**KAUNO TECHNOLOGIJOS UNIVERSITETAS**
**INFORMATIKOS FAKULTETAS**
**INFORMACIJOS SISTEMŲ KATEDRA**


**Mindaugas Praninskas**

# INFORMACINĖS SISTEMOS KLASIŲ MODELIO PATIKSLINIMAS VEIKLOS TAISYKLIŲ PAGRINDU

Magistro darbas

**Vadovas**
**doc. dr. T. Skersys**

**KAUNAS, 2009**

# Turinys

# Lentelių turinys

# Paveikslėlių turinys

# 1. ĮŽANGA

Programinės įrangos kūrimo procese reikalavimų rinkimo ir jų analizės etapai yra patys svarbiausi, nuo jų priklauso tolesnių programinės įrangos kūrimo etapų įgyvendinimas. Informacinių sistemų specifikavimui dažniausiai naudojamos projektavimo kalbos (UML, Z, DFD, OCL ir t.t.), kurių supratimui reikalingos specifinės žinios. Užsakovo projektavimo kalbos žinios dažnai yra minimalios, tad jis pasitiki projektuotojo žiniomis ir pateiktų reikalavimų interpretavimu. Projektavimo etape tinkamai užsakovo nesuprastų, formalia kalba specifikuotų, sistemos projektinių modelių netikslumai, dažnai išryškėja tik realizavimo etape. Klaidingai IT specialisto interpretuotų žinių padarinių taisymas yra brangus ir pailgina programinio produkto kūrimo trukmę.

Šiam nesusišnekimui išvengti buvo pasiūlyta dalykinę sritį aprašinėti veiklos taisyklėmis natūralios kalbos šablonais. Veiklos taisyklės yra struktūrizuotos ir vienareikšmiškai interpretuojamos tiek IT specialistų, tiek verslo atstovų. Šis dalykinės srities aprašymo metodas yra labai neseniai pasirodęs ir dar nėra nusistovėjusių standartų jo taikyme.

Šiame darbe bus kuriamas algoritmas, kuris veiklos taisyklių pagrindu patikslins nuo skaičiavimų nepriklausomo (arba nuo platformos nepriklausomo) lygmens duomenų modelius. Tuo savo ruoštu paspartindamas projektinių modelių kūrimą ir patikslinimį automatizuojant veiklos taisyklių ir klasių modelio analizę.

# 2. ĮVADAS

Šiame darbe bus pateiktas modeliais grindžiamo sistemų kūrimo principas, bus supažindinta su veiklos taisyklių sąvoka, jų tipais, užrašymo būdais. Bus apžvelgti veiklos taisyklių valdymo sistemų privalumai, veiklos taisyklių panaudojimas klasių modelio praplėtime ir kitose informacinių sistemų taikymo srityse. Šiame darbe bus pateiktas pasiūlymas klasių modelio išplėtimui veiklos taisyklių pagrindu, algoritmo papildymas bei jo projektiniai modeliai. Taip pat bus aprašytas MagicDraw UML klasių modelio praplėtimo ir įskiepio kūrimo mechanizmas.

Atliekant tyrimą yra būtina apibrėžti pagrindinius su darbu susijusius aspektus:

- **Tyrimo sritis -** kompiuterizuota informacinių sistemų inžinerija.
- **Tyrimo objektas -** informacinių sistemų projektinių sprendimų (klasių modelio) patikslinimas veiklos taisyklių pagrindu.
- **Tyrimo problema ir numatomas sprendimas -** sistemos klasių modelio patikrinimas dalykinės srities pagrindu yra komplikuotas, nes nėra formalių metodų. Šiame darbe bus analizuojamos veiklos taisyklės esančios veiklos taisyklių saugyklos duomenų bazėje ir jų pagrindu bus analizuojamas klasių modelis. Klasių modelio analizei ir papildymui bus remiamasi informacijos sistemų katedros mokslininkų pasiūlytu algoritmu, kurio funkcionalumas bus praplėstas šiame darbe.

Šiame darbe pateiktas metodas turėtų pagerinti klasių modelio korektiškumą. Korektiškumą pagerins veiklos taisyklėmis natūralia kalba aprašytos dalykinės srities žinių integravimas į klasės modelį. Kadangi veiklos taisyklių, užrašytų natūralia kalba, pagalba bus galima patikslinti ir generuoti klasių modelį, tai šiame darbe realizuotas algoritmas turėtų paspartinti informacinių sistemų kūrimą ir taip sumažinti jų kaštus bei kainą. Kadangi klasių modelis galės būti generuojamas automatiškai, tai tūrėtų išnykti užsakovo nesusišnekėjimo su informacinės sistemos projektuotoju, problema, kuriai išaiškėjus tik realizacijos metu būna sudėtinga ir brangu ją spręsti. Šiame darbe pasiūlytas algoritmas yra tik vienas iš daugelio žingsnių dalykinės srities žinių taikyme kuriant informacines sistemas.

## *2.1.DARBO TIKSLAS IR UŽDAVINIAI*

**Pagrindinis tikslas** yra pagerinti kuriamų informacinių sistemų projektinių sprendimų (klasių modelio) korektiškumą, pasiūlant veiklos taisyklėmis grindžiamo klasių modelio papildymo algoritmą ir jį realizuojantį įskiepio prototipą. Šis darbas yra atliekamas norint veiklos taisyklių pagrindu patikslinti nuo skaičiavimų nepriklausomo (arba nuo platformos nepriklausomo) lygmens duomenų modelius.

**Pagrindiniai darbo uždaviniai**:

1. Išanalizuoti esamus veiklos taisyklių taikymo būdus IS kūrimo procese.
2. Išanalizuoti ISK vykdomus tyrimus informacinių sistemų kūrimo proceso papildymo veiklos taisyklėmis srityje.
3. Išanalizuoti esamus veiklos taisyklių specifikavimo (užrašymo) būdus naudojant natūralios kalbos šablonus.
4. Atlikus analizę pateikti klasių modelio išplėtimo veiklos taisyklių (toliau VT) pagrindu algoritmo papildymo pasiūlymus.
5. Sukurti papildyto algoritmo realizacijos projektinę specifikaciją.
6. Išanalizuoti MagicDraw UML projektinių modelių ir praplėtimo galimybes ir būdus.
7. Išanalizuoti programinio MagicDraw UML paketo funkcionalumo išplėtimo galimybes ir mechanizmą.
8. Parengtos projektinės specifikacijos pagrindu realizuoti įskiepį.
9. Ištestuoti sukurtą įskiepį, atliekant algoritmo funkcionalumo verifikaciją.

# 3. VT IR JŲ TAIKYMO INFORMACINIŲ SISTEMŲ KŪRIME ANALIZĖ

## *3.1.ANALIZĖS TIKSLAS*

Analizės tikslas yra susipažinti su:

- modeliais grindžiamo sistemų kūrimo principu ir veiklos taisyklių vieta šioje architektūroje;
- veiklos taisyklėmis;
- veiklos taisyklių tipais;
- veiklos taisyklių užrašymo būdais;
- veiklos taisyklių valdymo sistemomis (toliau VTVS) ir saugyklomis (toliau VTS);
- veiklos taisyklių taikymu;
- Pasiūlytu veiklos taisyklių taikymo klasių modelio papildymui algoritmu;

## *3.2.ANALIZĖS METODAI*

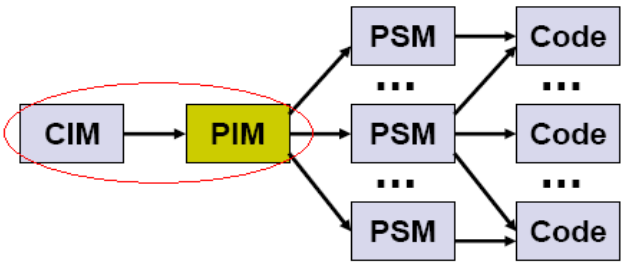
Kadangi nėra pilnai realizuoto veiklos taisyklėmis grindžiamo klasių modelio patikslinimo metodo – tai yra pasirinktas mokslinės literatūros arba teorinės analizės ir apibendrinimo metodas. Naudojantis šio metodo metodologija bus analizuojama mokslinė literatūra (publikacijos, moksliniai straipsniai, knygos, su sprendžiama problema susijusių metodų specifikacijos ir kt.).Kadangi bus bandoma pagerinti ISK kuriamą metodą, daugiausiai bus analizuojami ISK atlikti moksliniai darbai ir išleisti moksliniai straipsniai šia tema.

## *3.3.TYRIMO OBJEKTO ANALIZĖ*

Tyrimo objektas – informacinių sistemų projektinių sprendimų (klasių modelio) patikslinimas veiklos taisyklių pagrindu.

Kuriant informacinių sistemų projektus yra susiduriama su daug problemų: pradedant per menku informacijos iš kliento gavimu ir baigiant projektuotojo su klientu nesusišnekėjimu. Projektavimo etapo palengvinimui ir našumo bei kokybės didinimui yra kuriamos įvairios teorijos, metodai bei informacinių sistemų projektavimo įrankiai.

Viena iš lyderių informacinių sistemų specifikavimo srityje OMG[21] (angl. Object management group), pasiūlė specifikavimo ir projektavimo architektūrą - MDA[1] (pav.1.) (angl. Model – driven architecture), kuri projektavimo informacinės sistemos funkcionalumo specifikaciją atskiria nuo realizavimo konkrečioje platformoje specifikacijos.




1. pav. MDA architektūra [1]

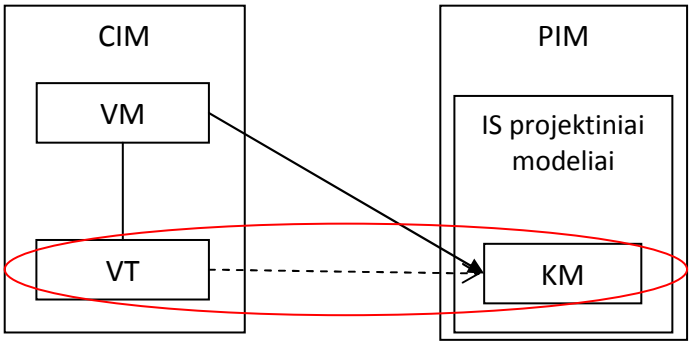
Ši architektūra susideda iš 3 modelių:

- nuo skaičiavimų nepriklausomas modelis (veiklos modelis) - CIM (angl. computation independent model),
- platformos nepriklausomas modelis – PIM (ang. Platform independent model),
- platformos priklausantis modelis – PSM (angl. platform specific model).

Jei ši architektūra būtų realiai įgyvendinta ir galima būtų iš specifikacijos modelio (CIM) automatizuotai gauti žemesnius projektavimo lygius ir jie būtų vienas nuo kito dalinai priklausomi, tai projektavimo našumas sparčiai padidėtų.

Kauno technologijos universiteto informacinių sistemų katedros (toliau ISK) mokslininkai kuria veiklos taisyklėmis grindžiamą informacinių sistemų projektavimo metodą, kuris turi palengvinti informacinės sistemos reikalavimų rinkimą, jų formalizavimą ir automatinį pritaikymą konkrečiam klasių modeliui. Šis metodas turėtų sumažinti nesusišnekėjimo problemą su klientais, nes yra kuriamas siekiant veiklos taisyklių įvedimo natūralia kalba.

Šiame darbe bus bandoma pagerinti vieną iš projektavimo etapų, t.y. bus bandoma suintegruoti(transformuoti) CIM modelyje formuojamą veiklos taisyklių modelį (žinių bazę apie kuriamą objektą) ir PIM modelyje formuojamą klasių modelį.




