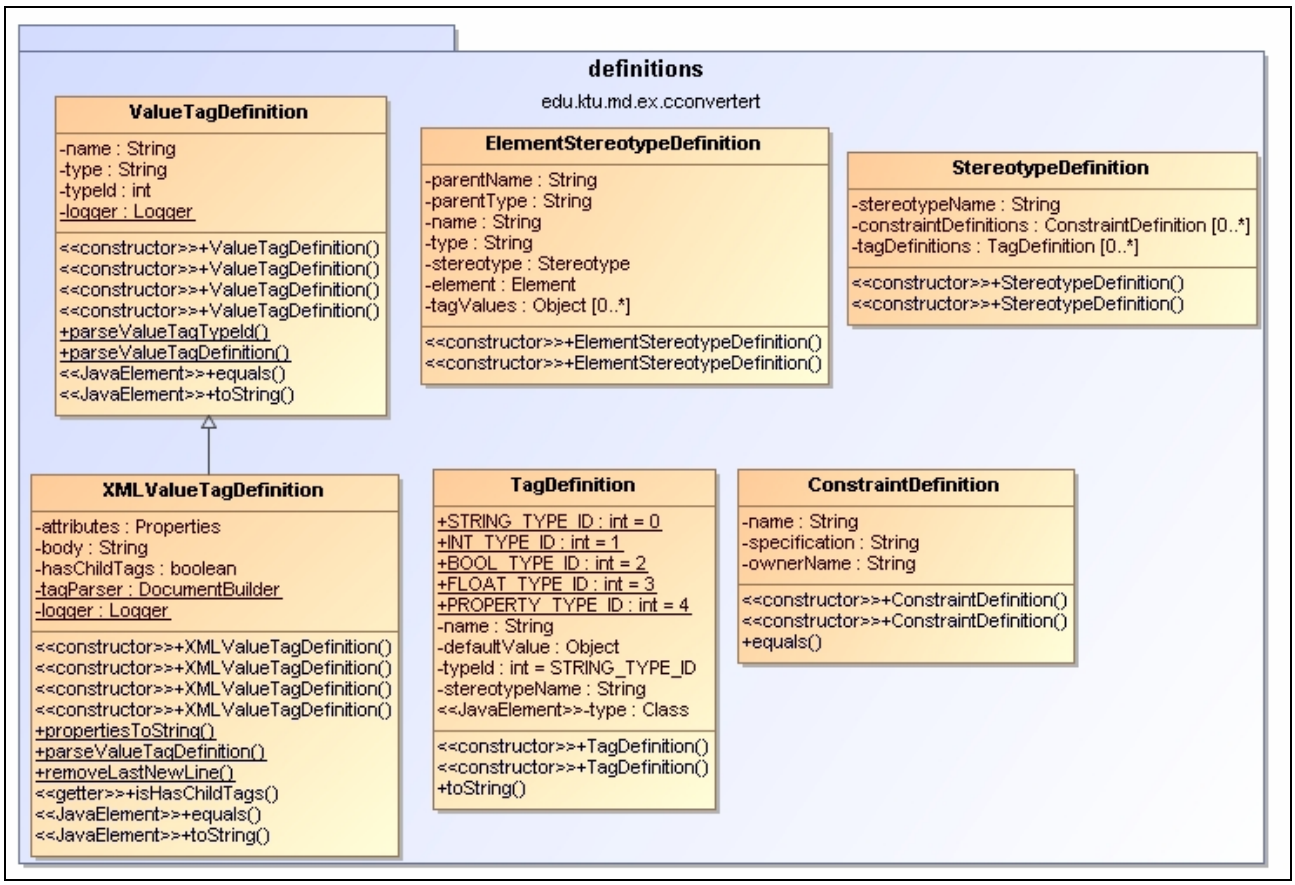
18 pav. *util paketo struktūros diagrama*

edu.ktu.md.ext.cconverter.transformation – transformacijos klasių paketas (19 pav.):



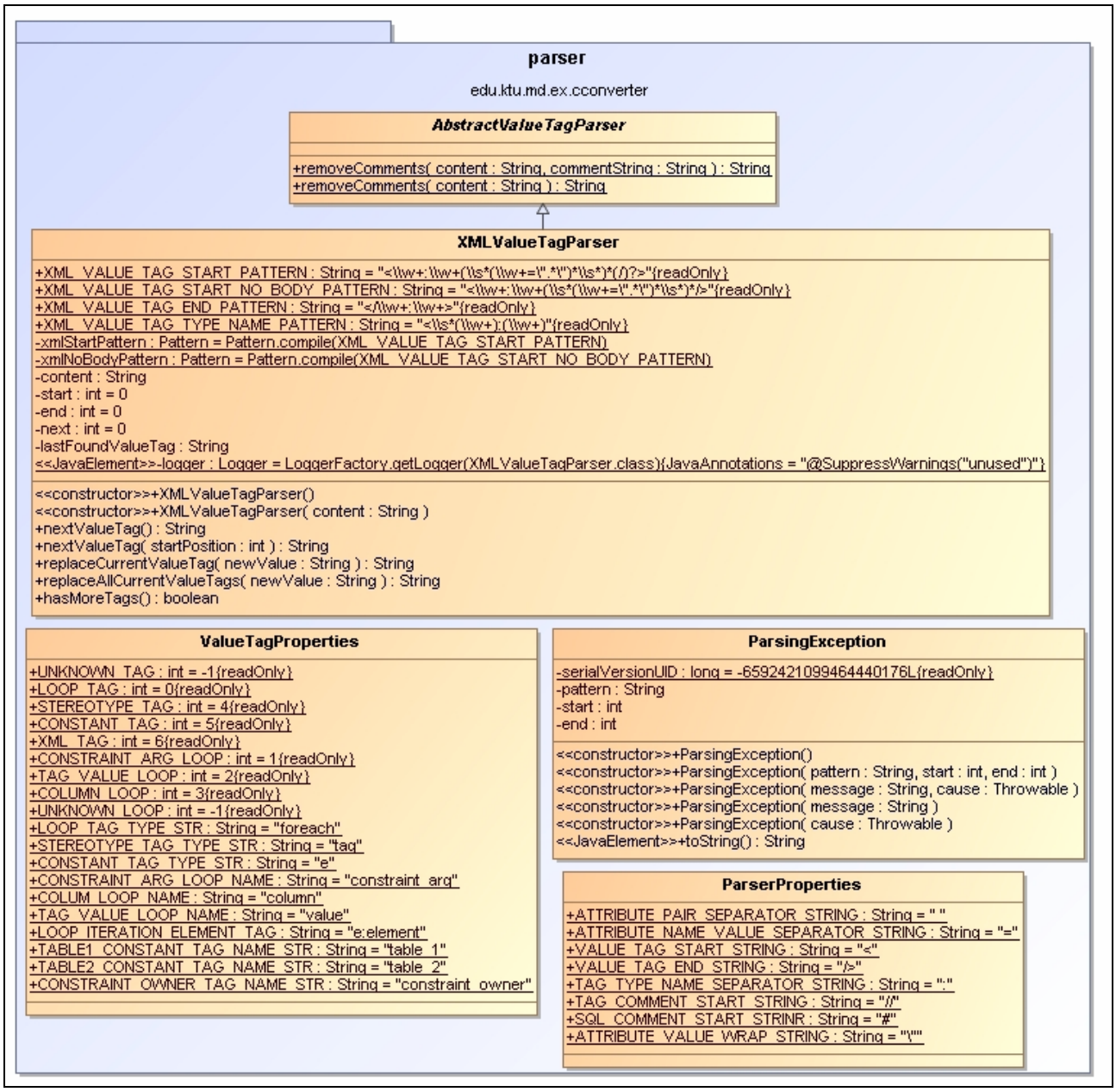
19 pav. *transformation paketo struktūros diagrama*

edu.ktu.md.ext.cconverter.definitions – SQL šablonų klasių paketas (20 pav.):



20 pav. *definitions paketo struktūros diagrama*

edu.ktu.md.ext.cconverter.parser – SQL šablonų skaitymo klasių paketas (21 pav.):

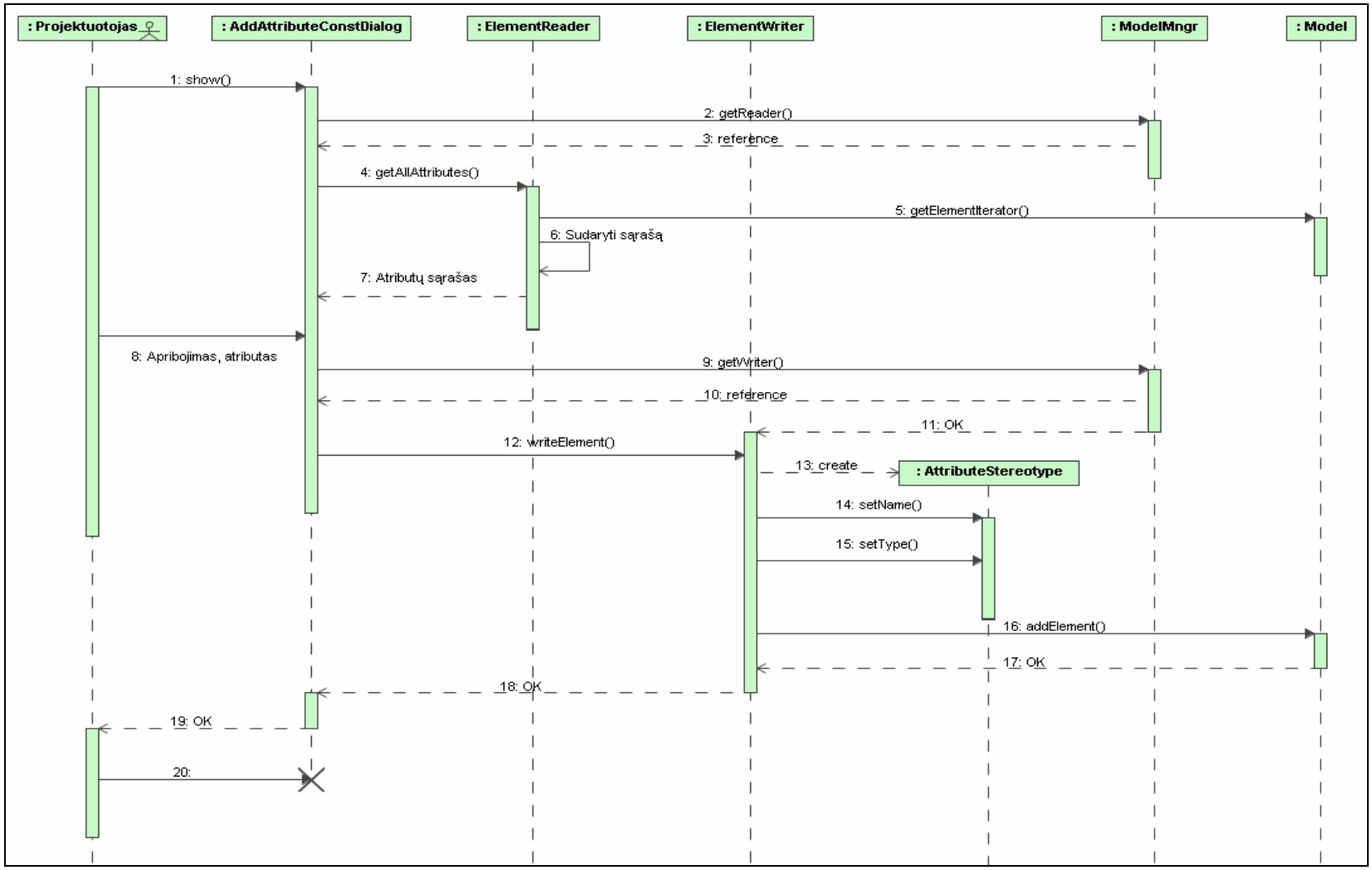


21 pav. *definitions paketo struktūros diagrama*

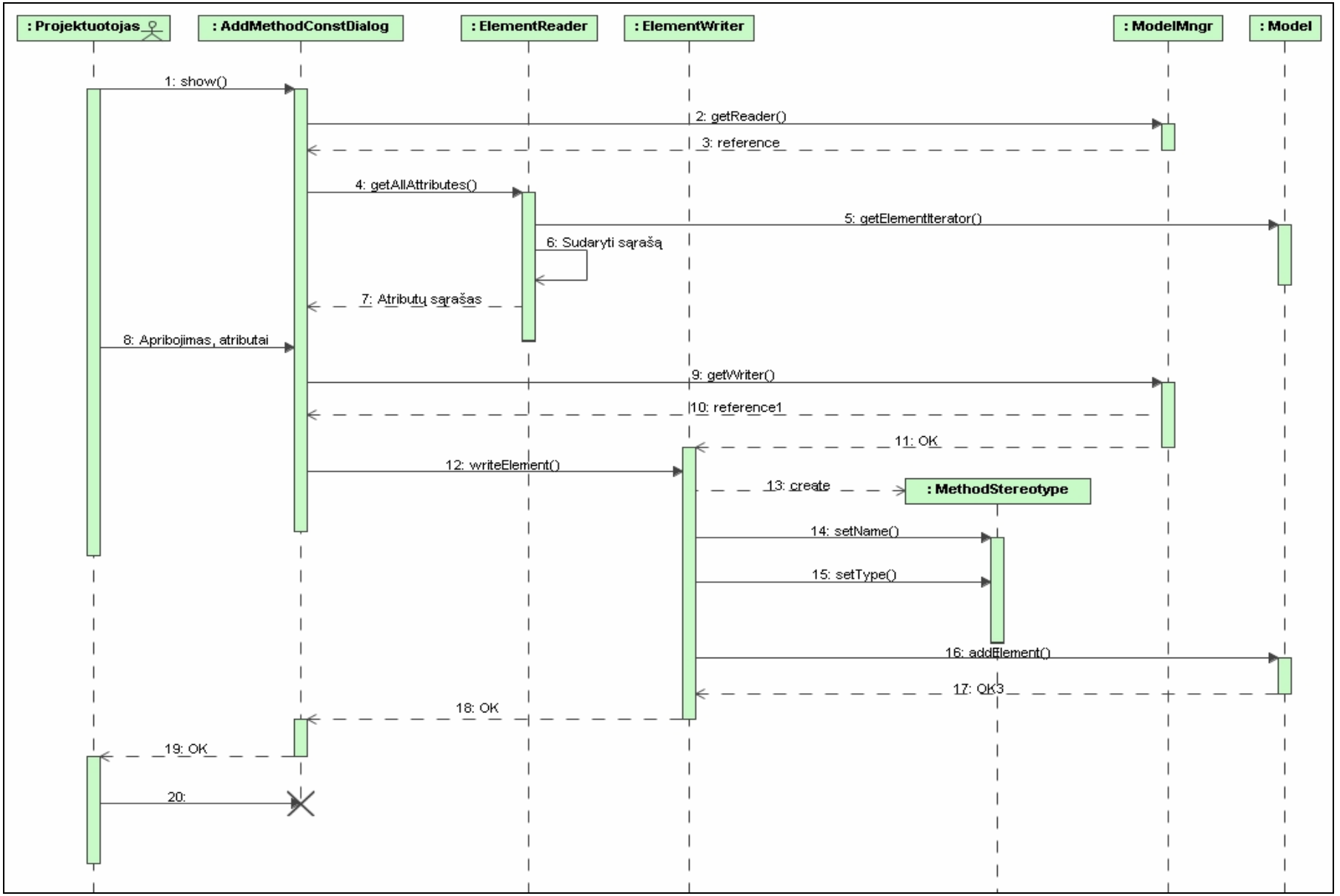
com.nomagic.magicdraw.plugins – firmos NoMagic klasių paketas įskiepių integracijai į MagicDraw programinį paketą.

4.2.4. Sistemos elgsenos modelis

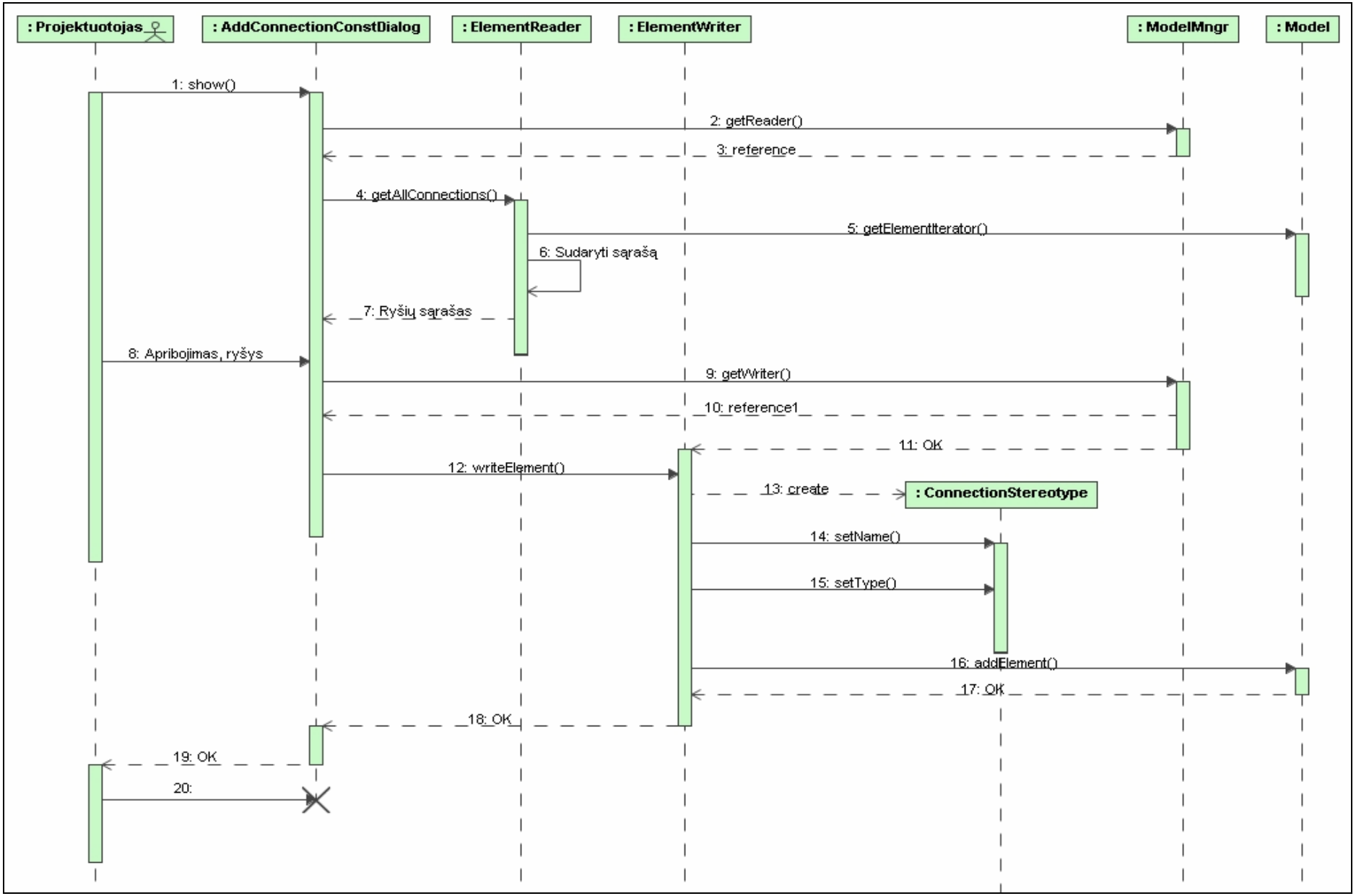
Šiame skyriuje pateikiamos sekų diagramos:



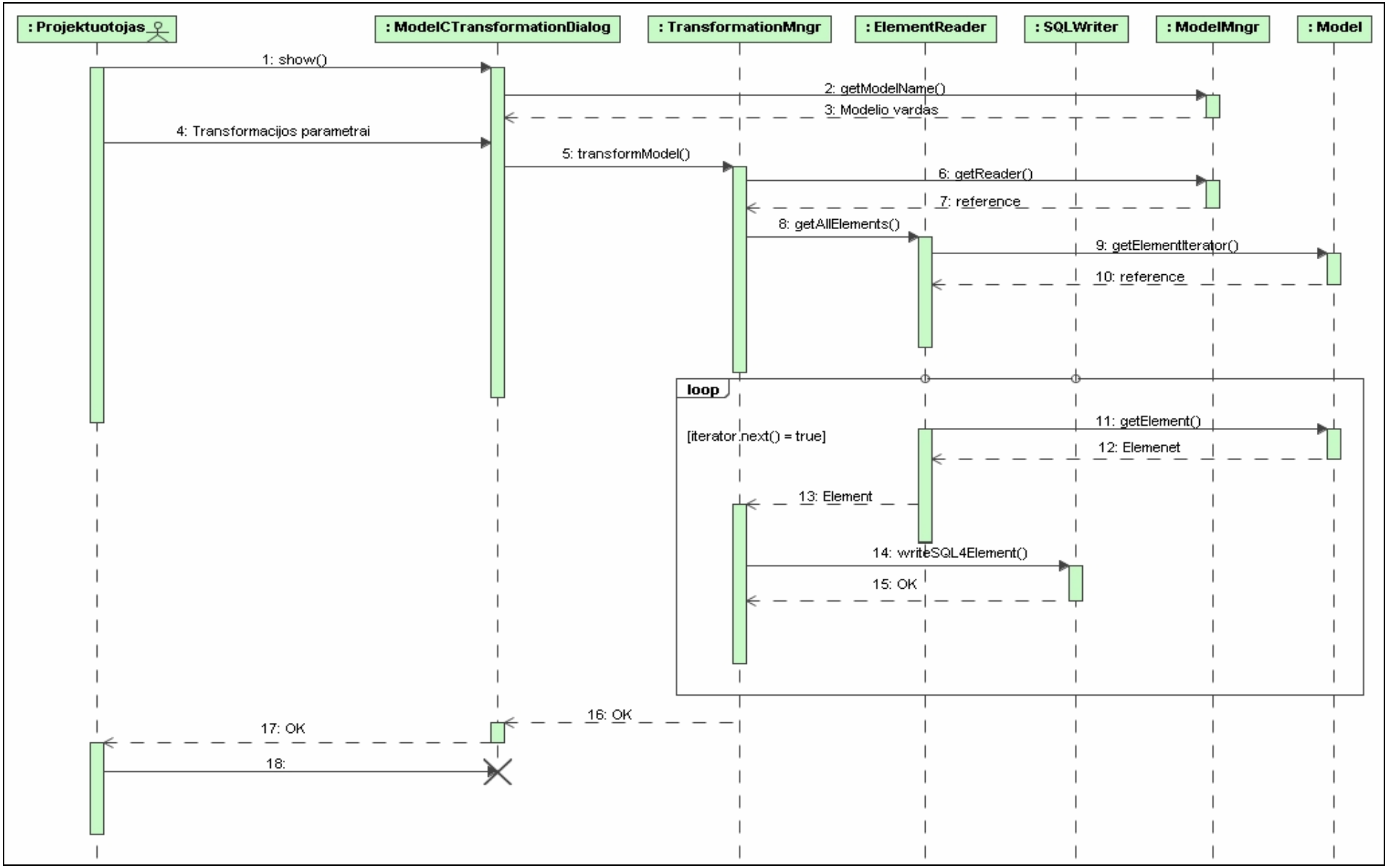
22 pav. *PA "Specifikuoti VA atributui" sekų diagrama.*



23 pav. *PA "Specifikuoti VA atributų grupei" sekų diagrama.*



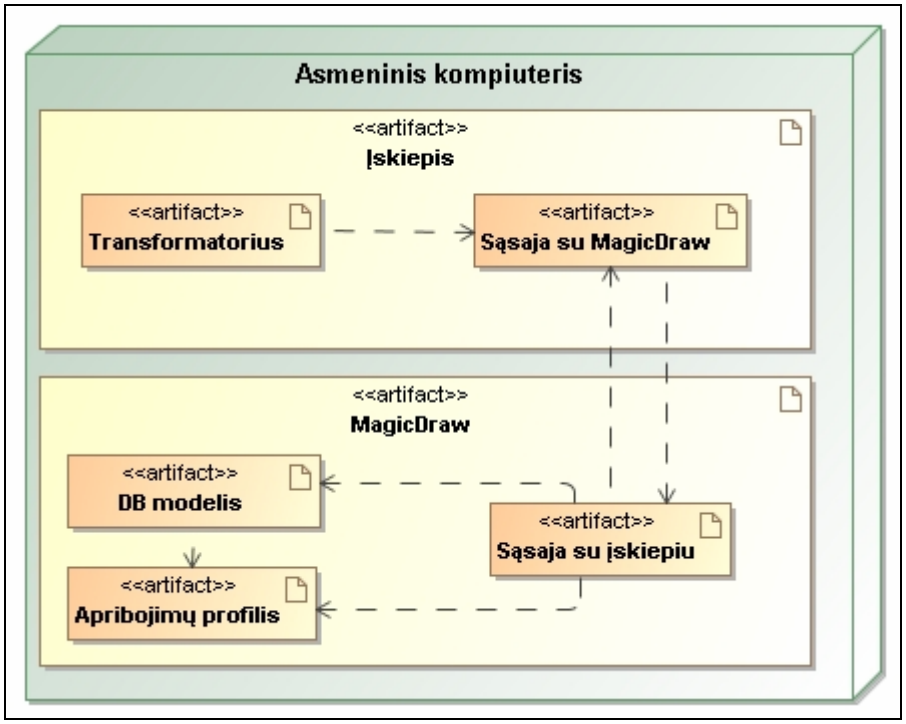
24 pav. *PA "Specifikuoti VA ryšiui" sekų diagrama.*



25 pav. *PA "Transformuoti modelį į SQL kodą" sekų diagrama.*

4.2.5. Realizacijos modelis

Komponentų išdėstymo diagrama pateikiama

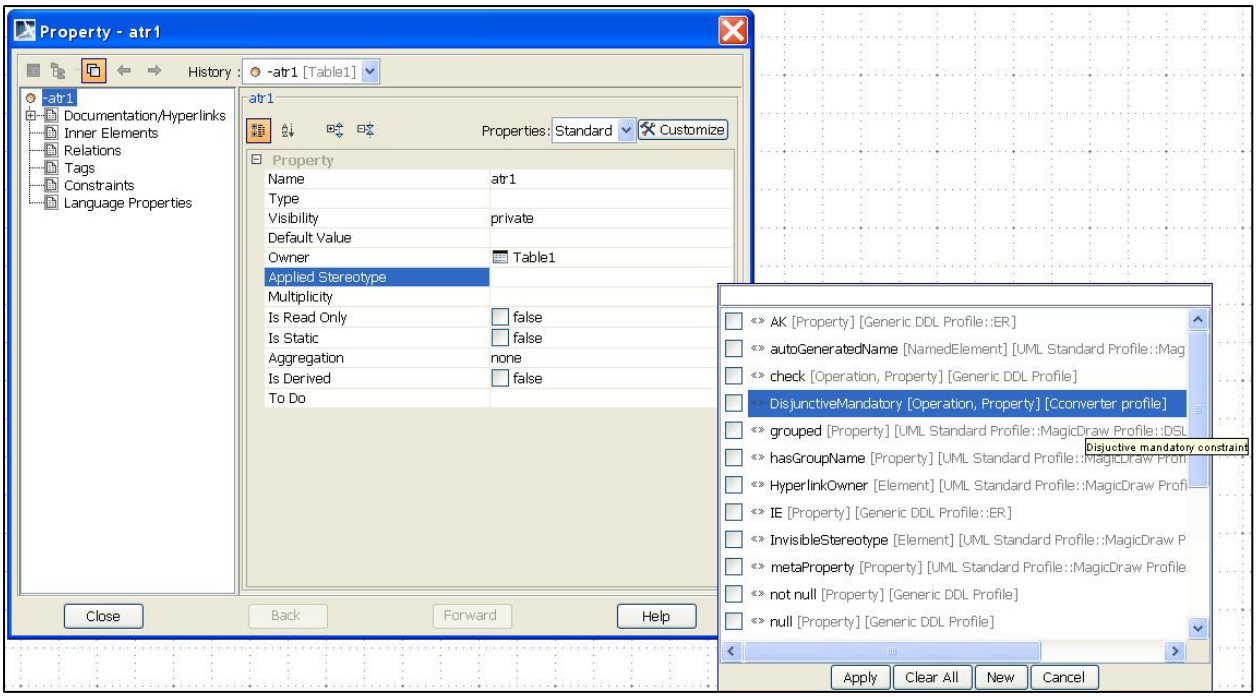


26 pav. *Sistemos išdėstymo diagrama.*

5. Sistemos realizacija

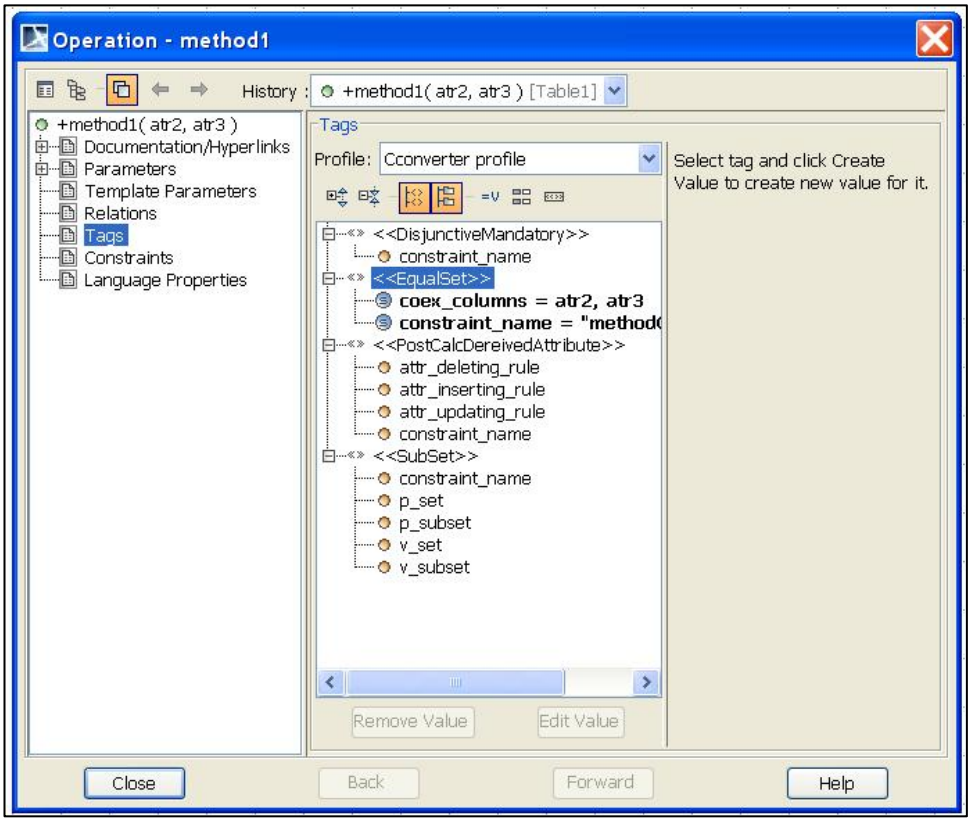
Šiame poskyryje pateikiamas vartotojo sąsajos aprašymas. Aprašomi pagrindiniai vartotojo sąsajos dialogai, pateikiami įskiepio integracijos į MagicDraw programinį paketą paveikslai.

Vartotojo sąsaja pilnai integruota su MagicDraw vartotojo sąsaja, kas palengvina vartotojų apmokymą darbui su magistrinio darbo realizacija. Vientisumo reikalavimas atributui specifikuojamas, iškviečiant standartinį MagicDraw atributo savybių dialogą (27 pav.). Jame vartotojas pasirenka vieną iš vientisumo reikalavimų stereotipų lentelės arba vaizdinio atributui.



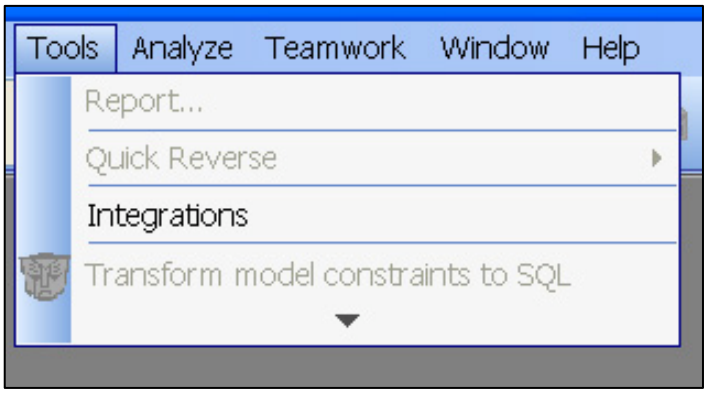
27 pav. *MagicDraw atributo stereotipo specifikacijos dialogas.*

Priskyrus diagramos elementui stereotipą, pasinaudojant elemento savybių dialogu (28 pav.), galima specifikuoti pasirinkto stereotipo žymėtųjų reikšmių reikšmes. Reikšmių tipai ir reikšmės pagal nutylėjimą yra nuskaitomos iš profilio, kuriam priklauso pasirinktas stereotipas.



28 pav. *MagicDraw diagramos elemento stereotipo žymėtųjų reikšmių specifikacijos dialogas.*

Duomenų bazės loginio modelio vientisumo reikalavimų transformacijos įskiepis gali būti iškviestas dviems būdais. Pirmas būdas yra pasinaudojant MagicDraw meniu punktu „Tools/Transform model constraints to SQL“ (29 pav.):



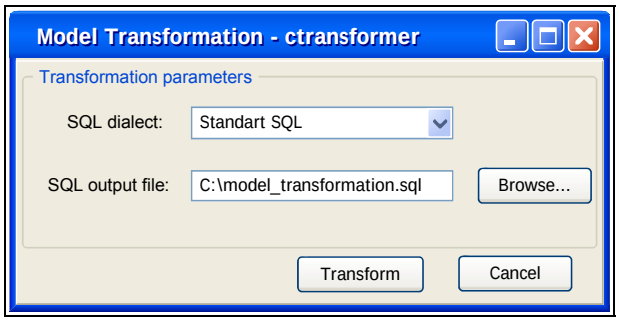
29 pav. *Ctransformer įskiepio išikvietimo meniu punktas*

.

Alternatyviai ctransformer įskiepis gali būti iškviestas naudojant MagicDraw įrankių juostą. Joje įdiegus įskiepį atsiranda *ctransformer* piktograma.

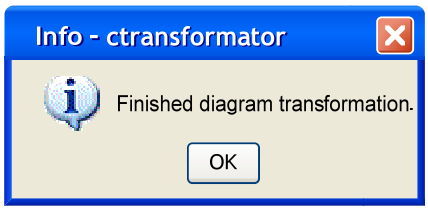
30 pav. *MagicDraw įrankių piktogramų juosta su ctransformer piktograma.*

Iškvietus įskiepį, vartotojui pateikiamas *ctransformer* loginio duomenų bazės modelio dialogas (31 pav.). Jame vartotojas nurodo transformacijos parametrus, t.y. generuojamo SQL dialektą ir transformacijos išeities bylą.



31 pav. *Ctransfomer loginio db modelio transformacijos dialogas.*

Sėkmingai atlikus loginio duomenų bazės modelio transformaciją į SQL kodą, vartotojui apie tai panešama sekančiu dialogu (32 pav.):



32 pav. *Ctransformer db modelio transformacijos patvirtinimo dialogas .*

6. Sistemos testavimas

Šiame poskyryje pateikiama magistrinio darbo medžiaga apie realizacijos testavimo tikslus, apimtį, apribojimus, rezultatus. Skyriaus pabaigoje pateikiamos testavimo išvados. Su testavimo protokolais galima susipažinti magistrinio darbo priede (Priedas nr. 5).

6.1. Testavimo tikslai ir objektai

Testavimo objektas yra ctransformer įskiepis jau egzistuojančiam CASE įrankiui, t.y. firmos NoMagic programiniam paketui MagicDraw. Šis išplėtimas suteikia vartotojui tokias galimybes:

- ✓ Grafiškai pavaizduoti vientisumo apribojimus duomenų bazės modeliuose.
- ✓ Transformuoti duomenų bazės modelius į SQL programinį kodą, kuris galėtų būti panaudotas fizinės duomenų bazės modelio sukūrimui.