2. pav. Tyrimo objektas

Taigi darbe bus mėginama integruoti/transformuoti dalį CIM modelio, kuris susideda iš veiklos modelio (VM) (angl. enterprise model (EM)) ir veiklos taisyklių (VT) (angl. business rules (BR)), struktūrizuotų natūralios kalbos šablonais, su PIM modelio dalimi – klasių modeliu (KM) (angl. class

model (CM)), aprašytu formalia modeliavimo kalba (UML), ko pasėkoje bus galima automatiškai papildyti ar apriboti klasės modelio objektus remiantis veiklos taisyklių žinių baze.

## *3.4. VARTOTOJŲ ANALIZĖ*

### Vartotojų aibė, tipai ir savybės

Kuriamo produkto vartotojų aibė susideda iš informacinių sistemų (IS) projektuotojų, IS kūrėjų ir IS užsakovų.

*IS projektuotojai* - specialistai, kurie gerai žino įvairias projektavimo kalbas (kaip UML, Z, DFD, OCL ir t.t.) ir jomis aprašo specifikacijoje sukauptas dalykinės srities žinias bei laukiamus rezultatus iš kuriamos IS.

*IS kūrėjai* – tai specialistai, kurie gerai žino į vairias projektavimo kalbas ir skaitydami IS specifikaciją pateiktą šia kalba, ją realizuoja (parašo programos kodą, sujungia aparatūrą ir kt.).

*IS užsakovas* – tai fizinis ar juridinis asmuo, kuris dažniausiai neturi jokių IS projektavimo kalbų žinių arba jos būna labai menkos ir ribotos, tačiau jis bando perteikti norimą kuriamos IS funkcionalumą IS projektuotojui.

### Vartotojų tikslai ir problemos

IS projektuotojų tikslas yra surinkti kuo daugiau dalykinės srities žinių iš užsakovo, jas tiksliai pecifikuoti natūralia kalba klientui ir projektavimo kalba IS kūrėjui.

IS užsakovo tikslas yra konkrečiai išdėstyti norimą IS funkcionalumą ir kruopščiai išnagrinėti projektuotojo pateiktą specifikaciją natūralia kalba prieš ją pateikiant realizacijai.

IS kūrėjų tikslas yra sukurti informacinę sistemą, kuri atitiktų specifikaciją. Informacinė sistema dažniausiai yra kuriama iš specifikacijos pateiktos formalizuota projektavimo kalba (UML).

Dažniausiai kyla problemos dėl kliento ir projektuotojo nesusišnekėjimo ir nevienodo konkrečių teiginių interpretavimo, ko pasėkoje yra gaunama informacinė sistema, kuri neatitinka užsakovo reikalavimo ir ją reikia taisyti. Kadangi reikia taisyti programą, kuri buvo blogai specifikuota, tai kuo vėliau yra pastebimi netikslumai tuo IS kūrėjai turi daugiau darbo, kad problema būtų išspręsta, kas labai padidina IS kūrimo kaštus.

Taigi iš kuriamo įrankio daugiausiai naudos turės IS projektuotojai ir IS užsakovai. Kadangi darbe bus bandoma integruoti specifikavimo metu gautas žinias natūralios kalbos šablonais, tai klientas be didelių informacinių sistemų projektavimo kalbų žinių galės nesunkiai suprasti specifikaciją ir IS kūrėjai pagal IS specifikaciją lengviau supras, ką reikia realizuoti.

### *3.5. VEIKLOS TAISYKLĖS (VT)*

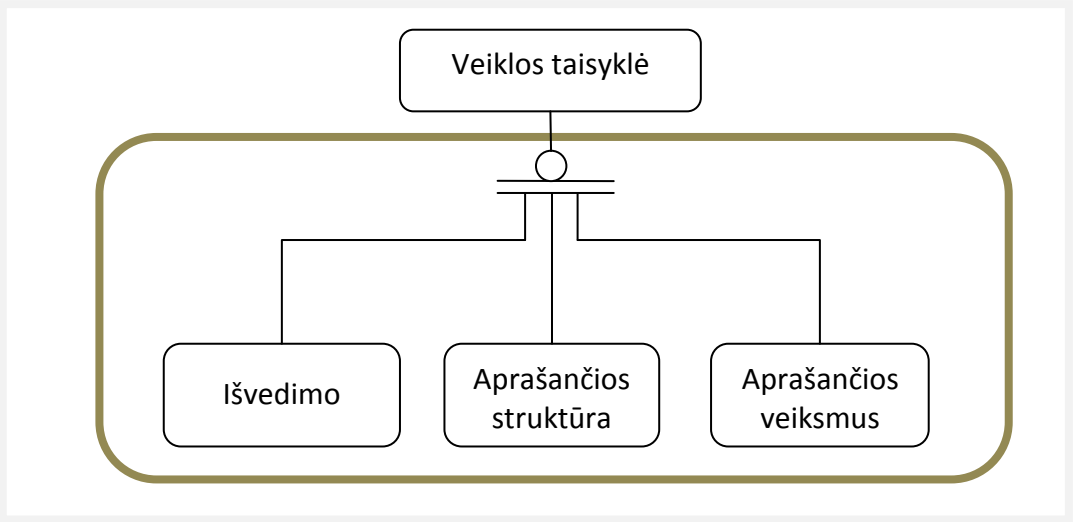
Informacinių sistemų analitikų grupė Business Rules Group [5] veiklos taisykles aprašė taip:

**Veiklos taisyklė** – tai išraiška, kuri apibrėžia arba riboja kai kurios veiklos aspektus. Ji gali būti kaip terminas ar faktas (kaip struktūrinis tvirtinimas (angl. asserration)), suvaržymas (kaip veiksmo tvirtinimas) arba išvedimas. Ji yra "atominė", nes ji negali būti išskaidytą į išsamesnes veiklos taisykles. Jeigu ji būtų sumažinama, tai būtų prarandama svarbi informacija apie veiklą.

Yra ir kitų veiklos taisyklių apibrėžimų [7]:

- Tai taisyklės, nustatančios, kaip turi būti vykdoma organizacijos veikla.
- Tai žinių apie organizacijos veiklą fragmentas.
- Tai teiginys, kuris apibrėžia arba sąlygoja tam tikrą veiklos aspektą. VT skirta apibrėžti veiklos struktūrai arba valdyti, įtakoti veiklos elgseną.
- Tai atominis, deklaratyviai specifikuotas, daugkartinio panaudojimo veiklos logikos fragmentas
- Tai teiginys, kuris kontroliuoja arba įtakoja veiklos procesų vykdymą arba apibrėžia resursų, reikalingų veiklai vykdyti, struktūrą.

Informacinių sistemų analitikų grupė Business Rules Group suskirstė veiklos taisykles į tris tipus (pav. 3).




3. pav. Veiklos taisyklių tipai[5]

Kiekviena „atominė“ veiklos taisyklė gali būti:

- **Aprašančios struktūrą** – aprašo fakto konceptą arba teiginį, kurio išraiška kažkuriuo aspektu aprašo dalykinės srities struktūrą.
- **Aprašančios veiksmus** – aprašymas veiksmų apribojimų arba sąlygų, kuriuos riboja arba kontroliuoja dalykinė veikla.
- **Išvedimo** – žinių rinkinys, kuris yra gaunamas iš kitų veiklos žinių.

IBM organizacija naudoja keletą specifinių veiklos taisyklių specifikavimo tipų [6]:

- **Nekintamumo taisyklės -** taisyklės, kurios užtikrina, kad daug pakeitimų, atliktų operacijų metu būtų tinkamai susiję vienas su kitu.
- **Scenarijaus taisyklės -** scenarijai naudoja "mikro-sekų" arba elektronikos galimybių palaikymą. Jie yra nedidelės veiklos proceso kintamųjų aibės, kurios gali suteikti maksimalią naudą galutiniam vartotojui iš taikomojo uždavinio.

Veiklos taisyklės, pagal joms būdingus bruožus, smulkiau buvo suklasifikuotos į tokius tipus (lentelė Nr.1.) [7]:

Lentelė Nr. 1. Veiklos taisyklių kvalifikacija

| **Veiklos Taisyklės tipas** | **Apibrėžimas** |
|---|---|
| Terminas | Su nagrinėjama veiklos sritimi susijęs žodis arba frazė. |
| Faktas | Teiginys, nustatantis veiklos srities atžvilgiu prasmingą ryšį tarp dviejų (ar daugiau) terminų. |
| Apribojimo | Teiginys, nusakantis būtiną veiklos objekto savybę, kuri turi būti patenkinta. |
| Išvesties | Išraiška, kurioje naudojamos loginės operacijos naujam faktui išvesti. |
| Veiksmo | Teiginys, nusakantis sąlygas veiksmui inicijuoti. |
| Skaičiavimo | Išraiška fakto reikšmei paskaičiuoti. |
| Rolės | Teiginys, kuris nurodo ryšyje dalyvaujančio objekto rolę fakte, kuriame aprašomas tas ryšys. |

Taigi taisyklėmis galima aprašyti faktus, terminus, apribojimus, loginius išvedimus, veiksmus, specifinius skaičiavimus ir roles.

Veiklos taisyklės yra rinkinys taisyklių, kurios nusako, riboja ar įtakoja konkrečius veiksmus, juos vykdant konkrečiomis sąlygomis. Veiklos taisyklės yra apibrėžiamos projekto dalykinės srities reikalavimų rinkimo etape, kai yra bandoma surinkti visus veiksnius, kurie turi įtakoti konkretų projekte kompiuterizuojamą objektą ar jo operaciją. Idealiu atveju veiklos taisyklių rinkinį turėtų būti galima papildyti bet kuriuo metu, kai to reikalauja verslo ypatumai ir automatiškai būtų pakeičiamas taikomojo uždavinio vykdymas. Be to veiklos taisyklių pagalba yra aprašomi ir kompiuterizuojamos sistemos vartotojai su jų funkcionalumo apribojimais. Taip pat yra aprašomi ir reikšmingi objektų kintamieji (struktūra, dydis, kiekis).

### 3.5.1. VEIKOS TAISYKLIŲ GYVAVIMO CIKLAS (VTGC)

Veiklos taisyklės yra projektavimo etapas ir kaip visi projektavimo etapai turi savo vystimosi raidą ir tvarką. Projektuojamas objektas (šiuo atveju veiklos taisyklės) turi vystimosi, procesą (dažniausiai iteracinį),kuris vadinamas gyvavimo ciklu.

Veiklos taisyklės sudarinėjamos etapais - vadinamu veiklos taisyklių gyvavimo ciklu. Toliau seks tipiški veiklos taisyklių etapai [22]:

- **Taisyklės užfiksavimas**

  Kiekvienas verslo lygis turi tam tikrą terminologiją, kuri yra unikali tam lygiui. Visi verslo politiką aprašantys dokumentai ir terminai yra apdorojami kol yra suformuluojamos taisyklės. Dėl to šių verslo terminų surinkimas yra pirmasis veiklos taisyklių evoliucijos etapas.

- **Taisyklių parengimas eksplotuoti**

  Taisyklių automatizavimo scenarijuje, taisyklių parengimas eksploatavimui – tai procesas, kai veiklos taisyklės yra paimamos iš įvairių popierinių dokumentų ir yra konvertuojamos į mašinai skaitomą kodą.

- **Taisyklių testavimas**

  Kartą sukurtos taisyklės turi būti kruopščiai ištestuotos prieš jas panaudojant sekančiuose veiksmuose. Taisyklių testavimo etape veiklos taisyklės yra pateikiamos testavimui, kad būtų įsitikinta, kad jos veiks taip kaip tikimasi.