Ctransformer yra realizuotas Java programavimo kalbą ir firmos NoMagic OpenAPI klasių biblioteką.

6.2. Testavimo apimtis

Programinės įrangos kokybės užtikrinimui bus naudojamos tokios testavimo rūšys:

- ✓ Vienetų (angl. *unit*).
- ✓ Integravimo (angl. *integration*).
- ✓ Validavimo (angl. *validation*).
- ✓ Aukšto lygmens (angl. *high level*). Šios rūšies testai apjungs sekančias testų rūšis: atsistatymo (angl. *recovery*), saugumo (angl. *security*), stresų (angl. *stress*), našumo (angl. *performance*).

Pagrindinis testavimo dėmesys bus sutelktas į kiekvieno atskiro vientisumo apribojimo transformavimo į programinį kodą teisingumą ir pilnumą.

Ctransformer naudoja jau egzistuojančią MagicDraw grafinę vartotojo sąsają. Testai susiję su grafine sąsaja bus atliekami kaip integravimo testų dalis. Didelis dėmesys bus skiriamas integravimo testams, su tikslu užtikrinti stabilų ctransformer ir MagicDraw programinio paketo veikimą.

6.3. Pagrindiniai reikalavimai

Projekto darbo grupę sudaro du žmonės: programuotojas ir darbo vadovas, kuris kartu yra užsakovas. Didesnę programines įrangos testavimo darbų dalį (iki 90%) atliks programuotojas. Projekto vadovas/užsakovas atliks validavimo testus. Dėl ribotų darbo jėgos

ir laiko resursų teks apriboti negatyvių testų scenarijų skaičių. Bus daroma prielaida, kad su programine įranga dirbs patyręs vartotojas, kuris jau turi darbo su MagicDraw patirtį.

Atliekant vienetų testus, buvo naudojamos JUnit ir Cobertura programines priemones testų scenarijų realizavimui, vykdymui, dokumentavimui. Kitų rūšių testams dokumentuoti buvo naudojama forma, pateikta priede nr. 5.

Aktuali MagicDraw versija yra 14.0. Jau yra išleista 15.0 programos versija. Ryšium su tuo, kad sunku šiuo metu prognozuoti/numatyti, kokie pakeitimai bus duomenų bazių projektavimo priemonėse ir OpenAPI, visi testai buvo atliekami naudojant 14.0 versiją.

Dėl laiko ir resursų trūkumo integracijos testai bus apriboti Fedora Core 6, Windows XP Prof. SP2 operacinėm sistemom ir Sun JDK 1.5 aplinkai.

6.4. Testuojama programinė įranga

Testuojamą programą galima būtų suskirstyti į dvi pagrindines dalis:

- ✓ Reikalavimų profilis ir jo panaudojimas vientisumo apribojimams aprašyti.
- ✓ Modelio transformacija į SQL programinį kodą.

Reikalavimų profilyje, specialiai sukurta kalba, aprašyti vientisumo reikalavimai esybės atributams, esybes operacijoms (operacijų pagalba apibrėžiami reikalavimai, į kurios įeina daugiau nei vienas esybės atributas) ir ryšiams tarp esybių. Viena iš pagrindinių užduočių yra ištestuoti kiekviena apribojimą iš šio profilio, t.y. specifikacijos teisingumas, sintaksės klaidų paieška, vientisumo reikalavimų specifikacijų rinkinio pilnumas.

Modelio transformacijos metu sujungiama informacija apie esybių modelį ir reikalavimų specifikaciją, iš reikalavimų profilio, rezultate gaunamas SQL programinis kodas, kuris privalo atitikti loginį esybių ryšių modelį. Testų metu turės būti ištestuota kiekvieno apribojimo, aprašyto reikalavimų profilyje, transformacija į SQL programinį kodą. Egzistuoja reikalavimai, kuriuos galima priskirti ne tik esybės atributams, bet ir esybės operacijoms. Norint užtikrinti testų pilnumą, teks ištestuoti skirtingus apribojimo priskyrimo būdus.

6.5. Sąsajos

Įskiepis naudoja esamą MagicDraw vartotojo sąsaja reikalavimų profilio užkrovimui, vientisumo reikalavimų ir jų žymėtų reikšmių (angl. *tag*) specifikavimui, todėl ši vartotojo sąsajos dalis neįeina į planuojamų sąsajų testų sąrašą.

Įskiepio grafinę sąsają su vartotoju sudaro vienas grafinis langas. Šiame lange vartotojas pasirenka:

- ✓ SQL dialektą, į kurį bus transformuojamas duomenų bazės loginis modelis.
- ✓ Išeities laikmeną, kurioje bus išsaugotas modelio transformacijos rezultatas.

Dialogo lango iškvietimui vartotojas naudos iš pagrindinio MagicDraw lango ‚Model Transformations-> DDL to SQL' meniu punktą.

6.6. Testavimo strategija

Visų žemiau aprašytų testų strategija kuo efektyviau ir kokybiškiau ištestuoti programinę įrangą. Kur galima, panaudoti testų automatizavimo priemones. Taikyti Java programinės įrangos kūrime jau standartu tapusias kokybės užtikrinimo ir dokumentavimo priemones. Naudoti XP (*angl*. Extreme Programming) principus, kuriant įskiepį, t.y. prieš rašant kodą, suprogramuoti automatinius testus.

Dėl ribotų personalo išteklių, efektyvi ir apgalvota testavimo strategija gali nulemti projekto sėkmingą arba prastą pabaigą.

6.7. Testavimo rezultatų dokumentacija

Testavimo rezultatai buvo saugomi ir dokumentuojami panaudojant automatines JUnit testų dokumentavimo priemones. Testų katalogų ir testų protokolų pavyzdžiai pateikti šio dokumento priede nr. 5.

6.8. Testavimo rezultatai ir išvados

Įvertinus testavimo protokolus (Priedas Nr.5) galima padaryti sekančias išvadas:

- ✓ Kodo padengimas testais sudaro 66%, kas yra 14% mažiau nei buvo planuojama. Įvertinus tai, kad didžiausią neištestuoto kodo dalį sudaro „*set*" ir „*get*" metodai, kurie buvo sugeneruoti automatiniu būdu, tai nėra kritiška. Galima teigti, kad programos ištestavimo lygis yra pakankamas pirmai produkcinei programos versijai.
- ✓ Per mažai ištestuota grafinė įskiepio vartotojo sąsaja. Įvertinus tai, kad grafinė įskiepio vartotojo sąsaja buvo projektuota kaip alternatyva MagicDraw programinio paketo grafinei sąsajai, tai nėra kritiška. Iškilus problemoms, vartotojas lengvai gali pasinaudoti MagicDraw vartotojo sąsaja.
- ✓ Toliau plečiant įskiepio funkcijas, reiktų pagerinti kodo padengimą testais iki 80%. Didesnis kodo padengimas neturi didelės prasmės.

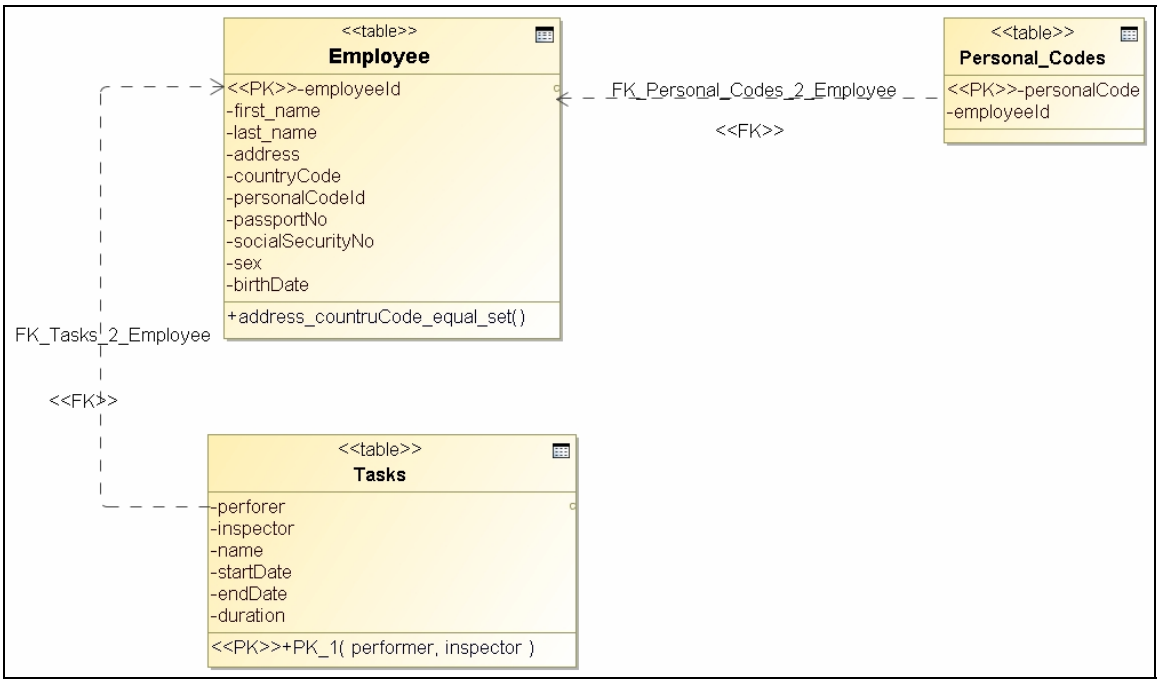
Apibendrinus testavimo išvadas galima teigti, kad programa yra pakankamai ištestuota, kad galėtų būti pateikta vartotojui.

7. Eksperimentinis tyrimas

Magistrinio darbo realizacijos veikimui ir efektyvumui patikrinti buvo atliktas eksperimentas, kurio metu buvo generuojamas SQL kodas apribojimų pavyzdžiams, atliekamas išplėtimas naujais stereotipais ir dialektais bei DB modelio kūrimo greičio nustatymas.

7.1. Vientisumo reikalavimų kodo generavimo pavyzdys

Eksperimentui buvo pasirinktas duomenų bazės loginio modelio fragmentas:



33 pav. *Duomenų bazės loginio modelio fragmentas.*

Šiame modelyje yra susietos trys esybės:

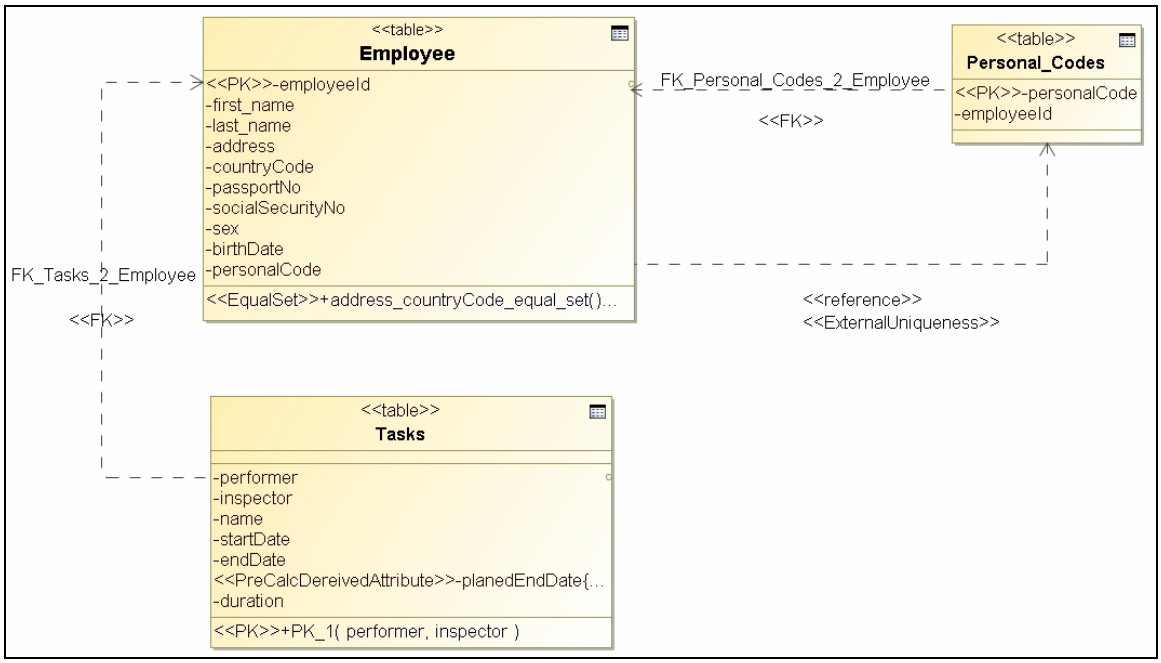
- ✓ *Employee* – saugo informaciją apie darbuotojus.
- ✓ *Tasks* – saugo informaciją apie darbuotojų užduotis ir jų trukmę.
- ✓ *Personal_Codes* – saugo darbuotojų personalinius numerius.

Modelį papildome vientisumo reikalavimais:

- ✓ Ekvivalentiškumo vientisumo reikalavimu *Employee* lentelės stulpeliams *countryCode* ir *address*;

- ✓ Išvestiniu atributo vientisumo reikalavimu *Tasks* lentelės stulpeliui *planedEndDate;*
- ✓ Išorinio unikalumo vientisumo reikalavimą ryšiui tarp lentelių *Employee* ir *Personal_Codes*.

Modifikavus pasirinktą duomenų bazės fragmentą, jis atrodytų taip (34 pav.):



34 pav. *Modifikuotas duomenų bazės loginio modelio fragmentas.*

Pasinaudojant UML2 išplėtimo galimybėmis, įvedami stereotipai su žymėtomis reikšmėmis. Jie detaliai aprašomi sekančiuose poskyriuose.

7.1.1. Išvestinio atributo vientisumo reikalavimas

Išvestinio atributo vientisumo reikalavimą, specifikuojame kaip *<<PreCalcDerivedAttribute>>* stereotipą. Šiam stereotipui specifikuojame žymėtąsias reikšmės:

Lentelė 5.1. 1. <<PreCalcDerivedAttribute>> stereotipo žymėtosios reikšmės

Pavadinimas	Tipas	Aprašymas	Reikšmė
constraint_name	String	VR pavadinimas	employee_equal_constr
attr_calc_rule	String	VR apskaičiavimo taisyklė	planedEndDate := dateadd(dd, duration, startDate)

Dinaminį SQL šabloną šiam vientisumo reikalavimui galima užrašyti taip:

```
CREATE OR REPLACE TRIGGER <tag:constraint_name/>
```

```
 BEFORE INSERT OR UPDATE OF <e:constraint_owner/> ON <e:table_1/>
 for each row
BEGIN
 <tag:attr_calc_rule/>
END <tag:constraint_name/>;
```

Palyginimui OCL kalba vientisumo reikalavimą galima būtų užrašyti taip:

```
context Person
 inv: self.planedEndDate = (self.startDate + self.duration)
```

7.1.2. Išorinio unikalumo vientisumo reikalavimas

Išorinio unikalumo vientisumo reikalavimą specifikuojame kaip <<*ExternalUniquenes*>> stereotipą. Šiam stereotipui specifikuojame žymėtąsias reikšmės:

Lentelė 5.1.2. <<ExternalUniquenes>> stereotipo žymėtosios reikšmės

Pavadinimas	Tipas	Aprašymas	Reikšmė
trigger_1_name	String	Pirmo trigerio pavadinimas	pcode_t_trigger1
trigger_2_name	String	Antro trigerio pavadinimas	employe_trigger_2
view_1_name	String	Pirmo vaizdinio pavadinimas	pc_view1
view_2_name	String	Antro vaizdinio pavadinimas	pc_view2
constraint_attr_1	Property	Pirmas apribotas atributas	Employee.personalCode
constraint_attr_2	Property	Antras apribotas atributas	Personal_Codes.personalCode
table_1_pk	Property {ordered}	Ryšio kliento lentelės PR	Employee.employeId
table_2_pk	Property {ordered}	Ryšio tiekėjo lentelės PR	Personal_Codes.personalCode
table_2_fk	Property {ordered}	Ryšio tiekėjo lentelės IR	Personal_Codes.employeeId
constraint_violation_error_code	Int	VR pažeidimo klaidos kodas. Pagal nutylėjimą lygus: -20107	-20107
constraint_violation_error_msg	String	VR pažeidimo klaidos pranešimas. Pagal nutylėjimą lygus: „External uniqueness constraint is violeted!“	External uniqueness constraint is violeted!