- **Taisyklių įtraukimas**

  taisyklių įtraukimo etape, taisyklės ar taisyklių valdymo sistema yra integruojama į taikomąjį uždavinį. Lengva integracija ir tikimas yra svarbi aplinkybė šiame etape.

- **Taisyklės vykdymas**

  taisyklių vykdymo etape, veiklos taisyklių modulis paleidžia taisykles taikomojo uždavinio viduje.

- **Taisyklių analizė ir pakeitimai**

  taisykles reikia inspektuoti dažnais intervalais, kad būtų įsitikinta, kad jos išliko suderinamos su užsakovo reikalavimais. Taisyklių analizės ir pakeitimų etapas – tai toks taisyklių gyvavimo ciklo etapas, kuriame taisyklės yra analizuojamos ir keičiamos derinant jas su verslo raida.

- **Taisyklių palaikymas**

  taisykles reikia išlaikyti visą laiką. Turi būti palaikomas taisyklių versijavimas, palaikoma taisyklių integracija ir turi būti garantuojama, kad jos bus modifikuotos laiku, kad tiktų esamam verslo scenarijui.

Iš esmės, verslo taisyklių gyvavimo ciklas yra iteracinis procesas. Faktiškai, taisyklė gali pereiti daug kiekvieno gyvavimo ciklo etapo iteracijų prieš pereidama į kitą etapą. Veiklos taisyklės

yra rašomos, paleidžiamos, keičiamos ir netgi atmetamos po jų gyvavimo ciklo. Naują taisyklę reikia ilgai derinti su užsakovu. Kuo didesnė verslo dinamika tuo labiau reikalingi veiklos taisyklių pakeitimai. Ideali veiklos taisyklių sistema leis efektyviai valdyti kiekvieną iš šių anksčiau paminėtų etapų.

Taigi dalykinės srities veiklos taisyklių kūrimo etapai turi iteracinį vystymo gyvavimo ciklą. Veiklos taisyklės yra skirtos palengvinti (suteikti jiems daugiau ilgalaikiškumo ir pastovumo) tolimesnius projektavimo etapus, t.y. teoriškai bet kada papildžius veiklos taisyklių žinių bazę turėtų automatiškai pasipildyti ir tolimesniuose etapuose sukurti modeliai.

Veiklos taisyklių taikymas projektavimo darbuose yra naujas projektavimo etapas – žiniomis grindžiamas informacinių sistemų projektavimo etapas.

Kadangi IS yra kuriamos didžiąja dalimi verslui, o šiuolaikinis verslas yra labai dinamiškas ir greitai kintantis, tai šis IS projektavimo etapas yra neišvengiamas: tik panaudojus tokius projektavimo sprendimus galima greitai keisti vykdomų uždavinių logiką ir ribojimus be didelių, ilgų ir brangių IS pataisymų.

### 3.5.2. VEIKLOS TAISYKLIŲ UŽRAŠYMAS NATŪRALIOS KALBOS ŠABLONAIS

Viena didžiausių problemų dabartinėje informacinių sistemų inžinerijoje yra bendravimas tarp sistemos užsakovų ir kūrėjų. Dažniausiai iškyla problemos IS kūrime, kai pradiniame etape IS projektuotojas ne taip supranta užsakovo reikalavimus, o užsakovas patvirtina projektą, nors ir nesupranta IS projektuotojo sudaryto sistemos modelio struktūrizuoto technine kalba (pvz. UML). Kadangi pradinėje stadijoje iškyla nesusikalbėjimas užsakovo su projektuotoju, tai padarytos sistemos ar dalinai padarytos IS taisymas užtrunka ir projekto kaštai didėja, nes reikia taisyti ar papildyti sukurtą sistemą, kas dažniausiai yra žymiai sudėtingiau nei trūkstamų uždavinių sprendimą realizuoti pirminėje kūrimo stadijoje.

Šiai problemai išspręsti buvo pasiūlytas veiklos taisyklių užrašymas natūralios kalbos šablonais. Nacionaline kalba (mūsų atveju lietuvių) paremti šablonai suderinti su veiklos taisyklių saugykla, leidžia struktūrizuoti taisyklių užrašymą, dalykinės srities žinovams suprantamesne kalba. Šio VT užrašymo mechanizmas yra paremtas VT užrašymo anglų kalba principais, tačiau pritaikytas mūsų valstybinės kalbos ypatumams.

Veiklos taisyklių užrašymas natūralios kalbos šablonais yra kuriamas remiantis veiklos taisyklių struktūra pagal EBNF[1] ir kitus su natūralios kalbos šablonais susijusius straipsnius [26][8][9].

VT natūralios kalbos šablonai ir jų tipai:

- <Template_Fact>::= <Term> <ResWord> <Term> – šablonas Faktams;

[1]EBNF [23] - Extended Backus-Naur notacija (labiau žinoma kaip EBNF ar Extended Backus-Naur Form) tai formalus matematinis kalbos apibrėžimas, kuris yra sukurtas John Backus (kaip žinoma, gali būti ir Peter Naur) aprašyti Algol 60 programavimo kalbos sintaksei.

•<Template_Constraint>::= (<Fact> | <Term>) <ResWord> {[<ResWord>] | [<ResWord> <Value>]} [(<Fact> | <Term> | <Value>)] – šablonas ribojimams; •<Template_Inference>::= <ResWord> <Fact> <ResWord> (<Fact>|<Value>) [{<ResWord> <Fact> <ResWord> (<Fact>|<Value>)}] <ResWord><Fact> <ResWord>(<Fact> | <Value>) – šablonas išvedimo VT;

• <Template_Action>::= <ResWord> <Fact> <ResWord> (<Fact> | <Value>) [{<ResWord> <Fact> <ResWord> (<Fact> | <Value>)}] <ResWord> <Action> (<Term> | <Fact> | <BR>) – šablonas veiksmo taisyklėms.

• <Template_Computation>::= <Fact> <ResWord> (<Fact> | <Value>) {[<ResWord> (<Fact> | <Value>)]} – šablonas skaičiavimo VT.

Išanalizavus Semantics of Business Vocabulary and Business Rules [9] buvo nuspręsta, kad būtų naudojamas dar vienas šablonas:

- <Template_Role>::=<Term><ResWord><Fact>”-“<Term> - rolės šablonas.

Šiuo šablonu yra aprašoma termino rolė, kuria jis atlieka dalyvaudamas fakte, kuriame yra aprašomas ryšys.

Taigi veiklos taisyklių užrašymas natūralia kalba remiasi struktūrizuota ir šablonizuota kalba, kurioje atitinkami natūraliems kalbos sinonimams yra priskiriama konkreti taisyklėje dalyvaujančio žodžio reikšmė. Veiklos taisyklės struktūrizuotu pavidalu yra saugomos veiklos taisyklių saugykloje ar saugyklose apie kurias yra aprašyta 9.4 skyriuje.

### 3.5.3. KITI VEIKLOS TAISYKLIŲ UŽRAŠYMO SPRENDIMAI

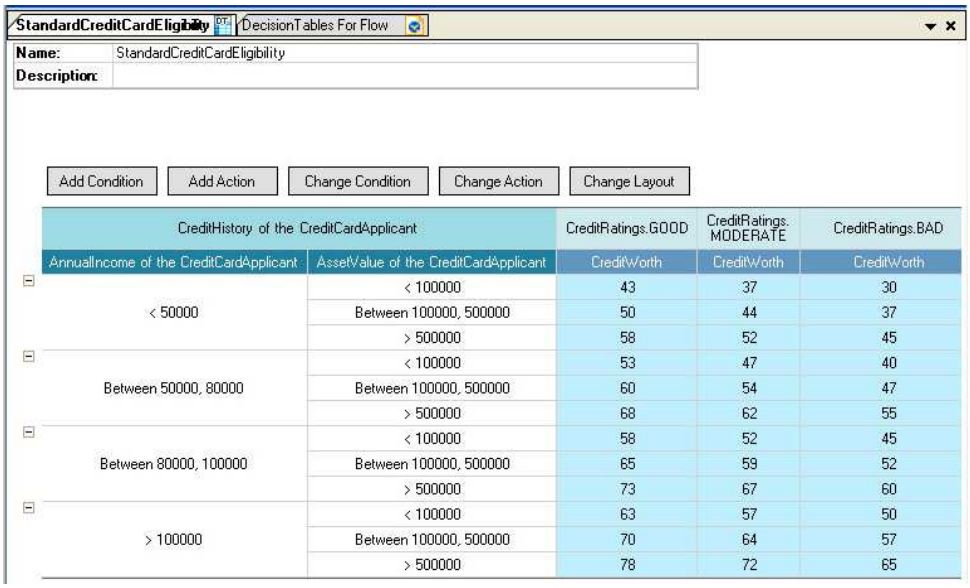
Kadangi žinių baze grindžiama informacinių sistemų inžinerija yra labai jauna IS inžinerijos šaka, tai nėra apibrėžtų tikslių standartų ir daugelis sprendimų yra specializuoti tik individualiam produktui (CASE įrankiui) ir jų naudojami algoritmai yra kūrėjų pramoninė paslaptis. Daugelis realizuotų VT taikymo įrankių yra orientuoti į konkrečias platformas (JAVA, C#, C ir kt.). Visi sukurti sprendimai yra taikomi prie jau nusistovėjusių ir standartu tapusių modeliavimo kalbų ir jų modelių (UML, OCL ir kt.).

Veiklos taisyklių užrašymo būdai taip pat yra skirtingi nuo rašymo formalia struktūrizuota kalba iki pateikiamų natūralios kalbos šablonų su ribotais veiksmų pasirinkimais [11].

Veiklos taisyklės gali būti skirtingų tipų priklausomai nuo faktų skaičiaus kurie aprašo jų sudėtingumo lygį. Kai tam tikrom taisyklėms užtenka paprasčiausio „Jei-Tai“ formato, tai daugeliui kitų yra daug sudėtingesni išraiškos būdai. Todėl atsiranda būtinybė leisti veiklos taisykles aprašyti skirtingais tipais, kol garantuojama, kad matomumas - atvaizdavimas nenukentės.

Labiausiai paplitusios taisyklės yra **JEI-TAI** tai paprasčiausias veiklos taisyklės tipas, kuris turi sąlygą ir veiksmus. Kompanija YASU Technologies yra realizavusi keletą netradicinių VT užrašymo būdų, tokių kaip VT sprendimų lentelė ir VT srautų diagramas.