Dinaminį SQL šabloną šiam vientisumo reikalavimui galima užrašyti taip:

```
#7.1. Pirmiausiai sukurimas view is apribojime dalyvaujanciu leneteliu:
#7.1.1. Lentelei table_1

CREATE OR REPLACE VIEW <tag:view_1_name/> AS
SELECT <foreach:column name="table_1" separator=","/> FROM <e:table_1/>;
```

```
#7.1.2  Lentelei table_2

CREATE OR REPLACE VIEW <tag:view_2_name/> AS
SELECT <foreach:column name="table_2" separator=","/> FROM <e:table_2/>;

#7.2. Sukuriamas trigeris atvaizdui
#7.2.1. Sukuriamas trigeris view_1_name atvaizdui

CREATE OR REPLACE TRIGGER <tag:trigger_1_name/>
  INSTEAD OF UPDATE ON <tag:view_1_name/>
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  SELECT count(p.<tag:constraint_attr_2/>) INTO a_Exists
  FROM <e:table_2/> p, <e:table_1/> c
  WHERE p.<tag:table_2_fk/> = c.<tag:table_1_pk/>
  and c.<tag:constraint_attr_1/> = :new.<tag:constraint_attr_1/>
  and p.<tag:constraint_attr_2/> IN (SELECT p.<tag:constraint_attr_2/>
                    FROM <e:table_2/> p, <e:table_1/> c
                    WHERE p.<tag:table_2_fk/> = c.<tag:table_1_pk/>
                    and c.<tag:constraint_attr_1/> = :old.<tag:constraint_attr_1/>);
  IF a_Exists > 0 THEN
    RAISE Invalid_record;
  ELSE
    UPDATE <e:table_1/> c set
    <foreach:column name="table_1" separator=",">
      c.e:element = :new.e:element
    </foreach:column>
    where c.<tag:table_1_pk/> = :old.<tag:table_1_pk/>;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> <tag:constraint_violation_error_code/>,
         msg=> '<tag:constraint_violation_error_msg/>');
END <tag:trigger_1_name/>;

#7.2.2. Sukuriamas triggeris view_2_name

CREATE OR REPLACE TRIGGER <tag:trigger_2_name/>
  INSTEAD OF INSERT OR UPDATE ON <tag:view_2_name/>
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
   SELECT count(temp.rowid) INTO a_Exists
   FROM (SELECT c<tag:constraint_attr_1/>, p.<tag:constraint_attr_2/>, p.<tag:table_2_pk/>
      FROM <e:table_2/> p, <e:table_1/> c
      WHERE p.<tag:table_2_fk/> = c.<tag:table_1_pk/> temp, <e:table_1/> cc
   WHERE temp.<tag:constraint_attr_1/> = cc.<tag:constraint_attr_1/>
   AND cc.<tag:table_1_pk/> = :new.<tag:table_2_fk/>
   AND temp.<tag:constraint_attr_2/> = :new.<tag:constraint_attr_2/>
   AND temp.<tag:table_2_pk/> <> :old.<tag:table_2_pk/>;
   IF a_Exists > 0 THEN
     RAISE Invalid_record;
   ELSIF updating THEN
     update <e:table_2/> p set
     <foreach:column name="table_2" separator=",">
       p.e:element = :new.e:element
     </foreach:column>
     where p.<tag:table_2_pk/> = :old.<tag:table_2_pk/>;
   ELSIF inserting THEN
     INSERT INTO <e:table_2/>(<foreach:column name="table_2" separator=", "/>)
     VALUES(<foreach:column name="table_2" separator=", ">
                 :new.e:element
               </foreach:column>);
```

```
  END IF;
 EXCEPTION
   WHEN Invalid_record THEN
    RAISE_APPLICATION_ERROR (
      num=> <tag:constraint_violation_error_code/>,        //default value = -20107
      msg=> '<tag:constraint_violation_error_msg/>');      //default value = 'External uniqueness constraint is violeted!'
END <tag:trigger_2_name/>;
```

Palyginimui OCL išraiškos šablonas išorinio unikalumo apribojimui objekto A atributui `c1` ir objekto B atributui `c2 [12, 13]:`

```
context A
       inv: self → exists (a1,a2:A|
        a1.B.c1=a2.B.c1 and a1.c2=a2.c2 and c1=c2 and
c1.oclIsKindOf(ExternalUniqueness)
  implies a1=a2)
```

7.1.3. Ekvivalentiškumo vientisumo reikalavimas

Ekvivalentiškumo vientisumo reikalavimą, specifikuojame kaip *<<EqualSet>>* stereotipą. Šio stereotipo žymėtosios reikšmės atrodo taip:

Lentelė 5.1. 3. <<EqualSet>> stereotipo žymėtosios reikšmės

Pavadinimas	Tipas	Aprašymas	Reikšmė
constraint_name	String	VR pavadinimas	employee_equal_constr
coex_columns	Property	Koegzistuojančių stulpelių rinkinys	Employee.countryCode, Employee.address

Dinaminį SQL šabloną šiam vientisumo reikalavimui galima užrašyti taip:

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK ((<foreach:value name="coex_columns"  separator=" AND ">
  e:element IS NOT NULL
</foreach:value>
) OR (
<foreach:value name="coex_columns" separator=" AND ">
  e:element IS NULL
</foreach:value>));
```

Palyginimui OCL išraiškos šablonas ekvivalentiškumo apribojimo objekto A neprivalomų atributų grupei `c1,…,cn [12, 13]`:

```
context A
inv: (self.c1.isUndefined() implies
 self.c2.isUndefined() … and
   self.ci.isUndefined() … and
   self.cn.isUndefined()))
 and
 (self.c2.isUndefined() implies
   self.c1.isUndefined() and
```

```
   Self.c3.isUndefined() … and
   self.ci.isUndefined() … and
   self.cn.isUndefined())
 … and
 (self.ci.isUndefined() implies
   self.c1.isUndefined() and
   Self.ci-1.isUndefined() … and
   self.ci+1.isUndefined() … and
   self.cn.isUndefined())
… and
 (self.cn.isUndefined() implies
   self.c1.isUndefined() and
   self.ci.isUndefined() … and
   self.cn-1.isUndefined())
or
(not(self.c1.isUndefined()) implies
  not(self.c2.isUndefined()) … and
  not(self.ci.isUndefined()) … and
  not(self.cn.isUndefined()))
 and
 (not(self.c2.isUndefined()) implies
   not(self.c1.isUndefined()) and
   not(self.c3.isUndefined()) … and
   not(self.ci.isUndefined()) … and
   not(self.cn.isUndefined()))
 … and
 (not(self.ci.isUndefined()) implies
   not(self.c1.isUndefined()) and
   not(self.ci-1.isUndefined()) …and
   not(self.ci+1.isUndefined()) …and
   not(self.cn.isUndefined()))
… and
 (not(self.cn.isUndefined()) implies
  (not(self.c1.isUndefined()) and
   not(self.ci.isUndefined()) …and
   not(self.cn-1.isUndefined()))
```

Atlikus šio modelio vientisumo reikalavimų transformaciją į SQL kodą, gauname kodą:

```
----- PreCalcDerivedAttribute -------------------------
CREATE OR REPLACE TRIGGER Task_precalc_constr
  BEFORE INSERT OR UPDATE OF planedEndDate ON Tasks
  for each row
BEGIN
  planedEndDate :=  dateadd(dd, duration, startDate)
END Task_precalc_constr;
------------ Equal Set -----------------------------------
ALTER TABLE Employee
ADD CONSTRAINT employee_equal_constr
CHECK ((address IS NOT NULL AND countryCode IS NOT NULL) OR (address IS NULL AND countryCode IS NULL));

------------- External Uniquenes -------------------------
# SQL code for diagram element Class Employee:

ALTER TABLE Employee
ADD CONSTRAINT employee_equal_constr
CHECK ((address IS NOT NULL AND countryCode IS NOT NULL ) OR (address IS NULL AND countryCode IS
NULL));
# SQL code for diagram element Class Tasks:

CREATE OR REPLACE TRIGGER Task_precalc_constr
  BEFORE INSERT OR UPDATE OF planedEndDate ON Tasks
  for each row
BEGIN
  planedEndDate :=  dateadd(dd, duration, startDate)
END Task_precalc_constr;

# SQL code for diagram element Class Personal_Codes:
# SQL code for diagram element Dependency :

#7. Isorinio unikalumo reikalavimas
```

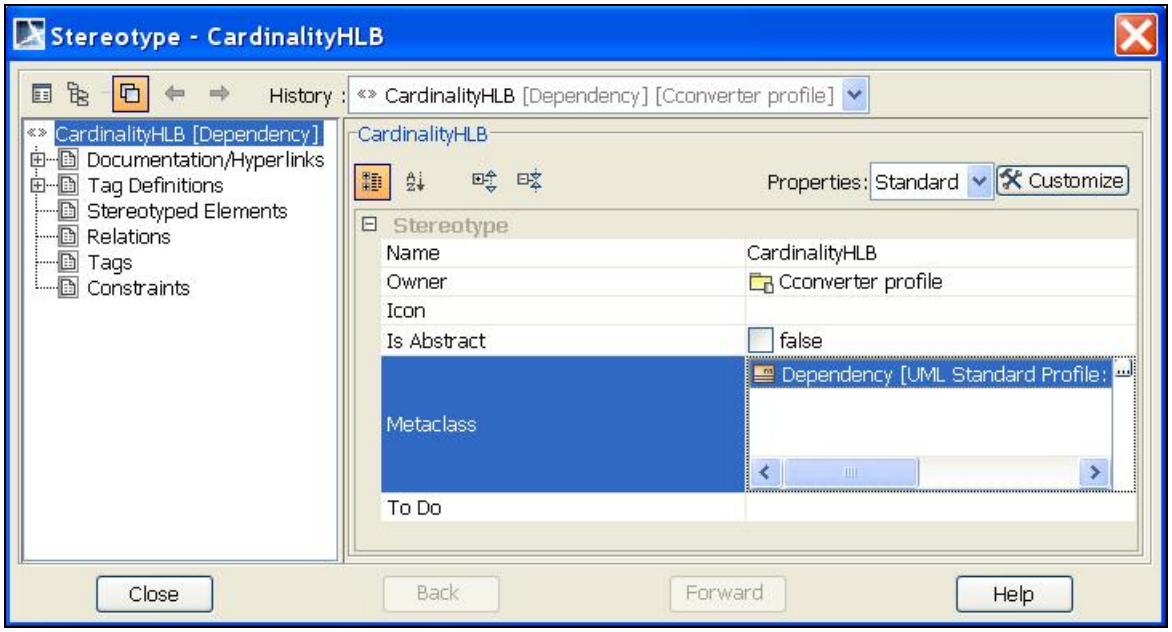
```
#7.1. Pirmiausiai sukurimas view is apribojime dalyvaujanciu leneteliu:
#7.1.1. Lentelei table_1

CREATE OR REPLACE VIEW pc_view1 AS
SELECT sex,personalCode,address,employeeId,socialSecurityNo,countryCode,last_name,birthDate,first_name,passportNo
 FROM Employee;

#7.1.2  Lentelei table_2

CREATE OR REPLACE VIEW pc_view2 AS SELECT personalCode,employeeId FROM Personal_Codes;
#7.2. Sukuriamas trigeris atvaizdui
#7.2.1. Sukuriamas trigeris view_1_name atvaizdui

CREATE OR REPLACE TRIGGER pcode_t_trigger1
  INSTEAD OF UPDATE ON pc_view1
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  SELECT count(p.personalCode) INTO a_Exists
  FROM Personal_Codes p, Employee c
  WHERE p.employeeId = c.employeeId
  and c.personalCode = :new.personalCode
  and p.personalCode IN (SELECT p.personalCode
                  FROM Personal_Codes p, Employee c
                  WHERE p.employeeId = c.employeeId
                  and c.personalCode = :old.personalCode);
  IF a_Exists > 0 THEN
    RAISE Invalid_record;
  ELSE
    UPDATE Employee c set
    c.sex = :new.sex,c.personalCode = :new.personalCode,c.address = :new.address,c.employeeId =
:new.employeeId,c.socialSecurityNo = :new.socialSecurityNo,c.countryCode = :new.countryCode,c.last_name =
:new.last_name,c.birthDate = :new.birthDate,c.first_name = :new.first_name,c.passportNo = :new.passportNo
    where c.employeeId = :old.employeeId;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
      RAISE_APPLICATION_ERROR (
        num=> -20107,
        msg=> 'External uniqueness constraint is violeted!');
END pcode_t_trigger1;

#7.2.2. Sukuriamas triggeris view_2_name

CREATE OR REPLACE TRIGGER employe_trigger_2
  INSTEAD OF INSERT OR UPDATE ON pc_view2
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
   SELECT count(temp.rowid) INTO a_Exists
   FROM (SELECT cpersonalCode, p.personalCode, p.personalCode
      FROM Personal_Codes p, Employee c
      WHERE p.employeeId = c.employeeId temp, Employee cc
   WHERE temp.personalCode = cc.personalCode
   AND cc.employeeId = :new.employeeId
   AND temp.personalCode = :new.personalCode
   AND temp.personalCode <> :old.personalCode;
   IF a_Exists > 0 THEN
     RAISE Invalid_record;
   ELSIF updating THEN
     update Personal_Codes p set
     p.personalCode = :new.personalCode,p.employeeId = :new.employeeId
     where p.personalCode = :old.personalCode;
   ELSIF inserting THEN
     INSERT INTO Personal_Codes(personalCode, employeeId)
     VALUES(:new.personalCode, :new.employeeId);
```