**Sprendimų lentelės** apjungia susietas taisyklių lentelės „MS Excel“ lentelių pavidalu. Sprendimų lentelė YASU Technologies sukurtame „QuickRules.NET“ pakete yra unikalus sprendimas, kurį galima įvertinti Decision table Engine technologija. Šis komponentas yra specialiai sukurtas įvertinti netgi labai didelėms sprendimų lentelėms per nykstamai mažą laiką. Šių taisyklių užrašymo forma programiniame paketa pavaizduota paveikslėlyje (pav. 4):



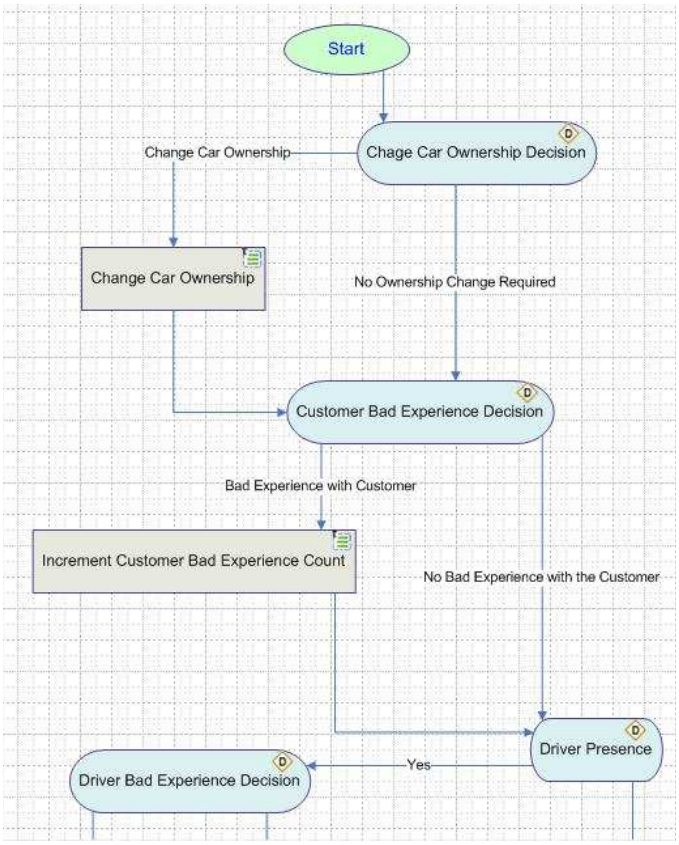

| CreditHistory of the CreditCardApplicant | | CreditRatings.GOOD | CreditRatings.MODERATE | CreditRatings.BAD |
|---|---|---|---|---|
| AnnualIncome of the CreditCardApplicant | AssetValue of the CreditCardApplicant | CreditWorth | CreditWorth | CreditWorth |
| < 50000 | < 100000 | 43 | 37 | 30 |
| | Between 100000, 500000 | 50 | 44 | 37 |
| | > 500000 | 58 | 52 | 45 |
| Between 50000, 80000 | < 100000 | 53 | 47 | 40 |
| | Between 100000, 500000 | 60 | 54 | 47 |
| | > 500000 | 68 | 62 | 55 |
| Between 80000, 100000 | < 100000 | 58 | 52 | 45 |
| | Between 100000, 500000 | 65 | 59 | 52 |
| | > 500000 | 73 | 67 | 60 |
| > 100000 | < 100000 | 63 | 57 | 50 |
| | Between 100000, 500000 | 70 | 64 | 57 |
| | > 500000 | 78 | 72 | 65 |

4. pav. Sprendimų lentelių redaktorius[22]

Taisyklės turinčios daug sąlygų (sąlygų seką) ir veiksmų yra geriausiai pateikiamos sprendimų lentelės forma. Ši forma yra priimtinausia siekiant pagerinti VT matomumui ir supratimui. Šioje lentelėje pateikiamas sąrašas sąlygų, kur įvertinami įvairūs parametrai, kurie aprašomi VT.

**VT srautų taisyklės** priešingai nei taisyklės, sudarytos „Jei-Tai“ forma ar sprendimų lentelės pavidalu, šis taisyklių formatas suteikia galimybę sudaryti VT kaip sprendimus ir užduotis. T.y. taisyklės užrašomos kaip galimas sprendimų ir užduočių sąrašas. Šiuo atveju sprendimai aprašomi kaip sąlygų rinkinys, o užduotys aprašomos kaip veiksmai, kurie yra vykdomi priklausomai nuo tuo metu tenkinamų sąlygų. Toks sąlygų ir pasekmių (veiksmų) išskyrimas suteikia galimybę sudaryti kompleksines, sudėtines taisykles. Taip suteikiamas geresnis taisyklės supratimas bei tokia forma jos tampa geriau skaitomos. Priešingu atveju, jei šios kompleksinės taisyklės nebūtų užrašomos šiuo formatu, būtų sunku suprasti užduočių vykdymo seką. Šiuo atveju VT pateikiamos kaip nuoseklios vykdymo sekos. Panašiai kaip „UML Sequence“ diagramos.

Programinio paketo grafinės sąsajos pavyzdys, sudarinėjant nuoseklaus vykdymo formato VT pateiktas paveikslėlyje Nr.5.




5. pav. Nuosekliosios (sekų) VT (VT rinkiniai) [22]

Šis formatas yra turbūt lengviausiai suprantamas VT atvaizdavimo formatas. Jame tiesiogiai matosi, kas bus atliekama esant tam tikrom sąlygom. Pažvelgus į seką, veiklos analitikui matoma visos reikalingos VT ir lengvai suprantama VT vykdymo seka. Šis „QuickRules.NET“ metodas naudoja „MS Visio“ kaip pagrindą, todėl užtikrinamas papildomas integralumas su „MS Office“ programiniu paketu.

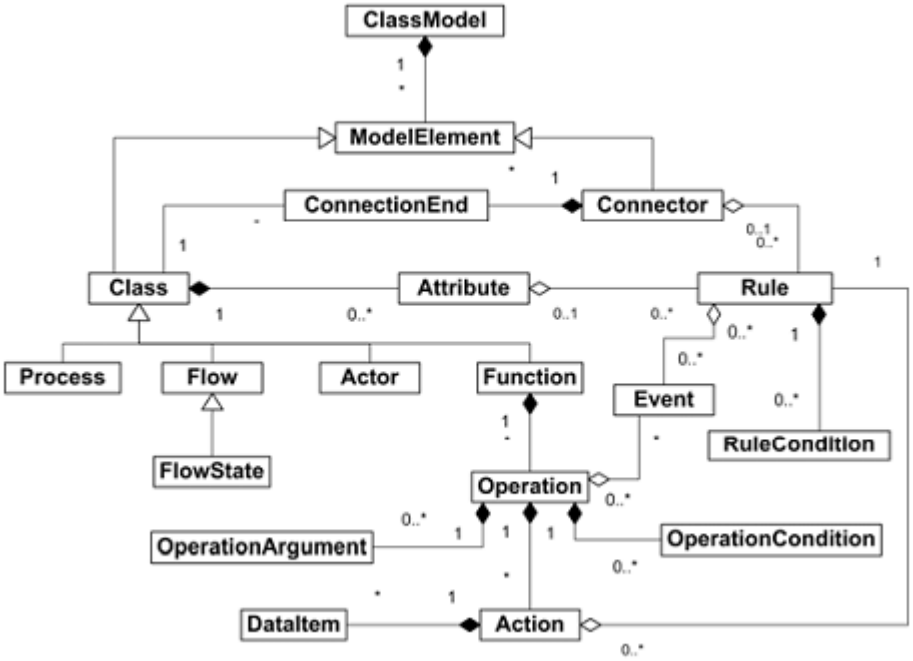
Taigi veiklos taisyklės gali būtį užrašomos įvairiais būdais ir įvairiomis priemonėmis. Kadangi yra sukurtų įvairių projektinių modelių kūrimo įrankių tai daugelyje įrankių yra stengiamasi suderinti su jau nusistovėjusiais standartais ir populiariais CASE įrankiais (pvz. MS Visio ) ar lentelių kūrimo paketais (pvz. MS Excel), taip pat yra stengiamasi realizuoti modelių eksportą ir importą XML dokumento pavidalu.

Kaip ir buvo minėta anksčiau, visi technologiniai sprendimai yra slepiami, nes šios technologijos yra kuriamos didelėms įmonėms ir yra brangios. Randamoje informacijoje būna pateiktos tik galimybės ir konkrečių VT taikymo būdų koncepcija.

## *3.6.KLASIŲ METAMODELIO IŠPLĖTIMAS VEIKLOS TAISYKLĖMIS*

Klasių modelis yra pagrindinis projektavimo modelis, kuriame yra aprašomi kuriamos IS objektai, jų atributai, funkcionalumas ir ryšiai tarp objektų. Klasių modelis eina per visus tolimesnius projektavimo etapus (PIM, PSM). Taigi šis modelis yra labai svarbus ir yra siektina kuo anksčiau išsiaiškinti kuriamos IS objektus ir juos kuo tiksliau aprašyti, nes šio modelio pataisymai labai brangūs ir ilgai trunkantys.

ISK mokslininkai pasiūlė išplėstinį klasių modelį. Šis klasės modelio išplėtimas pateiktas paveikslėlyje. Nr. 6. Šis išplėstas modelis gali būti ribojamas/papildomas veiklos taisyklėmis.




6. pav. [3] Išplėsto klasių metamodelio fragmentas (užrašytas UML)

Iš šio metamodelio yra matoma, kaip klasių metamodelis gali būti papildomas veiklos taisyklėmis (*Rule*). Šis kelių projektavimo lygių apjungimas leis efektyviau integruoti gautas žinias dalykinės srities specifikavimo etape ir leis greičiau modifikuoti klasių modelį keičiant tik veiklos taisyklių žinių bazę.

Lentelėje Nr. 2. yra pateikti veiklos taisyklių metamodelio ryšiai su išplėstiniu klasių modeliu.

Lentelė Nr. 2. VT saugyklos metamodelio ryšys su KM metamodeliu (VTSMM → KMM)

| VT saugyklos metamodelio elementas | Ryšys | Išplėstas UML KM metamodelio elementas |
|---|---|---|
| <VTSM.BRStructured> | φ1 | <KMM.Rule> |
| <VTSM.Condition> | φ2 | <KMM.RuleCondition> |
| <VTSM.Event> | φ3 | <KMM.Event> |

Toliau bus pateiktas ISK mokslininkų pasiūlytas algoritmas, kaip šiuos ryšius realizuoti.

## *3.7. PASIŪLYTAS ALGORITMAS KLASIŲ MODELIO PAPILDYMUI VEIKLOS TAISYKLIŲ PAGRINDU*

Klasių modelis (pav. 7) su veiklos taisyklių modeliu yra siejamas per klasės atributus, ryšius ir operacijas. Taigi būtina išanalizuoti šias sąsajas. Tad modelio išplėtimo veiklos taisyklėmis algoritmas susideda iš trijų pagrindinių etapų:

**1 etapas:** Klasių modelio atributų tarp klasių verifikavimas ir specifikavimas (pav. 8). Šiame etape yra aptinkami ir pridedami trūkstami atributai. Kaip ir pirmame etape yra naudojamos Apribojimų ir Faktų veiklos taisyklės.

**Algoritmo aprašymas:**

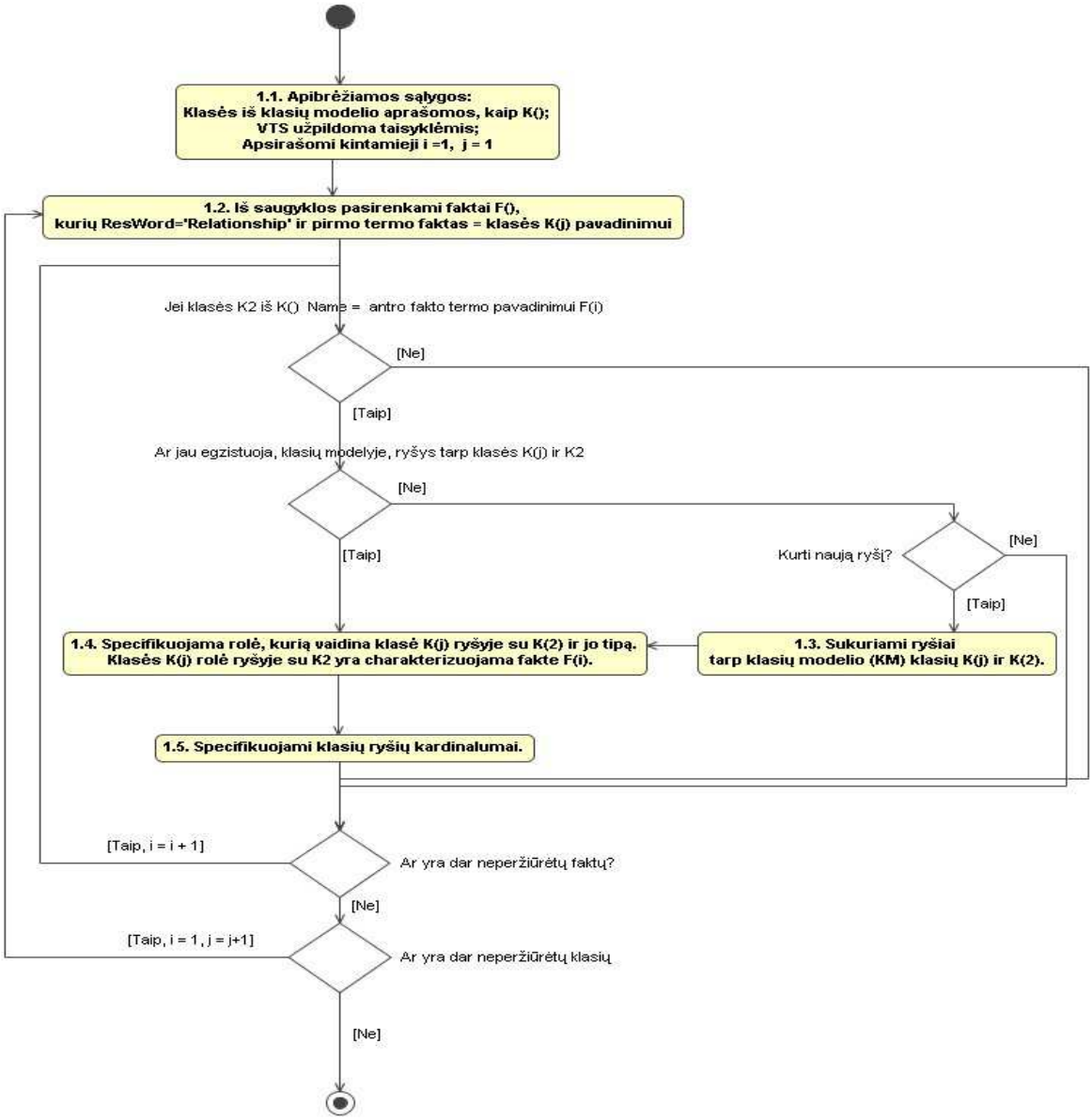
***Žingsnis 1.1*** Apibrėžiamos pradinės sąlygos.

***Žingsnis 1.2*** Faktų iš VT saugyklos pasirinkimas. Parenkami faktai F() iš veiklos taisyklių saugyklos (VTS). Parinkti faktai yra susiejami su konkrečia klase K(j) iš klasių modelio per fakto pavadinimą ir per reikšmę (ResWord), jei ji yra ryšys( angl. relationship).

Jei klasės K2 pavadinimas lygus fakto F(i) vardui einama prie kito žingsnio. Jei nėra K2 klasės atitinkančio fakto F(i) imamas sekantis faktas F(i+1) ir žingsnis kartojamas.

Jei klasių modelyje egzistuoja ryšys tarp K2 ir K(j) yra pereinama prie 1.4 arba 1.3 žingsnio.

***Žingsnis 1.3*** Sukuriami ryšiai tarp klasių modelio (KM) klasių K(j) ir K(2). Į šį žingsnį papuolama jei klasių modelyje nėra nurodytų ryšių tarp klasių K(j) ir K2. Algoritmas aptinka ir pasiūlo pridėti šį ryšį remiantis faktu F(i), kuris yra patvirtintas sistemos analitiko. Kai sukuriamas ryšys tarp klasių K(j) ir K2 pereinama prie 1.4. žingsnio.




7. pav. Ryšių tarp klasių nustatymas remiantis veiklos taisyklėmis [3]

**Žingsnis 1.4** Specifikuojama rolė, kurią vaidina klasė K(j) ryšyje su K(2) ir jo tipą.

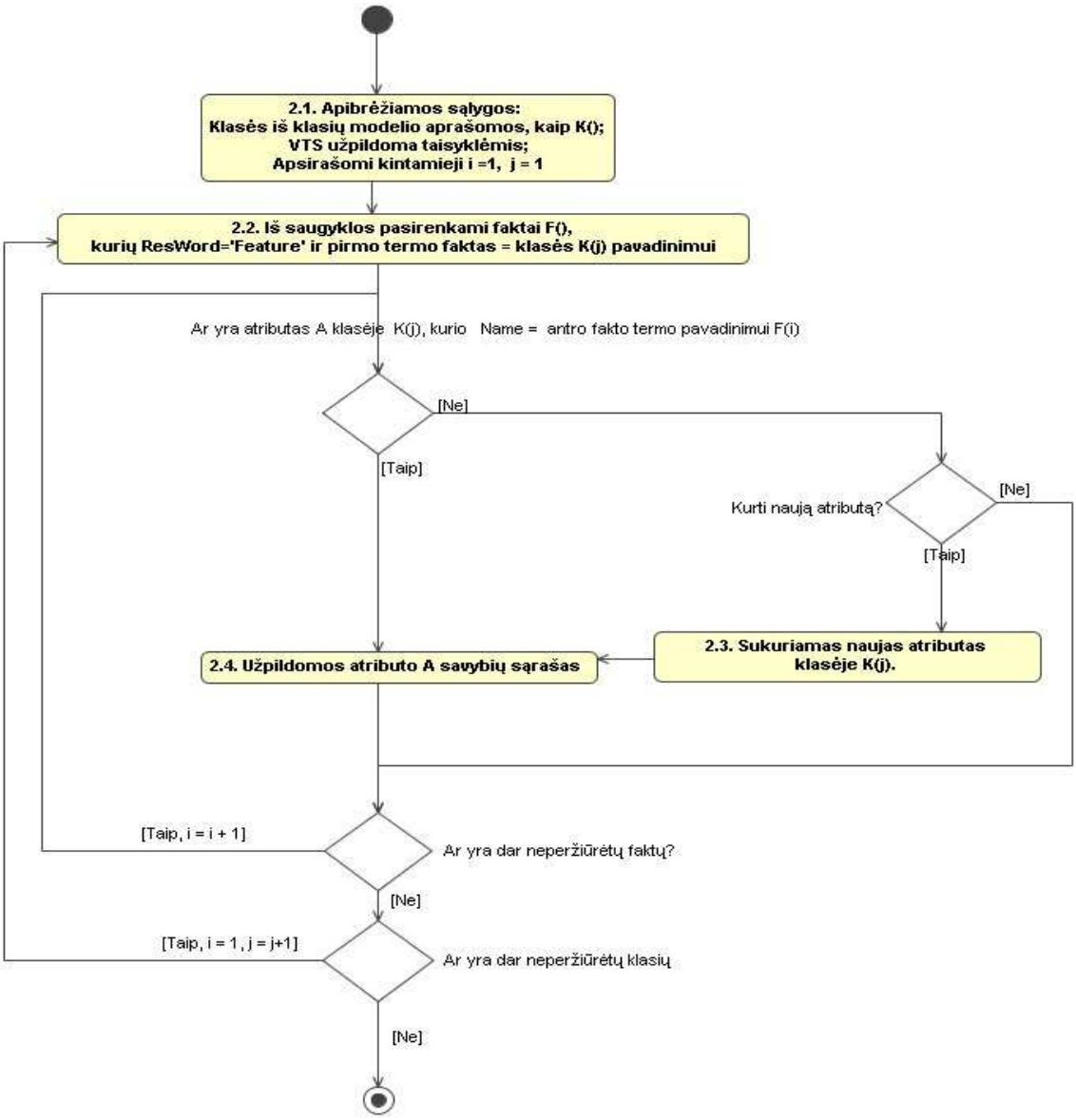
Klasės K(j) rolė (asociacija, agregacija ir t.t) ryšyje su K2 yra charakterizuojama fakte F(i) susijusiame termine.

**Žingsnis 1.5** Specifikuojami klasių ryšių kardinalumai. Ryšių kardinalumai nustatomi pagal ribojimų taisykles aprašytas VT saugykloje. Ribojimai pasirenkami pasinaudojus paprasta sąlyga – taisyklė turi paveldėti keletą ribojimų tarp dviejų verslo objektų, aprašytų klasių pavidalu klasių modelyje.

Vidinis algoritmo ciklas pavaizduotas paveikslėlyje (pav. 7) baigiasi, kai visi faktai yra peržiūrimi. Kitu atveju pradedamas naujas ciklas su nauju faktu F(i+1).

Išorinis ciklas baigiamas, kai visos klasių modelio klasės yra peržiūrėtos. Jei ne visos klasės yra peržiūrėtos pradedamas naujas ciklas su K(j+1) klase. Arba pirmas algoritmo etapas yra baigtas.

**2 etapas:** Klasių modelio ryšių tarp klasių verifikavimas ir specifikavimas (pav.9). Šiame etape yra identifikuojami trūkstami ryšiai ir modelio kūrėjo pagalba jie yra sukuriami. Šiame etape yra panaudojamos Apribojimų ir Faktų veiklos taisyklės.




8. pav. Klasių atributų lygmens papildymas remiantis veiklos taisyklėmis [3]

Antro algoritmo etapo išplėstas algoritmas pavaizduotas paveikslėlyje nr. 8. Šioje dalyje klasės yra papildomos atributais, kurie yra apibrėžti veiklos taisyklių saugykloje.

Šis algoritmas yra pasiūlytas ISK mokslininkų. Tai dalis metodų VT taikymo IS projektavimo procese. Tolimesniame darbo vystymo etape šis algoritmas gali kisti, tačiau pagrindas liks toks, kuris yra aprašytas. Šio darbo tikslas yra pasiūlyti šio algoritmo papildymus, kurie bus pasiūlyti po atliktos dalykinės srities analizės.

**3 etapas:** Klasių modelio operacijų tarp klasių verifikavimas. Tradiciniame į objektus orientuotame IS kūrime metodų (operacijų) verslo logika yra paslepiama programos kode, ko pasėkoje yra labai sudėtingas jų keitimas pagal dinamišką šiuolaikinį verslą.

Naudojantis veiklos taisyklėmis išplėstu IS kūrimo metodu, galima identifikuoti ir sekti kiekvieną kodo eilutę susijusią su verslo logika.

Skaičiavimų, išvadų ir veiksmų tvirtinimas remiantis veiklos taisyklėmis yra šio metodo branduolys.

## *3.8. KURIAMAS VEIKLOS TAISYKLIŲ PANAUDOJIMO SPRENDIMAS*

Šiame darbe veiklos taisyklės bus panaudotos projektinio klasių modelio išplėtimui bei patikslinimui CIM (arba PIM) duomenų modelio lygmenyse. Šis darbas turėtų pagerinti klasės modelio korektiškumą, nes jo patikslinimui ir praplėtimui bus naudojamos veiklos taisyklės užrašytos natūralios kalbos šablonais. Kadangi veiklos taisyklės bus užrašomos natūralios kalbos šablonų pagalba, tai specifikuota sistema bus vieningai interpretuojama tiek užsakovo tiek informacinės sistemos projektuotojo. Kuriamas sprendimas analizuos struktūrizuotas veiklos taisykles veiklos taisyklių saugyklos duomenų bazėje ir sulygins duomenų bazėje ir klasių modelyje aprašytus klasių modelio elementus. Šiame darbe bus praplėstas klasių modelio metamodelis, kuris bus papildytas atributo, ryšio ir klasės *BusinessRule* elementu. Kuriamas sprendimas turėtų pagreitinti informacinės sistemos kūrimo procesą, sumažindamas kūrimo kaštus ir padidindamas kuriamo produkto kokybę.