```
 END IF;
EXCEPTION
  WHEN Invalid_record THEN
   RAISE_APPLICATION_ERROR (
     num=> -20107,
     msg=> 'External uniqueness constraint is violeted!');
```

Modelio transformacijai buvo panaudotas ctransformer MagicDraw įskiepis.

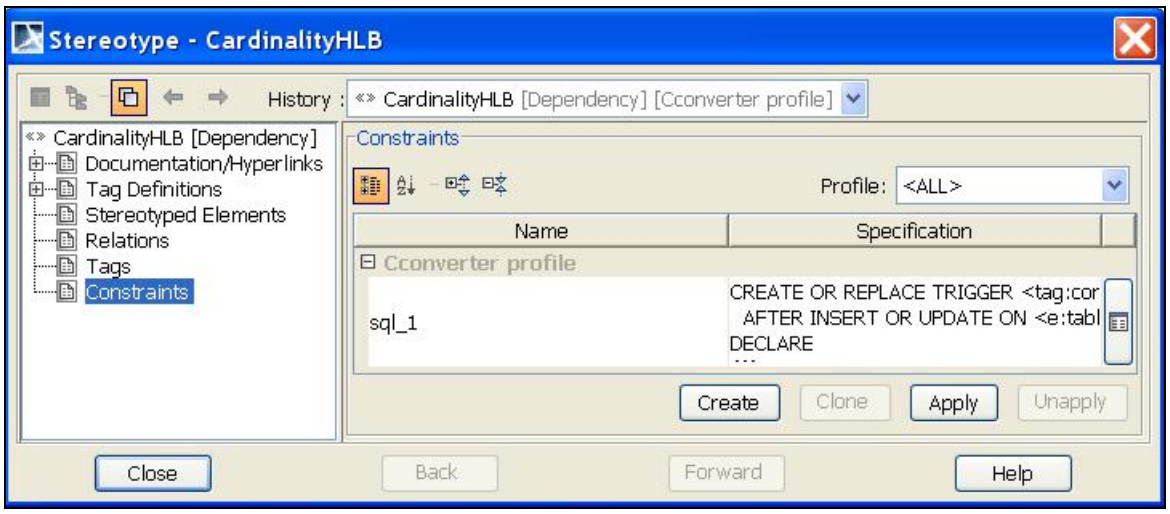
7.2. Sistemos išplėtimais naujais vientisumo reikalavimais

1. MagicDraw pagalba atsidaryti ctransformer reikalavimų profilį.
2. Sukurti naują stereotipą.
3. Apibrėžti stereotipo tipą, t.y. kokiems diagramos elementams jis gali būti taikomas.



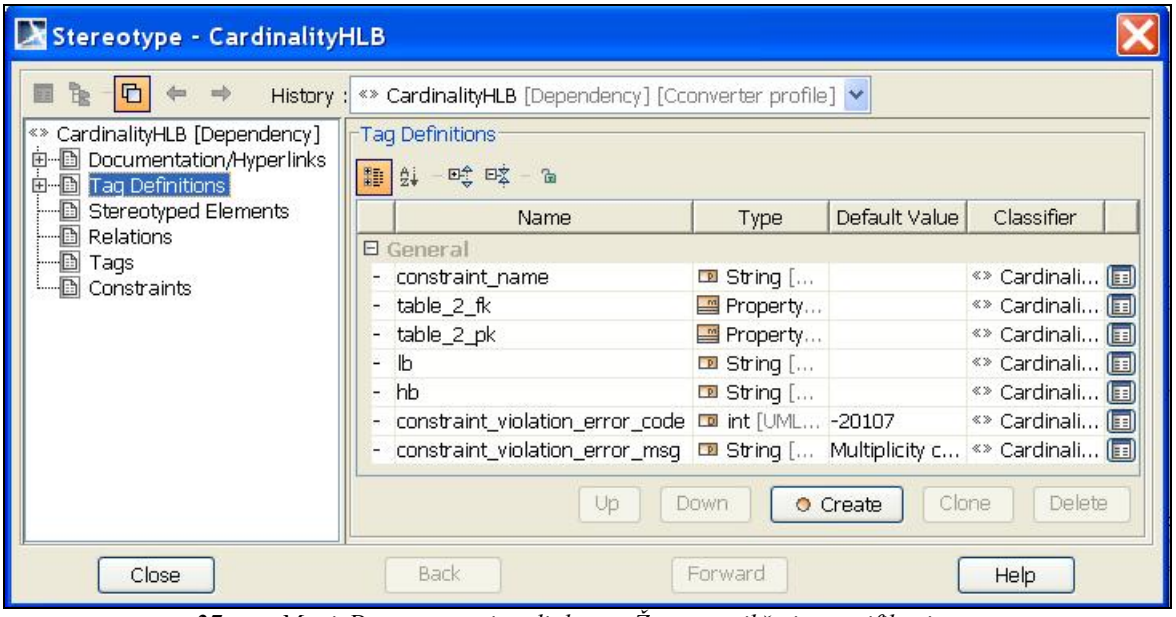
35 pav. *MagicDraw stereotipų dialogas. Metaklasių specifikavimas.*

4. Apibrėžti stereotipo apribojimus ir SQL šablonus:



36 pav. *MagicDraw stereotipų dialogas. Reikalavimų specifikavimas.*

5. Apibrėžti panaudotas žymėtąsias reikšmes.

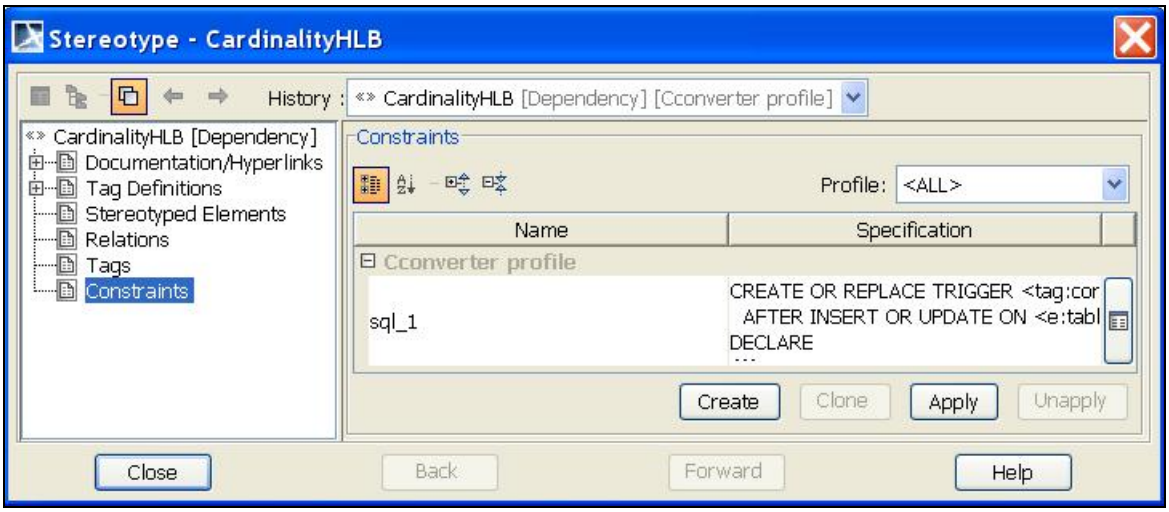


37 pav. *MagicDraw stereotipų dialogas. Žymėtų reikšmių specifikavimas.*

6. Išsaugoti pakeitimus ir perstartuoti MagicDraw programinį paketą.

7.3. Sistemos išplėtimas naujais SQL dialektais

1. *ctransformer* INI faile specifikuoti naujo dialekto pavadinimą.
2. MagicDraw pagalba atsidaryti ctransformer reikalavimų profilį.
3. Kiekvienam stereotipui, naudojant stereotipų dialogą, specifikuoti naują apribojimą (*angl*. constraint) ir SQL šablonus



38 pav. *MagicDraw stereotipų dialogas. Reikalavimų specifikavimas.*

7.4. DB fizinio modelio kūrimo greičio įvertinimas naudojant sistemą

Šiam įvertinimui atlikti buvo sukurta duomenų bazės loginė sistema. Sistemą sudarė 60 lentelių ir vaizdinių. Lenteles ir vaizdiniai buvo sujungti ryšiais. Kievienai lentelei ir ryšiui tarp jų buvo priskyrtas vientisumo reikalavimo stereotipas. Atlikus loginio duomenų bazės modelio transformaciją į „Standart SQL“ dialekto fizinį vientisumo reikalavimų modelį, buvo sugeneruotas 500 eilučių SQL skriptas. Duomenų bazės loginio modelio transformacija užtruko 526ms. Palyginimui tokio pačio SQL skripto nukopijavimas rankiniu būdu užėmė 20 minučių. Atlikus pakartotinę nukopijuoto skripto analizę, buvo aptiktos 2 klaidos (pralesitas simbolis, sukeisti vietom simboliai).

8. Sistemos įvertinimas ir taikymo perspektyvų apibendrinimas

Magistrinio darbo realizacijos įvertinimas buvo atliktas pagal tris kriterijus:

- ✓ Realizuotų vientisumo apribojimų skaičių;
- ✓ Nefunkcinių reikalavimų tenkinimą;
- ✓ Duomenų bazės fizinio modelio kūrimo greitį.

Kiekvienas iš įvertinimo kriterijų yra detaliai išnagrinėtas šio skyriaus poskyriuose.

8.1. Įvertinimas pagal realizuotų vientisumo reikalavimų skaičių

Lentelė 5.4.1. Sistemos palyginimas su kitais CASE ir DBVS įrankiais

	Apribojimo pavadinimas	**Stereotipas**	**Vaizduojamas ir realizuojamas CASE įrankiuose**	**Realizuojamas standartinėmis DBVS priemonėmis**	**Vaizduojamas ir realizuojamas naudojant įskiepį**
1.	Pirminio rakto (atributui)	PK	+	+	–
2.	Pirminio rakto (atributų grupei)		+	+	–
3.	Ryšio	FK	+	+	–
4.	Unikalus indeksas (atributui)	Unique	+	+	–
5.	Unikalus indeksas (atributų grupei))		+	+	–
6.	Atributo reikšmės apribojimas	–	+	+	–
7.	Disjunktyvaus būtinumo (atributui)	DisjunctiveMandatory	–	–	+
8.	Disjunktyvaus būtinumo (ryšiui)		–	–	+
9.	Išorinio unikalumo ryšiui	ExternalUniquenes	–	–	+
10.	Išvestinio atributo[1] (skaičiuojamas prieš įterpimą)	PreCalcDerivedAttribute	–	–	+
11.	Išvestinio atributo (skaičiuojamas po įterpimo)	PostCalcDerivedAttribute	–	–	+
12.	Išvestinio atributo (grupei, skaičiuojamas po įterpimo)		–	–	+
13.	Kardinalumo apribojimas (viršutiniam ir apatiniam rėžiams)	CardinalityHLB	–	–	+

14.	Kardinalumo apribojimas (viršutiniam rėžiui)	CardinalityHB	–	–	+
15.	Apibendrinimo (Disjoint)	DisjoinedSpecification	–	–	+
16.	Apibendrinimo (Complete)	CompleteSpecification	–	–	+
17.	Apibendrinimo (DisjointComplete)	CombinedSpecification	–	–	+
18.	Nerefleksyvaus ryšio	IrreflexiveAssociation	–	–	+
19.	Asimetrinio ryšio	AsymmetricAssociation	–	–	+
20.	Nesimetrinio ryšio	AntisymmetricAssociation	–	–	+
21.	Aciklinio ryšio	AcyclicAssociation	–	–	+
22.	Netranzityvaus	IntransitiveAssociation	–	–	+
23.	Ekvivalentiškumo	EqualSet	–	–	+
24.	Poaibio	SubSet	–	–	+

Įvertinus 5.4.1 lentelėje pateiktus duomenis, galima teigti, kad realizuotas įskiepis 3 kartus lenkia vientisumo reikalavimų tipų skaičių, realizuotą CASE įrankiuose MagicDraw arba Sparx Systems Enterprise Architect. Tokie patys rezultatai lyginant įskiepį su standartinėmis DBVS priemonemis realizuotais vientisumo reikalavimais. Apibendrinus palyginimo rezultatus, gali daryti išvadą, kad naudojant įskiepį kartu su CASE įrankiais galima žymiai pagerinti duomenų modelių su vientisumo reikalavimais vaizdavimą ir realizavimą.

8.2. Nefuncinių reikalavimų tenkinimo įvertinimas

Lentelė 5.4.2. Sistemos nefunkcinių reikalavimų įgyvendinimo įvertinimas

Reikalavimų grupės pavadinimas[1]	Įvertinimas[2] (%)	Komentaras
Reikalavimai sistemos išvaizdai	90	Pranešimai apie klaidas transformuojant db modelį išvedami tik į žurnalinį įskiepio failą.
Reikalavimai panaudojamumui	100	Produkto panaudojamumas nesiskiria nuo MagicDraw koncepcijos ir yra intuityviai suprantamas vartotojams.
Reikalavimai plečiamumui	100	Produktas yra išplėčiamas naujais streotipais ir SQL dialektais nekeičiant išeities kodo. Visi pakeitimai yra atliekami vientisumo apribojimų profilyje ir įskiepio INI konfiguracijos faile.

Reikalavimai sistemos suderinamumui ir realizacijai	80	Sistema nėra tiesiogiai suderinama su JMI ir XMI standartais. Šių standartų palaikymas atliekamas naudoajant MagicDraw programines priemones.
Reikalavimai vykdimo charakteristikoms	100	Įskiepio integracijos testų metu nebuvo pastebėta MagicDraw programinio paketo spartos pablogėjimo. Buvo pastebėtas tik MagicDraw startavimo laiko sulėtėjimas, tačiau jis toks mažas, kad sistemos vartotojui nepastebimas.
Sistemos veikimo sąlygos	100	Įskiepis nekelia jokiu papildomų apribojimų ir jam galioja visi apribojimai, iškelti MagicDraw programiniam paketui.
Reikalavimai sistemo priežiūrai	100	Įskiepis naudoja log4j žurnalinių įrašų rašymo biblioteką. Sistemos klaidos ir galimi jų sprendimo būdai yra pateikti sistemos varototjo atmintinėje.
Bendras įvertinimas	96	Aritmetinis vidurkis

[1] – Detalus nefunkciių reikalavimų grupių aprašymas yra pateiktas 4.1.5 skyriuje „Nefunkciniai reikalavimai".

[2] – Įvertinimo skalė yra nuo 0 iki 100 procentų. 100 procentų reiškia, kad reikalavimų grupė yra pilnai realizuota. 0 procentų reiškia, kad reikalavimus iš duotos grupės buvo neatsižvelgta. Įvertinimai tarp 0 ir 100 procentų reiškia, kad reikalavimų grupė buvo realizuota dalinai. Kuo didesnis įvertinimas, tuo pilnesnis yra duotos vertinamos grupės reikalavimų tenkinimas.

Apibendrinus nefunkcinių reikalavimų tenkinimo rezultatus, galima teigti, kad sukurta sistema pilnai tenkina jai projektavimo fazėje iškeltus nefunkcinius reikalavimus.

8.3. DB fizinio modelio kūrimo greičio įvertinimas naudojant sistemą

Įvertimus eksperimento rezultatus (7.4 poskyris) galima padaryti šias išvadas:

- ✓ Automatizuotas vientumo reikalavimų transformavimą į programinį kodą žymiai paspartina programinės įrangos kūrimo greitį.

- ✓ Automatinis metodas yra atsparesnis klaidoms, kadangi vientisumo apribojimų kodas generuojamas iš jau ištestuotų SQL šablonų.

9. Išvados

1. Atlikta egzistuojančių CASE įrankių analizė, skirta nustatyti jų tinkamumą iškeltai problemai spręsti. Iš penkių analizuojamų įrankių pasirinktas firmos NoMagic MagicDraw įrankis atlikti vientisumo reikalavimų specifikaciją ir transformaciją iš loginio duomenų bazės modelio į programinį SQL kodą.

2. Sukurti vientisumo reikalavimų stereotipai, šablonų užrašymo kalba ir transformavimo algoritmas leidžia įgyvendinti vientisumo reikalavimus apimančio SQL kodo generavimą UML CASE įrankiuose. Tai parodė jų realizacija UML CASE įrankyje MagicDraw.

3. Sukurta vientisumo reikalavimų loginiame duomenų dazių modelyje atvaizdavimo ir transformavimo į programinį SQL kodą programinė realizacija. Šios programinės realizacijos sukūrimas rodo, kad galima automatizuoti informacinių sistemų kūrimą, programiškai įgyvendinant loginio duomenų bazės modelio vientisumo reikalavimų specifikaciją loginiame duomenų bazės modelyje.

4. Galimybės specifikuoti vientisumo reikalavimus loginio modelio stereotipais leidžia sistemos analitikams daugiau laiko ir pastangų skirti dalykinės srities analizei ir vaizdavimui, o duomenų bazių specialistai gali užsiimti SQL šablonų optimizacija.

5. Vientisumo reikalavimus galima specifikuoti OCL kalba, tačiau kol kas SQL kodo generavimas iš OCL nėra efektyvus. Be to, OCL apribojimus užrašyti nėra paprasta, o generavimui SQL šablonai yra išbandyti ir optimizuoti ankstesniuose tyrimuose. Palyginus su OCL, pateikiamas sprendimas yra pranašesnis tuo, kad nekelia aukštų reikalavimų projektuotojams ir atsparesnis programavimo klaidoms.

6. Atliktas tyrimas vientisumo reikalavimų loginiame duomenų bazių modelyje specifikavimui ir transformacijai į programinį kodą ištirti. Ištirtas sukurto ctransformer įskiepio taikymas vientisumo reikalavimų specifikacijos ir transformacijos laiko sąnaudų mažinimui. Tyrimo rezultatai įrodo, kad ctransformer įskiepis sumažina projektuotojo darbo laiko sąnaudas, kuriamas kodas yra atsparesnis klaidoms, vientisumo reikalavimu SQL šablonai pakartotinai panaudojami. Sukurtas įskiepis pilnai tenkina projektavimo fazės metu iškeltus nefunkcinius reikalavimus. Realizuotų vientisumo reikalavimų skaičius 3 kartus (6: 18) lenkia CASE ir RBVS įrankiais realizuojamų vientiumo reikalvimų skaičių. Įskiepis lengvai išplėčiamas naujais vientisumo reikalavimais.

7. Magistro tezių pagrindu buvo paruoštas straipsnis ir skaitomas pranešimas konferencijoje „Informacinės technologijos 2008“.

10. Literatūra

[1] ArgoUML internetinė svetainė [žiūrėta 2008 05 10], prieiga per Interneta <*www.argouml.tigris.org* >

[2] ARMONAS, A.; NEMURAITĖ, L. Pattern based generation of full-fledged relational schemas from UML/OCL models. Information Technology and Control, 2006, Vol. 35, No 1, 27-33.

[3] A UML-based Specification Environment [Žiūrėta 2008 05 10], prieiga per Interneta <*http://www.db.informatik.uni-bremen.de/projects/USE/*>.

[4] BLAŽEVIČ, V.; NEMURAITĖ, L. Duomenų vientisumo reikalavimų įgyvendinimas naudojant SQL kodo generavimo šablonus. Informacinės technologijos – 2008: konferencijos pranešimų medžiaga. Kaunas, 2008, 111-114.

[5] Enterprise Architect internetinė svetainė [žiūrėta 2008 05 10], prieiga per Interneta <*www.sparxsystems.com* >

[6] MagicDraw internetinė svetainė [žiūrėta 2008 05 10], prieiga per Interneta <*http://www.magicdraw.com*>.

[7] Market Share: Application Development Software, Gartner [žiūrėta 2008 05 10], *prieiga Internete* <*http://news.thomasnet.com/fullstory/488096*>.

[8] MyEclipse internetinė svetainė [žiūrėta 2008 05 10], prieiga per Interneta <*www.myeclipseide.com* >

[9] MILIAUSKAITE, E.; NEMURAITĖ, L. Konceptualiojo modelio vientisumo reikalavimų taksonomija. Informacinės technologijos – 2005: konferencijos pranešimų medžiaga. Kaunas, 2005, 552-565.

[10] MILIAUSKAITE, Elita; NEMURAITĖ, Lina. Representation of integrity constraints in conceptual models. Information technology and control, 2005, Vol. 34, No 4, 355-365.

[11] NEMURAITĖ, L.; PARADAUSKAS, B.; STULPINAS, R. Informacinų sistemų schemų konceptualiojo normalizavimo galimybių analizė. Informacijos mokslai. Vilniaus universitetas, 2005, No. 33, 150-159.

[12] PAKALNICKIENĖ, E. Koncepcinis duomenų modeliavimas unifikuota modeliavimo kalba su vientisumo reikalavimais. Daktaro disertacija, Kaunas, KTU, 2008.

[13] PAKALNICKIENĖ, Elita; NEMURAITĖ, Lina. Checking of conceptual models with integrity constraints. Information technology and control, 2007, Vol. 36, No 3, 285 – 294.

[14] PowerDesigner 12.0 Highlights [žiūrėta 2008 05 10], prieiga per Interneta <*http://www.sybase.com/products/modelingmetadata/powerdesigner/highlights*>.

11. Santrauka užsienio (anglų) kalba

UML CASE TOOL EXTENSION FOR GENERATING CODE FOR IMPLEMENTATION OF INTEGRITY REQUIREMENTS

SUMMARY

Fulfillment of data integrity constraint requirements is one of main problems in information system development process. The constraints are implemented at physical system development level. Despite the fact that most of current UML CASE tools allow to specify constraints using OCL language, for the most of system analytics and architects use of OCL is too complicated. Currently SQL code generated from OCL constraint implementation is not effective. In this study we will present alternative approach for constraint requirement implementation using UML2 extension possibilities – profiles and stereotypes with tag values. Also we will present SQL code generation template language and code template based constraint requirement transformation to SQL code algorithm. It was used to extend UML Case tool MagicDraw.

12. Terminų ir santrumpų žodynas

CASE įrankis – (angl. Computer-Aided Software Engineering) įrankis, skirtas programinės įrangos kūrimo automatizavimui.

CIL – (angl. Common Intermediate Language) yra žemiausio žmogaus skaitoma programavimo kalba .NET bibliotekoje.

CORBA – Bendroji Objektų Paraiškos Rezervavimo Architektūra (angl. The Common Object Request Broker Architecture) yra standartas nustatytas OMG, kuris suteikia galimybe programoms parašytoms skirtingomis programavimo kalbomis ir veikiančiom skirtinguose kompiuteriuose tarpusavyje sąveikauti.

DB – duomenų bazė, duomenų saugykla.

DDL – (angl. Data Definition Language) programavimo kalba duomenų specifikacijai. XML schema yra DDL.

EJB – (angl. Enterprise Java Beans) programinės įrangos komponentė.

IS – (angl. Information System) informacijos sistema.

JMI – (angl. Java Metadata Interface) java kalbos meta duomenų sąsaja.

HTML - (Hypertext Markup Language "Hiperteksto žymėjimo kalba") - tai kompiuterine žymėjimo kalba, naudojama pateikti turini internete. Kalba standartizuoja W3 konsorciumas.

Vartotojas – sistemos vartotojas, kuris dirba su sistema.

IDL - (angl. Interface Definition Language) programinės įrangos interfeisu specifikacijos kalba sukurta Lamb, Wulfo and Nestoro Queen's Universitete.

IT – informacinės technologijos.

KTU – Kauno Technologijos Universitetas.

.NET – Microsoft firmos produktų ir technologijų rinkinys.

MDA – (angl. Model Driven Architecture) modeliu paremta architektūra.

MSIL – (angl. Microsoft Intermediate Language) bitinio kodas rinkinys, kurį naudoja Microsoft .NET technologijos programinio kodo vykdymui.

OCL – (angl. Object Constraint Language) objektinė reikalavimų kalba.

OMG – (angl. The Object Management Group) korporacija, sukurta objektiškai orientuotu plačiai paplitusių sistemų standartizacijai, šiuo metu užsiima modeliavimo (programų, sistemų ir verslo procesų) priemonių standartizavimu.

PA – panaudojimo atvejis.

PDF - (angl. Portable Document Format) yra atviras failo formatas skirtas technologiškai neutraliam dvimačio dokumento atvaizdavimui. PDF sukurtas ir tvarkomas korporacijos Adobe System.

RDBVS – Reliacinių duomenų bazių valdymo sistema.

UML – (angl. Unified Modeling Language) suvienyta modeliavimo kalba.

VA – vientisumo reikalavimas.

WSDL – (angl. Web Services Description Language) XML formato kalba Web servisams aprašyti.

XML – (angl. Extensible Markup Language) i yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

XMI – (angl. XML Metadata Interchange) formatas, skirtas keitimuisi UML modeliais tarp CASE įrankių, pagrįstų XML.

13. Priedai

13 .1. Priedas Nr.1. Straipsnis „Duomenų vientisumo reikalavimų įgyvendinimas naudojant SQL kodo generavimo šablonus“

Duomenų vientisumo reikalavimų įgyvendinimas naudojant SQL kodo generavimo šablonus

Valdemar Blaževič, Lina Nemuraitė

Kauno Technologijos Universitetas

Viena iš aktualių informacinių sistemų kūrimo problemų yra duomenų vientisumo reikalavimų užtikrinimas, kuris praktiškai atliekamas fizinės realizacijos lygmenyje. Nors UML CASE įrankiais galima specifikuoti šiuos apribojimus OCL kalba, daugeliui analitikų ir projektuotojų naudoti OCL pernelyg sudėtinga, be to, iš OCL reiškinių sugeneruotas SQL kodas kol kas nėra efektyvus. Šiame straipsnyje pateikiamas alternatyvus būdas vientisumo reikalavimų aprašymui panaudojant UML2 išplėtimo galimybes – įvedant profilius bei stereotipus su žymėtosiomis reikšmėmis. Vientisumo reikalavimams įgyvendinti sukurta SQL kodo generavimo šablonų kalba ir sudarytas šablonais specifikuotų reikalavimų transformacijos į SQL kodą algoritmas, realizuotas išplečiant UML CASE įrankį MagicDraw.

Įvadas

Šiame straipsnyje pateikiama viena iš galimybių automatiškai įgyvendinti duomenų modelių vientisumo reikalavimus ir tuo padidinti esamų UML CASE įrankių efektyvumą. Šiuo metu egzistuojantys UML CASE įrankiai gali sugeneruoti SQL kodą, atsižvelgdami tik į pagrindinius vientisumo reikalavimus, realizuojamus DBVS priemonėmis: pirminius ir išorinius raktus bei unikalius indeksus. Praktikoje vientisumo reikalavimų yra daug, ir ne pagrindinių reikalavimų išpildymas reikalauja didelių pastangų [1], [3]. Dažniausiai šie reikalavimai įgyvendinami fizinės realizacijos metu, o tai prieštarauja modeliais grindžiamos architektūros MDA (angl. *Model Driven Architecture*) principams, pagal kuriuos pagrindinis kūrimas vyksta aukštesnės abstrakcijos (t.y. modelių) lygyje. Tai leidžia padidinti kūrimo efektyvumą (padidinti greitį, pakartotinį naudojimą, plečiamumą) ir išvengti daugelio klaidų, kadangi visi formalizuojami darbai perduodami kompiuteriui.

Vientisumo reikalavimus galima užrašyti OCL kalba, tačiau OCL yra per sudėtinga daugeliui projektuotojų. Be to, SQL kodo generavimas iš OCL kol kas nėra efektyvus [4]. Straipsnyje pateikiamas alternatyvus vientisumo reikalavimus realizuojančio kodo generavimo metodas, paremtas UML išplėtimo mechanizmais ir SQL šablonais.

SQL kodui generuoti sukurta dinaminių SQL išraiškų užrašymo kalba, paremta XML. Ja galima aprašyti loginio duomenų modelio vientisumo reikalavimus ir transformuoti į programos kodą. Transformavimo algoritmas realizuotas kaip UML CASE įrankio *MagicDraw* išplėtimas (įskiepis, angl. *plug-in*) *Ctransformer*. Įskiepio sukurimo teorinis pagrindas yra disertacija [2] bei idėjos, aprašomos [1, 3, 5] straipsniuose. Šio įrankio sukurimas suteikia galimybes kurti programinę įrangą ir geresnės kokybės duomenų bazes [6] naudojant vieną ir tą pačią kalbą (UML) bei tą patį UML CASE įrankį.

Vientisumo reikalavimų metamodelis

Šiuo metu plačiai naudojama unifikuota modeliavimo kalba UML 2.0 skirta objektiniam programinės įrangos projektavimui, bet ne duomenų modeliavimui, todėl neturi kai kurių savybių, reikalingų duomenų bazėms projektuoti. Pagrindinė iš šių savybių yra pirminis raktas. Tačiau UML turi išplėtimo mechanizmus: profilius, stereotipus ir žymėtas reikšmes, leidžiančius pritaikyti ją įvairioms sritims.

Vientisumo reikalavimų, kuriais tikslinga išplėsti šiuo metu *MagicDraw* realizuojamų reikalavimų aibę, sąrašas pateikiamas 1 lentelėje. Sąrašas sudarytas remiantis reikalavimų taksonomija [1, 2, 3] bei atlikta CASE įrankio analize.

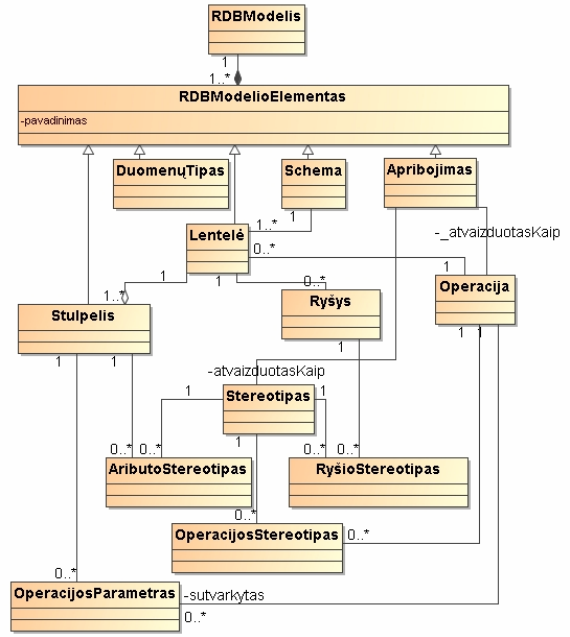
1 lentelė. Papildomų vientisumo reikalavimų sąrašas

Apribojimo pavadinimas	**Žymėjimas**	**Tipas**		
		Atributui	**Atributų grupei**	**Ryšiui**
Disjunktyvaus būtinumo	DisjunctiveMandatory	√	√	
Išorinio unikalumo	ExternalUniquenes			√
Išvestinio atributo (reikšmė skaičiuojama prieš įterpimą)	PreCalcDerivedAttribute	√		
Išvestinio atributo (reikšmė skaičiuojama po įterpimo)	PostCalcDerivedAttribute	√	√	

viršutinei ir apatinei ribai				
Kardinalumo (reikalavimas viršutinei ribai)	CardinalityHB			√
Apibendrinimo (išskaidymo specifikacija)	DisjoinedSpecification			√
Apibendrinimo (visiško apibendrinimo specifikacija)	CompleteSpecification			√
Apibendrinimo (visiško išskaidymo specifikacija)	CombinedSpecification			√
Nerefleksyvaus ryšio	IrreflexiveAssociation			√
Asimetrinio ryšio	AsymmetricAssociation			√
Nesimetrinio ryšio	AntisymmetricAssociation			√
Aciklinio ryšio	AcyclicAssociation			√
Netranzityvaus	IntransitiveAssociation			√
Ekvivalentiškumo	EqualSet		√	
Poaibio	SubSet		√	

Norint UML pavaizduoti duomenų vientisumo reikalavimus, buvo sudarytas šių reikalavimų metamodelis ir sukurtas profilis, kuriame vientisumo reikalavimai pavaizduotas stereotipais su žymėtosiomis reikšmėmis. Vientisumo reikalavimus realizuojantis SQL kodas specifikuojamas kaip atitinkamo stereotipo reikalavimas. Reikalavimų skaičius vienam stereotipui yra neribotas, todėl kiekvienas reikalavimas gali saugoti SQL šabloną skirtingiems SQL dialektams.

Loginio duomenų modelio elementai yra schemos, lentelės, stulpeliai (atributai), ryšiai (išorinių raktų stulpeliai arba ryšio atributai), reikalavimai (aprašantys vientisumo reikalavimus vienam atributui), operacijos (aprašančios vientisumo reikalavimus keliems atributams). Operacijų parametrai vaizduoja stulpelius, kuriems galioja atitinkamas vientisumo reikalavimas. Metamodelis pateikiamas 1 paveiksle.



39 pav. Duomenų loginio modelio UML metamodelis

Dinaminė SQL šablonų specifikavimo kalba

Straipsnyje pateikiama dinaminių SQL šablonų specifikavimo kalba, paremta XML žymėtų reikšmių užrašymo taisyklėmis. Šią kalbą sudaro:

- *„e“* konstantos:

<*e:table_1*/> – lentelės ar vaizdinio, kuriam priskirtas vientisumo reikalavimo stereotipas, pavadinimas;

<*e:table_2*/> – lentelės ar vaizdinio, su kuriuo sujungta vientisumo reikalavimo <*e:table_1*/> lentelė ar vaizdinys, pavadinimas;

<*e:constraint_owner*/> – stulpelio, metodo, lentelės ar vaizdinio, kuriam priskirtas reikalavimo stereotipas, pavadinimas;

<*e:element*> – ciklo *foreach* iteracijoje naudojamo elemento pažymėjimas.

- *„tag“* – žymėtosios reikšmės:

<*tag:“žymėtos reikšmės vardas“/*> – stereotipo žymėtosios reikšmės vardas.

- ***„foreach“*** ciklas, užrašomas pagal šabloną:

<foreach:[constraint_arg|column|value] [name="[table_1|table_2]|[tag'o pavadinimas]"]
[separator="skirtukas"],
[separated="[skirtukas, kuriuo išskirtos žymėtų reikšmių sąrašas]"]
[including_self="true|false" (pagal nutylėjimą = true)]>
e:element
</foreach:[constraint_arg|column|value]>

Išvestinio atributo vientisumo reikalavimas šia kalba užrašomas kaip <<*PreCalcDerivedAttribute*>> stereotipas. Jam specifikuojamos žymėtosios reikšmės pateikiamos 2 lentelėje.

2 lentelė. <<*PreCalcDerivedAttribute*>> stereotipo žymėtosios reikšmės

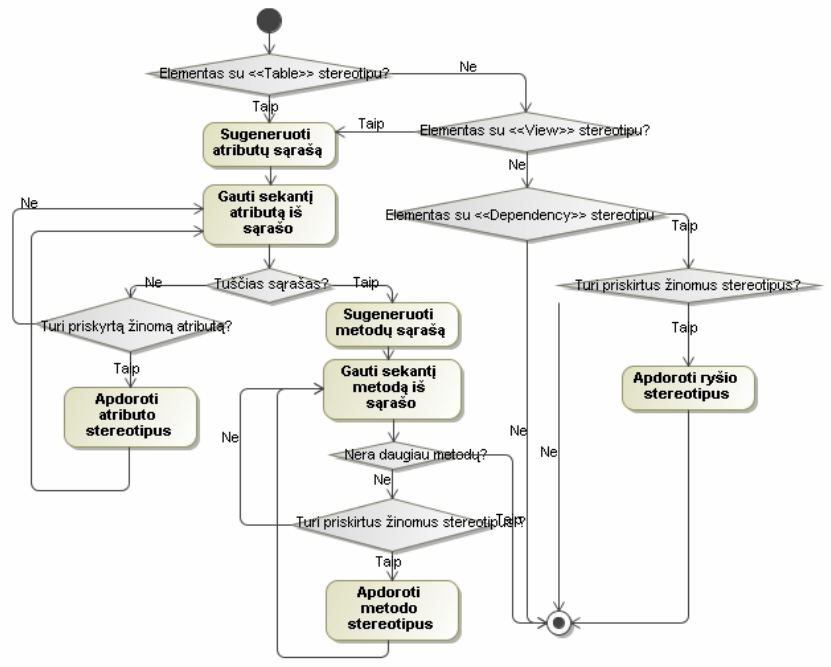
Pavadinimas	Tipas	Aprašymas	Reikšmė
Constraint_name	*String*	Vientisumo reikalavimo pavadinimas	*employee_equal_constr*
Attr_calc_rule	*String*	Vientisumo reikalavimo skaičiavimo taisyklė	*planedEndDate := dateadd(dd, duration, startDate)*

Šio vientisumo reikalavimo SQL šablonas užrašomas taip:

CREATE OR REPLACE TRIGGER <tag:constraint_name/>
BEFORE INSERT OR UPDATE OF <e:constraint_owner/> ON <e:table_1/>
for each row
BEGIN
<tag:attr_calc_rule/>
END <tag:constraint_name/>

Modelio transformavimo algoritmas

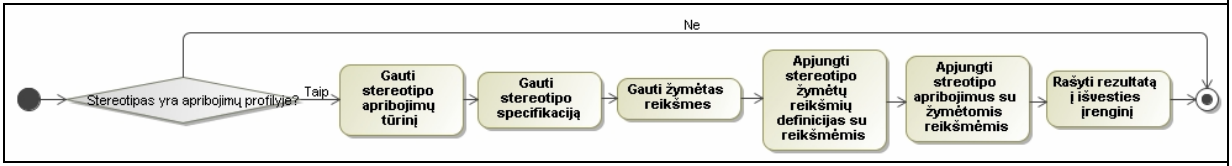
MagicDraw programiniame pakete duomenų bazės loginio modelio atvaizdavimui naudojama UML klasių diagrama. Duomenų modelio klasių (DDL) diagramą sudaro elementai, kurie pagal priskirtus stereotipus skirstomi į lenteles (klasės su <<*Table*>> stereotipu), vaizdinius (klasės su <<*View*>> stereotipu), ir ryšius – (klasių priklausomybes su <<*Dependency*>> stereotipu). Naudojant *MagicDraw* išplėtimo Java klasių biblioteką *OpenAPI*, galima nuskaityti norimą duomenų modelį vaizduojančių elementų sąrašą ir kiekvieną iš jų apdoroti pagal 2 paveiksle pateikiamą algoritmą.



40 pav. Duomenų modelio diagramos vaizdavimo elemento apdorojimas

Duomenų bazės modelio transformavimui patogiausia naudoti lankytojo (*angl. Visitor*) programavimo šabloną. Tokiu būdų užtikrinamas visų diagramos elementų dinaminis apdorojimas. Transformuojant duomenų

modelio elementus, iš vientisumo reikalavimų profilio nuskaitomos apdorojamo stereotipo žymėtų reikšmių specifikacijos (vardai, tipai, pradinės reikšmės) ir reikalavimai – SQL kodo generavimo šablonai. Kitame stereotipo apdorojimo žingsnyje iš loginio duomenų modelio nuskaitomos stereotipų žymių reikšmės. Žymėtosios reikšmės SQL kodo šablone pakeičiamos reikšmėmis iš loginio modelio ir vientisumo reikalavimų profilio. Gautas SQL kodas išvedamas į išvesties įrenginį. Šis algoritmas pavaizduotas 3 paveiksle.



41 pav. Duomenų modelio elemento stereotipo apdorojimas

Pagal pateiktą algoritmą apdorojus visus diagramos stereotipus, sukuriamas SQL kodas, kuriuo papildomas egzistuojančio *MagicDraw* SQL generavimo įrankio sugeneruojamas kodas.

Išvados

Sukurti vientisumo reikalavimų stereotipai, šablonų užrašymo kalba ir transformavimo algoritmas leidžia įgyvendinti vientisumo reikalavimus apimančio SQL kodo generavimą UML CASE įrankiuose. Tai parodė jų realizacija UML CASE įrankyje Magic Draw.

Galimybės specifikuoti vientisumo reikalavimus loginio modelio stereotipais leidžia sistemos analitikams daugiau laiko ir pastangų skirti dalykinės srities analizei ir vaizdavimui, o duomenų bazių specialistai gali užsiimti SQL šablonų optimizacija.

Vientisumo reikalavimus galima specifikuoti OCL kalba, tačiau kol kas SQL kodo generavimas iš OCL nėra efektyvus. Be to, OCL apribojimus užrašyti nėra paprasta, o generavimui SQL šablonai yra išbandyti ir optimizuoti ankstesniuose tyrimuose. Palyginus su OCL, pateikiamas sprendimas yra pranašesnis tuo, kad nekelia aukštų reikalavimų projektuotojams ir atsparesnis programavimo klaidoms.

Literatūra

[1] Miliauskaite E., Nemuraitė L. Konceptualiojo modelio vientisumo reikalavimų taksonomija. Informacinės technologijos – 2005: konferencijos pranešimų medžiaga. Kaunas, 2005, 552-565.

[2] Pakalnickienė, E. Koncepcinis duomenų modeliavimas unifikuota modeliavimo kalba su vientisumo reikalavimais. Daktaro disertacija, Kaunas, KTU, 2008.

[3] Miliauskaitė, Elita; Nemuraitė, Lina. Representation of integrity constraints in conceptual models. Information technology and control, 2005, Vol. 34, No 4, 355-365.

[4] Armonas A., Nemuraitė L. Pattern based generation of full-fledged relational schemas from UML/OCL models. Information Technology and Control, 2006, Vol. 35, No 1, 27-33.

[5] Pakalnickienė, Elita; Nemuraitė, Lina. Checking of conceptual models with integrity constraints. Information technology and control, 2007, Vol. 36, No 3, 285 – 294.

[6] Nemuraitė L., Paradauskas B., Stulpinas R. Informacinų sistemų schemų konceptualiojo normalizavimo galimybių analizė. Informacijos mokslai. Vilniaus universitetas, 2005, No. 33, 150-159.

IMPLEMENTATION OF DATA INTEGRITY REQUIREMENTS USING SQL CODE TEMPLATES

Fulfilment of data integrity requirements is one of main problems in information system development process as the majority of these requirements is implemented only at physical level. In this article we present approach for integrity requirement implementation using UML2 profiles, stereotypes and tagged values. We have created XML-based language for SQL code generation templates and algorithm for transformation of stereotyped UML data model to SQL code. The algorithm was implemented as plug-in for UML Case tool MagicDraw.

13.2. Priedas Nr.2. Vientisumo reikalavimų šablonai

DisjunctiveMandatory apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK (
<foreach:constraint_arg separator=" OR\n">
  e:element IS NOT NULL
</foreach:constraint_arg>);
```

ExternalUniquenes apribojimo šablonas

```
#7.1. Pirmiausiai sukurimas view is apribojime dalyvaujanciu leneteliu:
#7.1.1. Lentelei table_1

CREATE OR REPLACE VIEW <tag:view_1_name/> AS
SELECT <foreach:column name="table_1" separator=","/> FROM <e:table_1/>;

#7.1.2  Lentelei table_2

CREATE OR REPLACE VIEW <tag:view_2_name/> AS
SELECT <foreach:column name="table_2" separator=","/> FROM <e:table_2/>;

#7.2. Sukuriamas trigeris atvaizdui

#7.2.1. Sukuriamas trigeris view_1_name atvaizdui

CREATE OR REPLACE TRIGGER <tag:trigger_1_name/>
  INSTEAD OF UPDATE ON <tag:view_1_name/>
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
  SELECT count(p.<tag:constraint_attr_2/>) INTO a_Exists
  FROM <e:table_2/> p, <e:table_1/> c
  WHERE p.<tag:table_2_fk/> = c.<tag:table_1_pk/>
  and c.<tag:constraint_attr_1/> = :new.<tag:constraint_attr_1/>
  and p.<tag:constraint_attr_2/> IN (SELECT p.<tag:constraint_attr_2/>
                  FROM <e:table_2/> p, <e:table_1/> c
                  WHERE p.<tag:table_2_fk/> = c.<tag:table_1_pk/>
                  and c.<tag:constraint_attr_1/> = :old.<tag:constraint_attr_1/>);
  IF a_Exists > 0 THEN
    RAISE Invalid_record;
  ELSE
    UPDATE <e:table_1/> c set
    <foreach:column name="table_1" separator=",">
     c.e:element = :new.e:element
    </foreach:column>
    where c.<tag:table_1_pk/> = :old.<tag:table_1_pk/>;
  END IF;
  EXCEPTION
     WHEN Invalid_record THEN
      RAISE_APPLICATION_ERROR (
        num=> <tag:constraint_violation_error_code/>,          //default value = -20107
        msg=> '<tag:constraint_violation_error_msg/>');        //default value = 'External uniqueness constraint is violeted!'
END <tag:trigger_1_name/>;

#7.2.2. Sukuriamas triggeris view_2_name

CREATE OR REPLACE TRIGGER <tag:trigger_2_name/>
  INSTEAD OF INSERT OR UPDATE ON <tag:view_2_name/>
  FOR EACH ROW
DECLARE
  a_Exists NUMBER := 0;
  Invalid_record EXCEPTION;
BEGIN
   SELECT count(temp.rowid) INTO a_Exists
```

```
  FROM (SELECT c<tag:constraint_attr_1/>, p.<tag:constraint_attr_2/>, p.<tag:table_2_pk/>
    FROM <e:table_2/> p, <e:table_1/> c
    WHERE p.<tag:table_2_fk/> = c.<tag:table_1_pk/> temp, <e:table_1/> cc
  WHERE temp.<tag:constraint_attr_1/> = cc.<tag:constraint_attr_1/>
  AND cc.<tag:table_1_pk/> = :new.<tag:table_2_fk/>
  AND temp.<tag:constraint_attr_2/> = :new.<tag:constraint_attr_2/>
  AND temp.<tag:table_2_pk/> <> :old.<tag:table_2_pk/>;
  IF a_Exists > 0 THEN
    RAISE Invalid_record;
  ELSIF updating THEN
    update <e:table_2/> p set
    <foreach:column name="table_2" separator=",">
      p.e:element = :new.e:element
    </foreach:column>
    where p.<tag:table_2_pk/> = :old.<tag:table_2_pk/>;
  ELSIF inserting THEN
    INSERT INTO <e:table_2/>(<foreach:column name="table_2" separator=", "/>)
    VALUES(<foreach:column name="table_2" separator=", ">
                :new.e:element
              </foreach:column>);
  END IF;
 EXCEPTION
   WHEN Invalid_record THEN
    RAISE_APPLICATION_ERROR (
      num=> <tag:constraint_violation_error_code/>,        //default value = -20107
      msg=> '<tag:constraint_violation_error_msg/>');      //default value = 'External uniqueness constraint is violeted!'
END <tag:trigger_2_name/>;
```