## *3.9. KITI VEIKLOS TAISYKLIŲ PANAUDOJIMO SPRENDIMAI*

Veiklos taisyklės yra naudojamos ne tik projektinių modelių automatiniam generavimui, tačiau ir kituose procesuose, kurių dalis bus apžvelgta toliau. Veiklos taisyklės yra naudojamos:

**Informacinės sistemos analizės ir specifikavimo etape[13]**

Daugelyje tradicinių programinės įrangos kūrimo gyvavimo ciklo analizės etapas, yra orientuotas į sistemos reikalavimų rinkimą. Veiklos taisyklių požiūriu yra kitoks mąstymo būdas apie analizės procesą. Užuot sutelkiant dėmesį į sistemos reikalavimus, analizavimas yra nukreipiamas į veiklos procesą aprašant jį veiklos taisyklėmis. Vietoj vartotojų klausinėjimo, ko jie nori, kad jų sistema darytų, yra klausinėjama apie jų įmonės ar organizacijos veiklą.

Taip pat veiklos taisyklės padeda susistemintai ir struktūriškai aprašyti tiriamos organizacijos verslo logiką ir veiklos taisykles, kurios yra vienareikšmiškai suprantamos dalykinės srities atstovų ir informacinių sistemų kūrėjų. Tam yra naudojamos veiklos taisyklių valdymo sistemos.

**Veiklos taisyklių naudojimas darbo sekų sistemose [14][20]**

Į procesą orientuotos darbo sekos naudoja veiklos taisykles, kurios neišvengiamai įtakoja sprendimus priimamus per tą darbų seką. Tokio pobūdžio taisyklės yra įtrauktos į sekos taisyklių rinkinius ir taisyklių mechanizmas yra panaudojamas, konkretaus proceso žingsnio (veiksmo) apribojimui ir sąlygų vykdymui. Taip pat veiklos taisyklės gali vaidinti duomenų filtro vaidmenį, kurio pagalba bus leidžiami tik konkretūs duomenys dalyvaujantys veiklos procese.

**Veiklos taisyklių panaudojimas į paslaugas orientuotų sistemų architektūroje [15]**

James Taylor aprašė kaip veiklos taisyklės gali būti panaudojamos modernios programinės įrangos architektūroje. Jis aprašė septynis būdus, kaip panaudoti veiklos taisykles. Šie būdai yra:

***Sprendimų priėmimo komponentai*** **-** komponentas, naudojamas į sprendimą orientuotose programose. Taip pat jam turi būti suprojektuotas ir sukurtas sprendimo servisas. Jis gali būti dažnai keičiamas. Daugeliu požiūriu tai yra klasika "taisyklė-paslauga" veiklos taisyklių realizavime.

***Sprendimų veikla*** **-** veiklos procese yra sprendimas. Veiklos turėtų būti aiškiai identifikuojamos veiklos procese atskirai, nes gali kilti sinchronizavimo ir kitų problemų.

***Įvykių sprendimai*** **-** įvykis reikalauja sprendimo priėmimo. Tai galėtų būti paprastas įvykio fiksatorius arba susietas su ekonominės veiklos stebėjimo ar kompleksinių įvykių apdorojimo programinę įranga, kuri apdoroja daug inicijuotų procesų ir palaiko daug įvykių ir naudojasi sprendimo servisu, kai jis yra numatomas veikloje.

***Kompleksinis maršrutizavimo*** **-** vartų/šakojimosi mazgų procesas gali naudoti kompleksinių sprendimų mechanizmą - šiuo atveju bus ruošiamasi gauti maršruto taisykles naudojantis sprendimų paslauga.

***Liktinis požiūris*** **-** dalis liktinės sistemos turi palaikyti servisus. Ji yra orientuota į logiką ir nėra apkrauta duomenimis. Praktiškai tai reiškia, kad jūs turite kokią nors verslo logiką, kuri yra dalis jūsų senos programos ir ji yra jau pasenusi, tad ją reikia pakoreguoti. Šiuo atveju tai galima padaryti renovuojant šią logiką pasinaudojus veiklos taisyklių valdymo sistema. Perrašant taisykles į VTVS, sugeneruoti veiklos servisus ir leisti jais naudotis senoje sistemoje.

***Intensyviose transformacijose*** **-** yra aplinkybių, kuriomis yra intensyviai transformuojama įmonės informacija. Pavyzdžiui, daugelyje sluoksnių persidengiantys finansiniai dokumentai gali būti pertvarkomi iš vietos apskaitos įrašų į centralizuotus. Sprendimų paslaugos palaikančios šias transformacijas gali būti daug efektyvesnės ir lankstesnės nei taisyklių naudojimas techniniame lygyje.

***Verslo sprendimų priėmimas*** **-** vis daugiau kompanijų mato naudą apdorojant visus sprendimus, naudojantis verslo sprendimų priėmimų pagrindais arba universaliųjų sprendimų

priėmimo mechanizmu, kuris valdo visas operatyvius verslo sprendimus kaip paslauga ir daro jas prieinamas visiems kanalams ir sistemoms.

**Veiklos taisyklės ir verslo įžvalga [16]**

Veiklos taisyklės aprašo bet kokią veiklos valdymo sistemą ir bet kurį verslo įžvalgos (VĮ) projektą. Jos leidžia ataskaitų mechanizmams automatiškai interpretuoti duomenis, siekiant nustatyti tikslinius pagrindinės veiklos vykdymo rodiklius ir pasiūlyti priemones, užkertančias problemas.

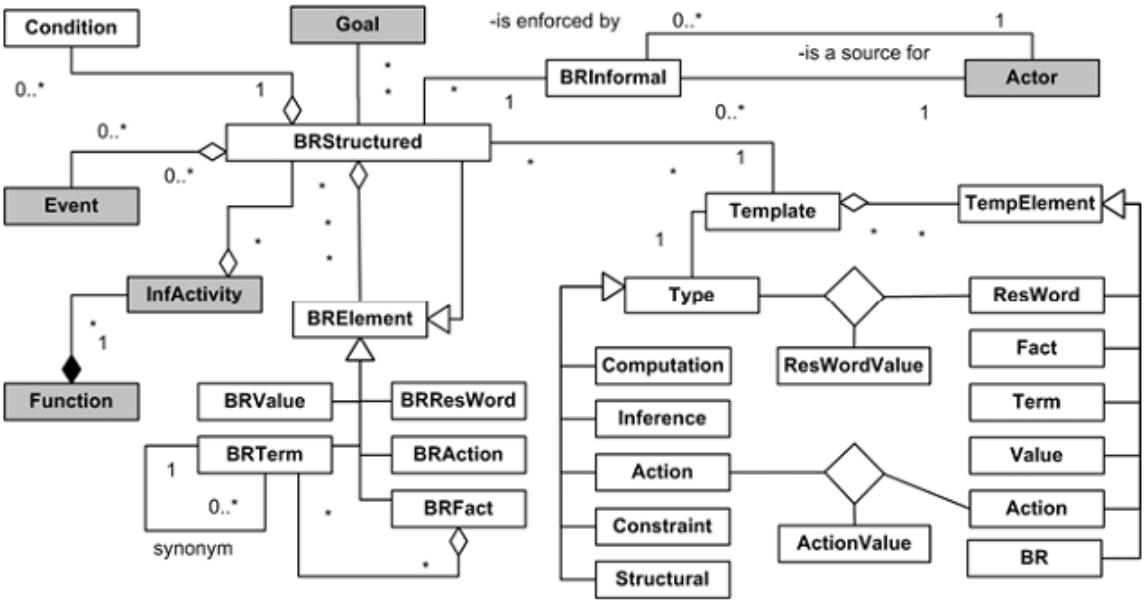
Atlikus veiklos taisyklių panaudojimo analizę galima teigti, jog jų panaudojimo galimybės yra labai plačios. Taigi veiklos taisyklės yra naudojamos nuo reikalavimų rinkimo proceso susisteminimo ir reikalavimų struktūrizavimo iki verslo įžvalgos procesų aprašymų.

Taip pat galima daryti išvadą, kad veiklos taisyklės būtų lanksčiai naudojamos - ši programinė įranga turi palaikyti paslaugas (angl. services). Kadangi veiklos taisyklių aprašomi procesai daugeliu atveju yra aprašinėjami paslaugomis, tai šios paslaugos galės būti panaudojamos ne viename taikomajame uždavinyje ir pakeitus paslaugos logiką visose programose pasikeis šis funkcionalumas.

## *3.10. VEIKLOS TAISYKLIŲ SAUGYKLA*

Veiklos taisyklės gali būti saugomos išorinėse duomenų bazėse, duomenų failuose (XML), OLE duomenų bazėse (MS Excel, MS Visio) ir kitokiais informacijos saugojimo būdais. Veiklos taisyklių saugykloje [3][2][19] yra saugomos veiklos taisyklės, kadangi dažniausiai vartojama duomenų bazių valdymo sistema (DBVS) (MS SQL Server, MySQL, Oracle ir t.t. ) yra saugomos ir vartotojų (VT kūrėjų) teisės, ryšiai tarp taisyklių ir kiti specifiniai struktūrizuoti duomenys susiję su VT.

ISK mokslininkai savo tyrimuose pasiūlė tokį veiklos taisyklių saugyklos modelį (pav. Nr. 9)[3].




9. pav. [3] Veiklos taisyklių saugyklos modelis (VTSM) (užrašytas UML)

Šiame VT saugyklos metamodelyje yra išskirtos pilkos figūros, kurios parodo, kaip pasiūlyta koncepcija yra susiejama su verslo modelio metamodeliu.

Kaip matome iš metamodelio, tai struktūrizuotos veiklos taisyklės gali susidėti iš pačių veiklos taisyklių, sąlygų, įvykių, VT elementų (VT reikšmių, VT terminų, VT faktų, VT veiksmų, VT verslo terminų), informacijos kokiomis sąlygomis yra taikoma VT.

Pagrindinis veiklos taisyklės struktūrinis elementas yra rezervuotas žodis (*BRResWord*).

Rezervuotas žodis aprašo VT elementų tarpusavio reikšmę arba tipą. Reikšmei gali būti priskiriamos ryšių savybės: asociacija, agregacija, kompozicija ar generalizacija.

Kadangi yra kuriama VT saugykla, kurioje bus galima VT užrašyti natūralios kalbos šablonais, tai šioje saugykloje yra ir taisyklių saugojimas šablonų pavidalu (aprašytu skyriuje 10.4.).

Veiklos taisyklių saugykla, pateikta ankstesniame paveikslėlyje Nr. 6, yra viena iš pagrindinių algoritmo realizacijai reikalingų dalių, kurioje bus saugomos struktūrizuotos veiklos taisyklės.

## *3.11. VEIKLOS TAISYKLIŲ VALDYMO SISTEMŲ (VTVS) NAUDOJIMAS*

Veiklos taisyklių valdymo sistemos (toliau VTVS) yra skirtos padėti organizacijoms tobulinti ir atnaujinti verslo taikomąją programinę įrangą greičiau ir suteikti verslo ekspertams didesnę kontrolę ir galimybę valdyti pažangaus verslo sprendimų įtakojamus programinės įrangos funkcionalumo pakeitimus.

Naudojant VTVS, bendra taikomųjų programų kaina yra mažinama, mažinant programavimo reikalavimus ir vėlavimų įgyvendinant veiklos taisyklėmis aprašytą funkcionalumą. Be to VTVS

suteikia galimybę kontroliuoti veiklos taisykles, kas yra nauja. Daugeliu atvejų įmonės verslo taisyklės yra aprašytos vadovuose ar kituose dokumentuose, arba visai nėra oficialiai apibrėžtos, nes daugelis organizacijų pasikliauja verslo valdytojų interpretuojama kompanijos politika, kaip verslo praktikos taisyklėmis. Tai gali būti neveiksminga ir sukelti nesuderinamumų taisyklių taikymo metu.

VTVS leis taisykles veiksmingai ir nuosekliai taikyti visuose sąlyčio taškuose, kur jos yra naudojamos. Didžiausią naudą VTVS duos toms organizacijoms, kurios turi daug klientų, ir jų veikla yra labai dinamiška, kur didelę įtaką turi operatyvių sprendimų priėmimas, kurie gali būti pakartojami ir apibendrinami. Taip pat tikėtina nauda yra plataus masto nacionalinėms ir daugiašalėms įmonėms, kuriose vyrauja nevienalytė IT platforma, ir daug liktinių sistemų.

VTVS yra naudinga, jei organizacijoje vyrauja bent vienas iš šių faktorių:

- **Labai sudėtingos taisyklės, kurias yra sudėtinga įgyvendinti kodo pagalba**

  VTVS taikymas leidžia kūrėjams atskirą verslo logiką nuo procedūrinio kodo. Kada verslo logikoje yra sudėtingų sąlygų, kurios turi būti įvertintos, arba tais atvejais, kai taisyklės keičia informaciją, kurią inicijuoja kitos taisyklės, kurios turi būti įvertintos. Tuomet kodo rašymas siekiant įgyvendinti taisykles yra sudėtingas procesas linkes į klaidingą realizavimą. Pasinaudojant VTVS išvedimo mechanizmu galima sparčiau realizuoti verslo logikos valdomus išvedimo procesų scenarijus.

- **Kai yra daug taisyklių ir svarbus jų administravimas**

  VTVS siūlo tvirtą struktūrinę saugyklą, leidžiančią daug taisyklių, kurios yra valdomos, versijuojamos, audituojamos ir pakartotinai naudojamos, kaip atominiai vienetai. Skirtingai nuo programino kodo, atskiri deklaratyvūs vienetai, gali būti valdomi dinamiškai. Taisyklių, kurios turėtų būti pakartotinai naudojamos arba kurias gali tekti dažnai keisti, nustatymas ir naudojimas yra paprastesnis, ir todėl programinės įrangos išlaikymo kaštai yra mažesni.

- **Kai taisyklės keičiasi taip dažnai, kad lankstumas ir vartotojo galimybės jas kontroliuoti yra svarbiausias veiksnys**

  Vykdant pakeitimus turi būti ne tik VTVS leidžiama keisti logiką, bet turi būti nesugadinamas programinės įrangos funkcionalumas. Taip pat turi būti leidžiama tiesiogiai atlikti reikiamus pakeitimus. VTVS leis pakeisti taisykles ir jas išbandyti VT saugykloje naudojantis patogia vartotojo sąsaja ir tuomet bus galima šį pakeitimą patalpinti į veikiantį taisyklės įtakojamą procesą taip, kad kviečiant šį procesą bus panaudojama naujų taisyklių apibrėžta logika, be programinė įrangos perkrovimo. Tai sumažina laiką tarp pareikalauto pakeitimo inicijavimo ir jo realizacijos.

- **Kai taisyklės reikalauja verslo ar techninės patirties joms suprasti arba redaguoti**

  Tais atvejais, kai taisyklėms interpretuoti, suprasti ir jas redaguoti yra reikiamos teisinės, medicininės ar kitos specifinės verslo žinios, VTVS leidžia ne techninio pobūdžio įmonių vadovams ir analitikams išlaikyti taikymo logiką. VTVS tai gali padaryti nepriklausomai nuo techninių išteklių, net jei jie yra pakeitimų testavimo ir pakeitimų stebėjimų apribojimuose. Tai sumažina programinės įrangos modifikacija laiką ir taip pagreitina organizacijos reagavimo laiką į pasikeitusias sąlygas. Ji taip pat sumažina bendrą programinės įrangos kainą bei pašalina klaidas ir nesusipratimus.
- **VTVS naudojimo nauda gali būti [4]**:
    - 25% - 75% laiko taupymas per visą taikomosios programos gyvavimo ciklą.;
    - spartesnis atsakas į pokyčius;
    - sistemos gali būti atnaujintos su minimaliais IT ištekliais;
    - individai, darbuotojai arba darbdaviai, gali keisti savo programinės įrangos elgseną;
    - puikus aptarnavimas reaguojant į atitinkamą situaciją;
    - verslo politikos panaudojimas taikomosiose programose ir kitose verslo srityse;
    - verslo duomenų panaudojimas įmonės taikomosiose programose;
    - žemesnė programinės įrangos gyvavimo ciklo kaina;
    - sutaupyti resursai reikalavimų rinkimo ir modeliavimo darbams;

## *3.12. ANALIZĖS IŠVADOS*

1. Analizės metu buvo išanalizuotos veiklos taisyklės, jų tipai ir savybės. Taigi galima teigti, jog veiklos taisyklės visuomet buvo ir yra naudojamos realiame gyvenime, tačiau jų struktūrizavimas, saugojimas ir analizavimas bei pritaikymas technologiniuose procesuose yra labai sudėtingas procesas ir jis vis dar yra kūrimo stadijoje. Dalies galimų veiklos taisyklių užrašymo būdų analizė parodė, jog dalykinės srities atstovui (ne IT specialistui) labiausiai priimtinas veiklos taisyklių užrašymas natūralios kalbos šablonais.
2. Išanalizavus veiklos taisyklių taikymą galime teigti, jog sistemos, kurios naudos veiklos taisykles, kaip logikos ir ribojimų pagrindą, turi būti orientuotos į paslaugas ir konkrečios taisyklės turi būti realizuotos, kaip atskiros paslaugos. Kadangi veiklos taisyklės bus realizuotos, kaip paslaugos, tai jas galima bus naudoti ne viename taikomajame uždavinyje ir pakeitus veiklos taisyklės savybes, visų taikomųjų uždavinių ribojimai ar logika pasikeis.
3. Buvo išanalizuoti veiklos taisyklių valdymo sistemų naudojimo atvejai ir šių sistemų privalumai. Automatinis veiklos taisyklių taikymas be veiklos taisyklių valdymo sistemų yra

komplikuotas. Šios valdymo sistemos padės struktūrizuotai surinkti ir ištestuoti veiklos taisykles, taip pat jos turėtų aptikti prieštaraujančias taisykles ir jas drausti.

4. Buvo išanalizuoti keli CASE įrankiai, kuriuose buvo realizuoti veiklos taisyklių taikymai. Kadangi IS kūrimas remiantis žinių baze yra labai jauna sritis, nėra griežtų standartų apibrėžiančių nei veiklos taisyklių užrašymo būdus, nei jų taikymą. Taigi sukurti CASE įrankiai slepia savo sukurtus veiklos taisyklių taikymo ir kaupimo algoritmus.
5. Kadangi informacinių sistemų katedra vykdo mokslinius tyrimus IS projektavimo remiantis žinių baze, buvo pasirinktas jų sukurtas algoritmas leisiantis išplėsti/apriboti klasės modelį veiklos taisyklių pagrindu. Šis algoritmas bus tik vienas iš kelių etapų žinių bazės analizavimo ir taikymo projektinių modelių automatinio kūrimo ir papildymo procese.

# 4. KURIAMO ALGORITMO PROJEKTINĖ SPECIFIKACIJA

## *4.1. TECHNINIAI TIKSLAI IR UŽDUOTYS*

- Suprojektuoti papildytą ISK mokslininkų algoritmą.
- Pasirinkti realizacijai naudojamas technologijas.
- Koks MagicDraw UML naudojamas klasių modelis?
- Ką reikia praplėsti MagicDraw UML klasių modelyje, užsibrėžtam tikslui pasiekti?
- Išsiaiškinti koks yra MagicDraw UML įskiepių kūrimo mechanizmas.
- Išsiaiškinti, kaip galima praplėsti MagicDraw UML modelių metamodelius.
- Suprojektuoti sąsają tarp išplėsto klasės modelio metamodelio ir veiklos taisyklių saugyklos metamodelio.
- Suprojektuoti MagicDraw UML plėtinį, kuris patikslintų išplėsto klasių modelio elementų savybes.

## *4.2. REALIZACIJAI NAUDOJAMOS TECHNOLOGIJOS*

Darbo realizacijai bus naudojamos laisvai pasirinktos technologijos, arba technologijos, kurios buvo naudojamos su darbu susijusiose sistemose.

Kadangi veiklos taisyklių saugykla (VTS) yra realizuota Microsoft (toliau MS) SQL Express 2005, tai bus naudojama MS SQL Express duomenų bazių valdymo sistema. Algoritmo funkcionalumas susijęs su duomenų atrinkimu iš duomenų bazės bus realizuotas MS SQL „Stored Procedure“ technologija, kuri leis duomenų bazės valdymo sistemos pagalba koreguoti šį algoritmo funkcionalumą.

MagicDraw UML yra sukurta JAVA technologijos pagrindu, todėl bus naudojamas laisvai pasirenkamas labai patogus JAVA kalbos programavimo sąsajos paketas Eclipse.

Darbui su MS SQL duomenų baze, kurioje laikomos veiklos taisyklės, bus naudojama „Microsoft SQL Server 2005 JDBC Driver“ tvarkyklė, kurios pagalba JAVA kalboje kuriamas įskiepis susijungs MS SQL duomenų baze.

Kuriamos sistemos projektavimui bus naudojamas MagicDraw UML paketas, kuris jau eilę metų yra pripažįstamas vienas geriausių projektavimo įrankių.

### 4.2.1. PAPILDYTO ALGORITMO APRAŠYMAS

Atlikus pasiūlyto ISK mokslininkų klasių modelio išplėtimo algoritmo analizę buvo nuspręsta jį papildyti ir pakeisti keletą realizacinių aspektų.

Algoritmo papildymo pasiūlymai:

1. Vartotojams turi būti leidžiama riboti analizuojamų klasių aibę, kuri yra sukaupta VT saugyklos duomenų bazėje.
2. Vartotojams turi būti leidžiama pasirinkti, kokius klasės modelio elementus analizuoti.
3. Vartotojai turi pasirinkti konflikto sprendimo atvejus (prioritetą suteikti VT ar klasės modeliui).
4. Pirmiausia reikia surinkti visus klasės tipo objektus iš taisyklių saugyklos, o tik vėliau juos analizuoti.
5. Algoritmas galės kurti ne tik atributus ir ryšius, bet ir pačias klases.
6. Yra kuriami ne tik ryšiai ir atributai, bet ir pildomas konkrečios klasės *BusinessRule* elementas, kurio pagalba kito iškvietimo metu bus nebetikrinamos jau realizuotos veiklos taisyklės.
7. Pirmiausia yra išrenkamos konkretaus klasės tipo objekto veiklos taisyklės, o vėliau jas analizuojant yra kuriami ryšiai arba atributai.
8. Analizuoti rolės taisykles, kurios tipizuoja ryšius.

Atlikus šiuos pakeitimus, šis algoritmas taps lankstesnis ir bus taupomi techninės įrangos resursai, kas yra labai aktualu dirbant su dideliu duomenų kiekiu. Kadangi bus leidžiama pasirinkti analizuojamų elementų aibę, tai nepasirinkus *BusinessRule*, šis algoritmas gali būti pritaikomas ne tik išplėstam klasių metamodeliui. Šis nagrinėjamų elementų aibės ribojimas padidina šio algoritmo panaudojimą, nes nereikalaujamas išplėsto klasių modelio profilis.