PreCalcDereivedAttribute apribojimo šablonas

```
CREATE OR REPLACE TRIGGER <tag:constraint_name/>
 BEFORE INSERT OR UPDATE OF <e:constraint_owner/> ON <e:table_1/>
 for each row
BEGIN
 <tag:attr_calc_rule/>                                    //“Išvedimo taisyklė užrašyta PL/SQL kalba”
END <tag:constraint_name/>;
```

PostCalcDereivedAttribute apribojimo šablonas

```
CREATE OR REPLACE TRIGGER <tag:constraint_name/>
 AFTER INSERT OR UPDATE of <foreach:constraint_arg, separator=", "/> OR DELETE ON <e:table_1/>
 for each row
BEGIN
 IF INSERTING THEN
    <tag:attr_inserting_rule/>          //“Išvedimo taisyklė užrašyta PL/SQL kalba”
 ELSIF UPDATING THEN
      <tag:attr_updating_rule/>        //“Išvedimo taisyklė užrašyta PL/SQL kalba”
 ELSIF DELETING THEN
    <tag:attr_deleting_rule/>          //“Išvedimo taisyklė užrašyta PL/SQL kalba”
 END IF;
END <tag:constraint_name/>;
```

CardinalityHLB apribojimo šablonas

```
CREATE OR REPLACE TRIGGER  <tag:constraint_name/>
 INSTEAD OF INSERT OR UPDATE on <tag:virtual_table_name/>
 for each row
DECLARE
 a_Count NUMBER := 0;
 Invalid_record EXCEPTION;
BEGIN
 select count(<tag:table_2_fk/>) into SK
 from <e:table_2/>
 where <e:table_2/>.<tag:table_2_fk/> = :new.<tag:table_2_fk/>;
 If :new.<tag:table_2_fk/> <> :old.<tag:table_2_fk/> and a_Count >= <tag:hb/> Then
   RAISE Invalid_record;
 ELSE
  IF INSERTING THEN
    INSERT INTO <e:table_2/>(<tag:table_2_fk/>, <tag:table_2_pk/>)
    VALUES (:new.<tag:table_2_fk/>, :new.<tag:table_2_pk/>);
  ELSIF UPDATING THEN
```

```
  UPDATE <e:table_2/> SET
  b.<tag:table_2_fk/> = :new.<tag:table_2_fk/>
  WHERE B.<tag:table_2_pk/> = :new.<tag:table_2_pk/>;
 END IF;
END IF;
 EXCEPTION
  WHEN Invalid_record THEN
   RAISE_APPLICATION_ERROR (
      num=> <tag:constraint_violation_error_code/>,      //default value = -20107
      msg=> '<tag:constraint_violation_error_msg/>');     //default value = 'Multiplicity constraint is violeted!'
END  <tag:constraint_name/>;
```

DisjoinedSpecification apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK ((<tag:column_1/> IS NOT NULL AND <tag:column_2/> IS NULL) OR
(<tag:column_1/> IS NULL AND <tag:column_2/> IS NOT NULL) OR
(<tag:column_1/> IS NULL AND <tag:column_2/> IS NULL));
```

CompleteSpecification apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK (<tag:column_1/> IS NOT NULL OR < tag:column_2> IS NOT NULL);
```

CombinedSpecification apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK ((<tag:column_1/> IS NOT NULL AND <tag:column_2/> IS NULL) OR
(<tag:column_1/> IS NULL AND <tag:column_2/> IS NOT NULL));
```

IrreflexiveAssociation apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK (<tag:column_1/> <> <tag:column_2/>);
```

AsymmetricAssociation apribojimo šablonas

```
#19.1. Paketo sukurimas

create or replace package <tag:package_name/>
as
  type ridArray is table of rowid index by binary_integer;
  newRows ridArray;
  empty   ridArray;
end;

#19.2. Pirmas transakcijos trigeris

create or replace trigger <tag:trigger_1_name/>
before insert or update on <e:table_1/>
begin
  <tag:package_name/>.newRows := <tag:package_name/>.empty;
end;

#19.3. Antras transakcijos trigeris

create or replace trigger <tag:trigger_2_name/>
after insert or update on <e:table_1/>
for each row
begin
   <tag:package_name/>.newRows( <tag:package_name/>.newRows.count+1 ) := :new.rowid;
end;

#19.4. Trecias transakcijos trigeris
```

```
create or replace trigger <tag:trigger_3_name/>
after insert or update on <e:table_1/>
declare
  a_raw A_RELATIONSHIP %ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. <tag:package_name/>.newRows.count loop
    select t.* into a_raw
    from <e:table_1/> t
    where t.rowid = <tag:package_name/>.newRows(i);

    select a_raw.<tag:parent_id/> into a_raw.<tag:parent_id/>
    from dual
    where a_raw.<tag:parent_id/> IN
    (select t.<tag:child_id/>
     from (select tr.<tag:child_id/>, tr.<tag:parent_id/>
           from <e:table_1/> tr
           minus
           select a_raw.<tag:child_id/>, a_raw.<tag:parent_id/> from dual) t
     start with t.<tag:parent_id/> = a_raw.<tag:child_id/>
     connect by prior t.<tag:child_id/> = t.<tag:parent_id/>);

    if a_raw.<tag:parent_id/> is not null then

      RAISE Invalid_record;

    end if;
  end loop;

  EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> <tag:constraint_violation_error_code/>,        //default value = -20107
         msg=> '<tag:constraint_violation_error_msg/>');      //default value = 'Acyclic constraint violated!'
     WHEN no_data_found THEN null;
end;
```

AntisymmetricAssociation apribojimo šablonas

```
#18.1. Paketo sukurimas

create or replace package <tag:package_name/>
as
   type ridArray is table of rowid index by binary_integer;
   newRows ridArray;
   empty   ridArray;
end;

#18.2. Pirmas transakcijos trigeris

create or replace trigger <tag:trigger_1_name/>
before insert or update on <e:table_1/>
begin
   <tag:package_name/>.newRows := <tag:package_name/>.empty;
end;

#18.3. Antras transakcijos trigeris

create or replace trigger <tag:trigger_2_name/>
after insert or update on <e:table_1/>
for each row
begin
    <tag:package_name/>.newRows( <tag:package_name/>.newRows.count+1 ) := :new.rowid;
end;

#18.4 Trecias transakcijos trigeris

create or replace trigger <tag:trigger_3_name/>
after insert or update on <e:table_1/>
declare
```

```
  a_raw <e:table_1/> %ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. <tag:package_name/>.newRows.count loop
    select t.* into a_raw
    from <e:table_1/> t
    where t.rowid = <tag:package_name/>.newRows(i);

    select t.<tag:child_id/>, t.<tag:parent_id/> into a_raw
    from <e:table_1/> t
    where t.<tag:child_id/> = a_raw.<tag:child_id/>
    and t.<tag:parent_id/> = a_raw.<tag:parent_id/>;

    if a_raw.<tag:child_id/> is not null and (a_raw.<tag:child_id/> <> b_raw.<tag:parent_id/>) then
      RAISE Invalid_record;
     end if;
  end loop;
   EXCEPTION
     WHEN Invalid_record THEN
       RAISE_APPLICATION_ERROR (
         num=> <tag:constraint_violation_error_code/>,      //default value = -20107
         msg=> '<tag:constraint_violation_error_msg/>');    //default value = 'Antisymmetric constraint violated!'
      WHEN no_data_found THEN null;
end;
```

AcyclicAssociation apribojimo šablonas

```
#19.1. Paketo sukurimas

create or replace package <tag:package_name/>
as
   type ridArray is table of rowid index by binary_integer;
   newRows ridArray;
   empty   ridArray;
end;

#19.2. Pirmas transakcijos trigeris

create or replace trigger <tag:trigger_1_name/>
before insert or update on <e:table_1/>
begin
   <tag:package_name/>.newRows := <tag:package_name/>.empty;
end;

#19.3. Antras transakcijos trigeris

create or replace trigger <tag:trigger_2_name/>
after insert or update on <e:table_1/>
for each row
begin
    <tag:package_name/>.newRows( <tag:package_name/>.newRows.count+1 ) := :new.rowid;
end;

#19.4. Trecias transakcijos trigeris

create or replace trigger <tag:trigger_3_name/>
after insert or update on <e:table_1/>
declare
  a_raw A_RELATIONSHIP %ROWTYPE;
  invalid_record exception;
begin
  for i in 1 .. <tag:package_name/>.newRows.count loop
    select t.* into a_raw
    from <e:table_1/> t
    where t.rowid = <tag:package_name/>.newRows(i);

    select a_raw.<tag:parent_id/> into a_raw.<tag:parent_id/>
    from dual
    where a_raw.<tag:parent_id/> IN
    (select t.<tag:child_id/>
```

```
    from (select tr.<tag:child_id/>, tr.<tag:parent_id/>
          from <e:table_1/> tr
          minus
          select a_raw.<tag:child_id/>, a_raw.<tag:parent_id/> from dual) t
    start with t.<tag:parent_id/> = a_raw.<tag:child_id/>
    connect by prior t.<tag:child_id/> = t.<tag:parent_id/>);

   if a_raw.<tag:parent_id/> is not null then

     RAISE Invalid_record;

   end if;
  end loop;

  EXCEPTION
    WHEN Invalid_record THEN
      RAISE_APPLICATION_ERROR (
        num=> <tag:constraint_violation_error_code/>,        //default value = -20107
        msg=> '<tag:constraint_violation_error_msg/>');      //default value = 'Acyclic constraint violated!'
    WHEN no_data_found THEN null;
end;
```

IntransitiveAssociation apribojimo šablonas

```
#20.1. Paketo sukurimas
create or replace package <tag:package_name/>
as
   type ridArray is table of rowid index by binary_integer;
   newRows ridArray;
   empty   ridArray;
end;

#20.2. Pirmas transakcijos trigeris
create or replace trigger <tag:trigger_1_name/>
before insert or update on <e:table_1/>
begin
   <tag:package_name/>.newRows := <tag:package_name/>.empty;
end;

#20.3. Antras transakcijos trigeris
create or replace trigger <tag:trigger_2_name/>
after insert or update on <e:table_1/>
for each row
begin
    <tag:package_name/>.newRows( <tag:package_name/>.newRows.count+1 ) := :new.rowid;


CREATE or REPLACE trigger <tag:trigger_3_name/>
AFTER INSERT or update on <e:table_1/>
DECLARE
  a_raw <e:table_1/> %ROWTYPE;
  invalid_record exception;
BEGIN
  for i in 1 ..  <tag:package_name/>.newRows.count loop
    select t.* into a_raw
    from <e:table_1/> t
    where t.rowid =  <tag:package_name/>.newRows(i);

    select a_raw.<tag:parent_id/> into a_raw.<tag:parent_id/>
    from dual
    where a_raw.<tag:child_id/> IN
    (select t.<tag:child_id/>
     from (select tr.<tag:child_id/>, tr.<tag:parent_id/>
           from <e:table_1/> tr
           minus
           select a_raw.<tag:child_id/>, a_raw.<tag:parent_id/> from dual) t
     start with t.<tag:parent_id/> = a_raw.<tag:parent_id/>
     connect by prior t.<tag:child_id/> = t.<tag:parent_id/>);

    if a_raw.<tag:parent_id/> is not null then
```

```
      RAISE Invalid_record;
    end if;
  END LOOP;

  EXCEPTION
    WHEN Invalid_record THEN
     RAISE_APPLICATION_ERROR (
       num=> <tag:constraint_violation_error_code/>,      //default value = -20107
       msg=> '<tag:constraint_violation_error_msg/>');    //default value = 'Intransitive constraint violated!'
    WHEN no_data_found THEN null;
end;
```

EqualSet apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK ((<foreach:value name="coex_columns"  separator=" AND ">
  e:element IS NOT NULL
</foreach:value>
) OR (
<foreach:value name="coex_columns" separator=" AND ">
  e:element IS NULL
</foreach:value>));
```

SubSet apribojimo šablonas

```
ALTER TABLE <e:table_1/>
ADD CONSTRAINT <tag:constraint_name/>
CHECK ((
   <foreach:value name="v_subset" separator=" AND " separated=",">
      e:element IS NOT NULL
   </foreach:value>
     AND
   <foreach:value name="p_set" separator=" AND " separated=",">
     e:element IS NOT NULL
  </foreach:value>
) OR (
  <foreach:value name="v_subset" separator=" AND " separated=",">
     e:element IS NULL
  </foreach:value>
AND
   <foreach:value name="p_set" separator=" AND " separated=",">
      e:element IS NOT NULL
   </foreach:value>
) OR (
   <foreach:value name="v_subset" separator=" AND " separated=",">
     e:element IS NULL
   </foreach:value>
AND
   <foreach:value name="p_set" separator=" AND " separated=",">
      e:element IS NULL
   </foreach:value>
));
```

13.3. Priedas Nr.3.plugin.xml

ctransformer plugin.xml konfigūracinio failo pavyzdys

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
        id=" edu.ktu.md.ext.cconverter.connectors.magicdraw.MDCCPlugin"
        name="DB model transformation and constraint specification "
        version="1.0"
        provider-name="KTU IFM 2/2"
        class="edu.ktu.md.ext.cconverter.connectors.magicdraw.MDCCPlugin ">
        <requires>
           <api version="1.0"/>
        </requires>
        <runtime>
          <library name="mdccplugin.jar"/>
        </runtime>
</plugin>
```

13.4. Priedas Nr.4. ctransformer.properties

Įskiepis yra konfiguruojamas INI failo pagalba. INI failas randasi įskiepio instaliacijos direktorijoje.. Failo pavadinimas ***ctransformer.config***. Failas nuskaitomas MagicDraw programinio paketo startavimo metu, kai MagicDraw inicializuoja visus įskiepius.
INI failo pagalba galima konfiguruoti SQL dialektų vardus, kurie yra pateikiami vartotojui DB modelio transformacijos į SQL kodą dialoge.

ctransformer INI failo pavyzdys

```
//SQL dialect specification in constraint profile stereotype
SQL_1.SQL.dialect.name=Standard SQL
SQL_2.SQL.dialect.name=Oracle SQL

//Plug-in logger configuration
log4j.debug=false
log4j.rootLogger=info

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%t] (%F:%L) - %m%n

log4j.appender.cconverter=org.apache.log4j.RollingFileAppender
log4j.appender.cconverter.File=cconverter.log
log4j.appender.cconverter.MaxFileSize=1000KB
log4j.appender.cconverter.MaxBackupIndex=1
log4j.appender.cconverter.layout=org.apache.log4j.PatternLayout
log4j.appender.cconverter.layout.ConversionPattern=%d [%t] %-5p %c (%F:%L) - %m%n

log4j.logger.resources=INFO,cconverter
log4j.logger.edu.ktu.md.ext=INFO,cconverter
```

Kiekvienas naujas SQL dialektas užrašomas naudojant sekantį formatą:

ctransforer dialekto specifikavimo šablonas

```
<SQL dialektas profilyje>.SQL.dialect.name=<SQL dialektas dialoge>
```

- <SQL dialektas profilyje> - pavadinimas naudojamas ctransdormer reikalavimų profilyje aprašant apribojimus stereotipui.
- <SQL dialektas dialoge> - pavadinimas naudojamas modelio transformacijos dialoge, pateikiant galimų tranformacijos dialektų sąrašą.

Atlikus pakeitimus INI faile, reikia perstartuoti MagicDraw prograinį paketą, kad pakeitimai būtų nuskaityti.

13.5. Priendas Nr. 5. Junit ir Cobertura testavimo ataskaitų protokolai

Summary

Tests	Failures	Errors	Success rate	Time
262	0	0	100.00%	4.545

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

Packages

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
<none>	7	0	0	0.490	2008-01-10T19:20:58	localhost.localdomain
edu.ktu.md.ext.cconverter.definitions	88	0	0	0.623	2008-01-10T19:20:58	localhost.localdomain
edu.ktu.md.ext.cconverter.parser	15	0	0	0.051	2008-01-10T19:21:54	localhost.localdomain
edu.ktu.md.ext.cconverter.resources	8	0	0	0.021	2008-01-10T19:21:54	localhost.localdomain
edu.ktu.md.ext.cconverter.transformation	76	0	0	3.177	2008-01-10T19:21:54	localhost.localdomain
edu.ktu.md.ext.cconverter.util	53	0	0	0.163	2008-01-10T19:21:57	localhost.localdomain
resources	15	0	0	0.020	2008-01-10T19:21:57	localhost.localdomain

42 pav. *Ctransformer JUnit bendras testavimo protokolas.*

Package edu.ktu.md.ext.cconverter.definitions

Classes

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
ConstraintDefinitionTest	10	0	0	0.024	2008-01-10T19:20:58	localhost.localdomain
ElementStereotypeDefinitionTest	16	0	0	0.480	2008-01-10T19:21:53	localhost.localdomain
LoopDefinitionTest	18	0	0	0.046	2008-01-10T19:21:54	localhost.localdomain
StereotypeDefinitionTest	10	0	0	0.025	2008-01-10T19:21:54	localhost.localdomain
TaqDefinitionTest	14	0	0	0.010	2008-01-10T19:21:54	localhost.localdomain
ValueTaqDefinitionTest	18	0	0	0.016	2008-01-10T19:21:54	localhost.localdomain
XMLValueTaqDefinitionTest	2	0	0	0.022	2008-01-10T19:21:54	localhost.localdomain

43 pav *Paketo edu.ktu.md.ext.cconverter.definitions JUnit bendras testavimo protokolas.*

Package edu.ktu.md.ext.cconverter.parser

Classes

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
AbstractValueTaqParserTest	2	0	0	0.007	2008-01-10T19:21:54	localhost.localdomain
XMLValueTaqParserTest	13	0	0	0.044	2008-01-10T19:21:54	localhost.localdomain

44 pav. *Paketo edu.ktu.md.ext.cconverter.parser JUnit bendras testavimo protokolas.*

Package edu.ktu.md.ext.cconverter.util

Classes

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
ElementUtilityTest	32	0	0	0.019	2008-01-10T19:21:57	localhost.localdomain
LoggerFactoryTest	3	0	0	0.006	2008-01-10T19:21:57	localhost.localdomain
StereotypeUtilityTest	18	0	0	0.138	2008-01-10T19:21:57	localhost.localdomain

45 pav. *Paketo edu.ktu.md.ext.cconverter.util JUnit bendras testavimo protokolas.*

Package edu.ktu.md.ext.cconverter.transformation

Classes

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
AbstractTaqValueFactoryTest	5	0	0	0.034	2008-01-10T19:21:54	localhost.localdomain
CconverterVisitorTest	15	0	0	1.068	2008-01-10T19:21:54	localhost.localdomain
CconverterVisitorUCTest	16	0	0	1.709	2008-01-10T19:21:55	localhost.localdomain
ConstantTaqValueFactoryTest	5	0	0	0.056	2008-01-10T19:21:57	localhost.localdomain
LoopTransformatorTest	22	0	0	0.152	2008-01-10T19:21:57	localhost.localdomain
StereotypeTaqValueFactoryTest	9	0	0	0.123	2008-01-10T19:21:57	localhost.localdomain
TaqValueFactoryTest	4	0	0	0.035	2008-01-10T19:21:57	localhost.localdomain

46 pav. *Paketo edu.ktu.md.ext.cconverter.transformation JUnit bendras testavimo protokolas.*

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	45	66%	1279/1927	68%	523/764	1.5
edu.ktu.md.ext.cconverter.connectors.magicdraw	4	0%	0/107	0%	0/20	0
edu.ktu.md.ext.cconverter.definitions	8	89%	379/424	85%	146/172	2.312
edu.ktu.md.ext.cconverter.parser	6	85%	134/158	96%	27/28	1.429
edu.ktu.md.ext.cconverter.resources	1	56%	32/57	43%	6/14	0
edu.ktu.md.ext.cconverter.transformation	10	74%	367/496	71%	165/232	1
edu.ktu.md.ext.cconverter.ui.actions	5	0%	0/108	0%	0/22	1.824
edu.ktu.md.ext.cconverter.ui.dialgos	6	0%	0/26	N/A	N/A	1
edu.ktu.md.ext.cconverter.util	9	64%	293/458	64%	157/244	1
resources	1	80%	74/93	69%	22/32	0

Report generated by Cobertura 1.9 on 1/10/08 8:22 PM.

47 pav. *Bendra Cobertura kodo testais padengimo protokolas.*

Coverage Report - edu.ktu.md.ext.cconverter.ui.actions

Package	# Classes	Line Coverage		Branch Coverage		Complexity
edu.ktu.md.ext.cconverter.ui.actions	5	0%	0/108	0%	0/22	1.824

Classes in this Package	Line Coverage		Branch Coverage		Complexity
AbstractCCAction	0%	0/5	N/A	N/A	1
AddAttributeConstAction	0%	0/47	0%	0/12	0
AddConnectionConstAction	0%	0/4	N/A	N/A	1
AddMethodConstAction	0%	0/4	N/A	N/A	1
ModelCTransformationAction	0%	0/48	0%	0/10	3.333

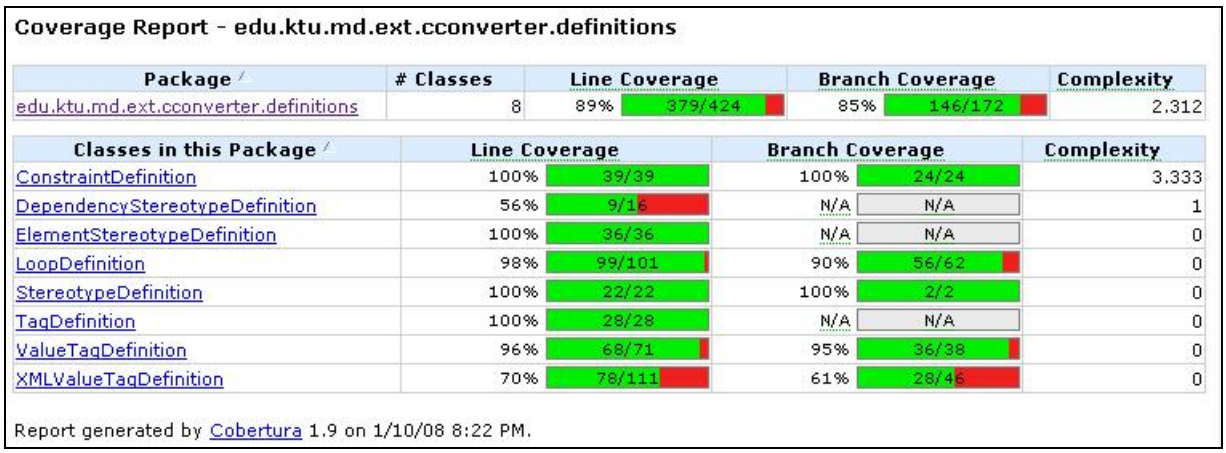
48 pav. *Paketo edu.ktu.md.ext.cconverter.ui.actions Cobertura kodo testais padengimo protokolas.*

Coverage Report - edu.ktu.md.ext.cconverter.connectors.magicdraw

Package	# Classes	Line Coverage		Branch Coverage		Complexity
edu.ktu.md.ext.cconverter.connectors.magicdraw	4	0%	0/107	0%	0/20	0

Classes in this Package	Line Coverage		Branch Coverage		Complexity
MDCCPlugin	0%	0/71	0%	0/6	0
MDCCPlugin$1	N/A	N/A	N/A	N/A	0
MDCCPlugin$MDCCDiagramContextConfigurator	0%	0/24	0%	0/12	0
MDCCPlugin$MDCCToolbarContextConfigurator	0%	0/12	0%	0/2	0

49 pav. *Paketo edu.ktu.md.ext.cconverter.ui.actions Cobertura kodo testais padengimo protokolas.*



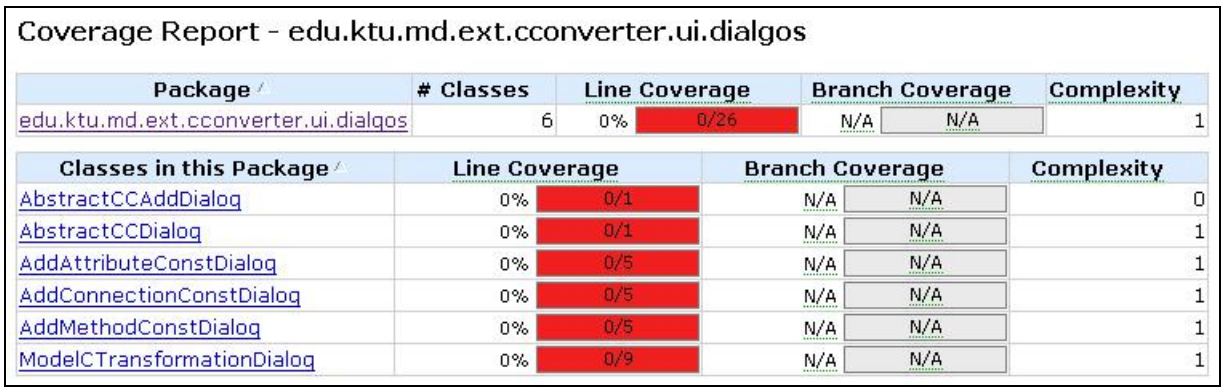
Coverage Report - edu.ktu.md.ext.cconverter.definitions

Package	# Classes	Line Coverage	Branch Coverage	Complexity
edu.ktu.md.ext.cconverter.definitions	8	89% 379/424	85% 146/172	2.312

Classes in this Package	Line Coverage	Branch Coverage	Complexity
ConstraintDefinition	100% 39/39	100% 24/24	3.333
DependencyStereotypeDefinition	56% 9/16	N/A N/A	1
ElementStereotypeDefinition	100% 36/36	N/A N/A	0
LoopDefinition	98% 99/101	90% 56/62	0
StereotypeDefinition	100% 22/22	100% 2/2	0
TagDefinition	100% 28/28	N/A N/A	0
ValueTagDefinition	96% 68/71	95% 36/38	0
XMLValueTagDefinition	70% 78/111	61% 28/46	0

Report generated by Cobertura 1.9 on 1/10/08 8:22 PM.

50 pav. *Paketo edu.ktu.md.ext.cconverter.definitions Cobertura kodo testais padengimo protokolas.*

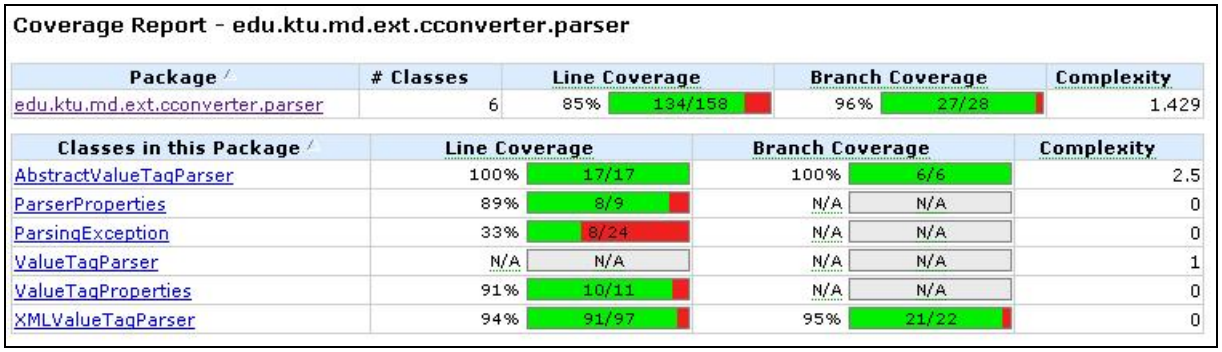


Coverage Report - edu.ktu.md.ext.cconverter.ui.dialgos

Package	# Classes	Line Coverage	Branch Coverage	Complexity
edu.ktu.md.ext.cconverter.ui.dialgos	6	0% 0/26	N/A N/A	1

Classes in this Package	Line Coverage	Branch Coverage	Complexity
AbstractCCAddDialog	0% 0/1	N/A N/A	0
AbstractCCDialog	0% 0/1	N/A N/A	1
AddAttributeConstDialog	0% 0/5	N/A N/A	1
AddConnectionConstDialog	0% 0/5	N/A N/A	1
AddMethodConstDialog	0% 0/5	N/A N/A	1
ModelCTransformationDialog	0% 0/9	N/A N/A	1

51 pav. *Paketo edu.ktu.md.ext.cconverter.ui.dialogs Cobertura kodo testais padengimo protokolas.*

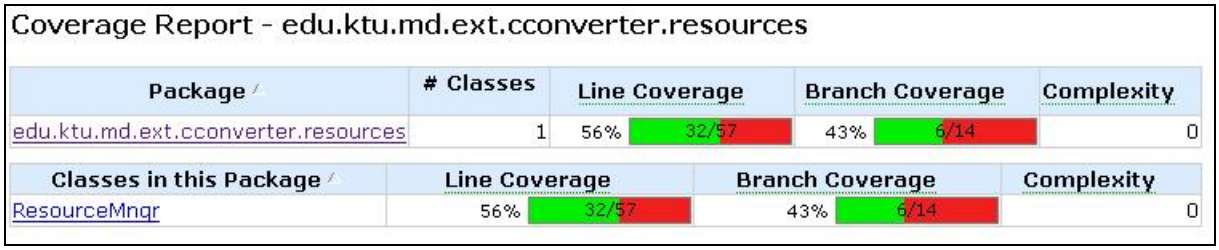


Coverage Report - edu.ktu.md.ext.cconverter.parser

Package	# Classes	Line Coverage	Branch Coverage	Complexity
edu.ktu.md.ext.cconverter.parser	6	85% 134/158	96% 27/28	1.429

Classes in this Package	Line Coverage	Branch Coverage	Complexity
AbstractValueTagParser	100% 17/17	100% 6/6	2.5
ParserProperties	89% 8/9	N/A N/A	0
ParsingException	33% 8/24	N/A N/A	0
ValueTagParser	N/A N/A	N/A N/A	1
ValueTagProperties	91% 10/11	N/A N/A	0
XMLValueTagParser	94% 91/97	95% 21/22	0

52 pav. *Paketo edu.ktu.md.ext.cconverter.parser Cobertura kodo testais padengimo protokolas.*



Coverage Report - edu.ktu.md.ext.cconverter.resources

Package	# Classes	Line Coverage	Branch Coverage	Complexity
edu.ktu.md.ext.cconverter.resources	1	56% 32/57	43% 6/14	0

Classes in this Package	Line Coverage	Branch Coverage	Complexity
ResourceMngr	56% 32/57	43% 6/14	0

53 pav. *Paketo edu.ktu.md.ext.cconverter.resources Cobertura kodo testais padengimo protokolas.*

Coverage Report - edu.ktu.md.ext.cconverter.transformation

Package	# Classes	Line Coverage	Branch Coverage	Complexity
edu.ktu.md.ext.cconverter.transformation	10	74% 367/496	71% 165/232	1

Classes in this Package	Line Coverage	Branch Coverage	Complexity
AbstractTaqValueFactory	86% 6/7	50% 3/6	0
AbstractTaqValueFactory$1	100% 1/1	N/A N/A	0
AbstractTaqValueFactory$SqlFactoryId	86% 6/7	N/A N/A	0
CconverterVisitor	65% 155/239	68% 54/80	0
ConstantTaqValueFactory	96% 26/27	94% 15/16	0
LoopTransformator	80% 106/133	65% 56/86	0
ModelException	0% 0/1	N/A N/A	0
StereotypeTaqValueFactory	90% 46/51	86% 31/36	0
TaqValueFactory	95% 19/20	75% 6/8	0
TransformationException	20% 2/10	N/A N/A	1

54 pav. *Paketo edu.ktu.md.ext.cconverter.transformation Cobertura kodo testais padengimo protokolas.*

Coverage Report - edu.ktu.md.ext.cconverter.util

Package	# Classes	Line Coverage	Branch Coverage	Complexity
edu.ktu.md.ext.cconverter.util	9	64% 293/458	64% 157/244	1

Classes in this Package	Line Coverage	Branch Coverage	Complexity
ClassDiagramReader	0% 0/40	0% 0/16	0
ClassDiagramReaderInterface	N/A N/A	N/A N/A	0
ElementUtility	87% 118/135	80% 77/96	0
LoggerFactory	62% 33/53	62% 5/8	0
ModelAdapter	0% 0/16	N/A N/A	0
PatternClassDiagramReader	0% 0/10	N/A N/A	0
SQLWriter	0% 0/11	N/A N/A	1
StereotypeUtility	75% 142/189	60% 75/124	0
WriterException	0% 0/4	N/A N/A	1

55 pav. *Paketo edu.ktu.md.ext.cconverter.util Cobertura kodo testais padengimo protokolas.*