## Algoritmo aprašymas

Algoritmo iliustracija yra pateikta paveikslėlyje nr. 10. Algoritmo dalis sprendžianti konfliktus, susijusius su kuriamų ryšių prieštaravimais, yra išskirta raudonu ovalu ir detaliau pavaizduota paveikslėlyje nr. 11.

**Žingsnis 1:** Gaunamas visų klasių sąrašas iš DB pagal ResWordMeaning ="Feature" arba "Realationship". Šis veiksmas yra realizuojama duomenų bazės valdymo sistemoje „Stored Procedure“ pagalba.

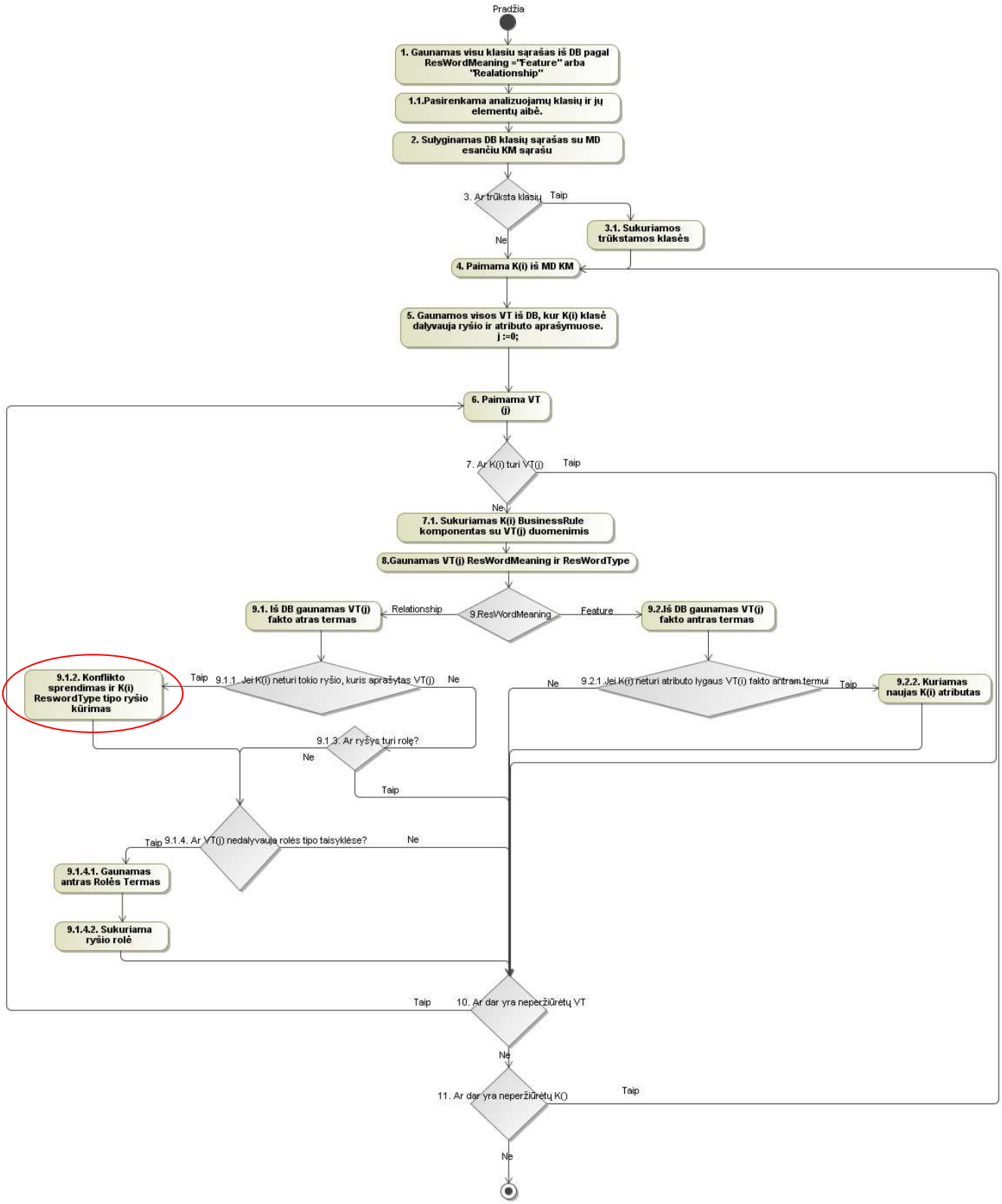
**Žingsnis 1.1:** Pasirenkama analizuojamų klasių ir klasės analizuojamų elementų aibė.

**Žingsnis 2:** Sulyginamas DB klasių sąrašas su MD esančiu KM sąrašu.

**Žingsnis 3:** Tikrinama, ar trūksta MD esančiu, KM klasių. Jei trūksta, einama į 3.1 žingsnį.

**Žingsnis 3.1:** Sukuriamos trūkstamos klasės.

**Žingsnis 4:** Vykdomas ciklas, kuris pereina per visas MD KM‘io klases. Paimama K(i).




10. pav. Siūlomas algoritmas.

**Žingsnis 5:** Gaunamos visos VT iš DB, kur K(i) klasė dalyvauja ryšio ir atributo aprašymuose. Tam naudojama duomenų bazės valdymo sistemoje aprašyta procedūra. Priskiramas pradinės analizuojamos veiklos taisyklės indeksas klasės veiklos taisyklių masyve: j :=0;

**Žingsnis 6:** Vykdomas ciklas, kuris pereina per visas VT priklausančias klasei. Paimama VT(j).

**Žingsnis 7:** Tikrinama, ar klasė K(i) turi taisyklę VT(j). Jei ne, einama į 7.1.

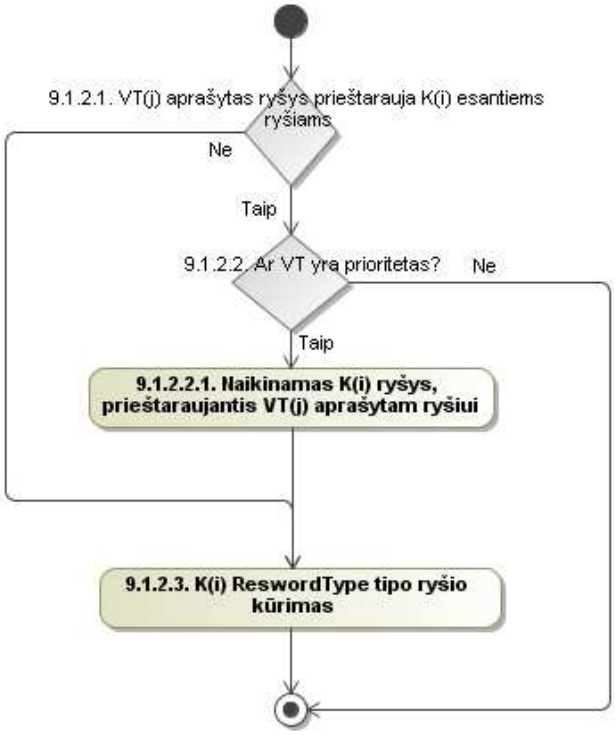
**Žingsnis 7.1:** Sukuriamas K(i) *BusinessRule* komponentas su VT(j) duomenimis.

**Žingsnis 8:** Gaunamas VT(j) *ResWordMeaning* ir *ResWordType*.

**Žingsnis 9:** Tikrinama *ResWordMeaning* reikšmė. Jei „*Relationship*“ yra, einama į 9.1.1 Jei “*Feature*” einama į 9.2.1

**Žingsnis 9.1.1:** Tikrinama, ar K(i) neturi tokio ryšio, kuris aprašytas VT(j). Jei neturi einama į 9.1.2. arba į 10.

**Žingsnis 9.1.2:** Konflikto sprendimas ir K(i) ReswordType tipo ryšio kūrimas (pav. 11) .




11. pav. Konfliktų sprendimo algoritmas

**Žingsnis 9.1.2.1:** Tikrinama, ar VT(j) aprašytas ryšys prieštarauja K(i) esantiems ryšiams? Jei ne, eina į 9.1.2.3 žingsnį. Jei taip, einama į 9.1.2.2 žingsnį.

**Žingsnis 9.1.2.2:** Tikrina pasirinktą konflikto sprendimo būdą: ar VT yra prioritetas? Jei taip eina į 9.1.2.2.1 žingsnį. Jei ne, eina į 9.1.3 žingsnį.

**Žingsnis 9.1.2.2.1:** Naikinamas K(i) ryšys, prieštaraujantis VT(j) aprašytam ryšiui.

**Žingsnis 9.1.2.3:** K(i) ReswordType tipo ryšio kūrimas.

**Žingsnis 9.1.3:** Tikrinama, ar ryšys jau turi rolę? Jei taip, einama į 10. Jei ne, einama į 9.1.4.

**Žingsnis 9.1.4:** Tikrinama, ar VT(j) nedalyvauja rolės tipo taisyklėse? Jei ne, einama į 10-ą žingsnį. Jei taip, einama į 9.1.4.1.

**Žingsnis 9.1.4.1:** Gaunamas antras rolės tipo veiklos taisyklės terminas, kuris reiškia ryšyje dalyvaujančios klasės rolę.

**Žingsnis 9.1.4.2:** Sukuriama ryšio rolė.

**Žingsnis 9.2.1:** K(i) neturi atributo lygaus VT(i) fakto antram terminui. Jei neturi, einama į 9.2.2. Jei turi, einama į 10-ą žingsnį.

**Žingsnis 9.2.2:** Kuriamas naujas K(i) atributas, kurio pavadinimas lygus VT(i) fakto antram *Termui*.

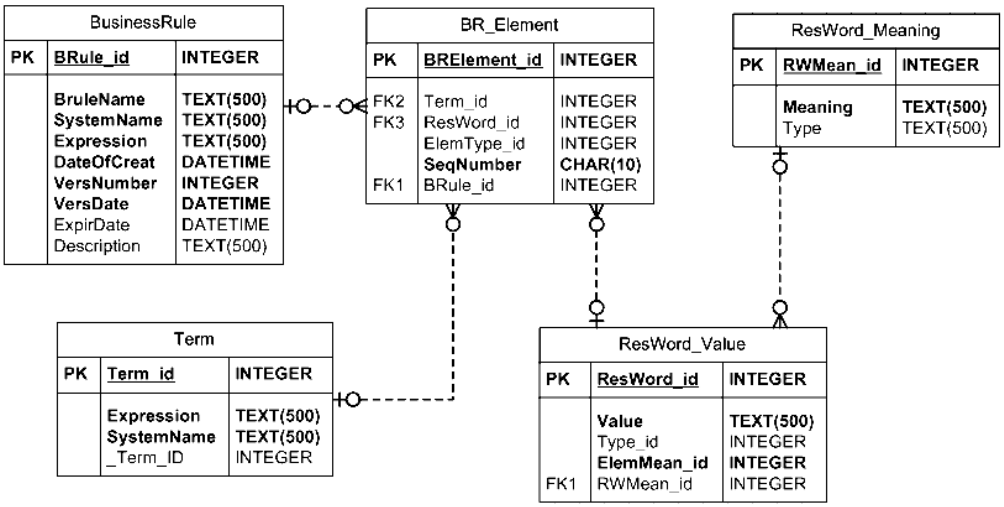
**Žingsnis 10:** Tikrinama, ar dar yra neperžiūrėtų VT. Jei yra, grįžtama į 6-ą žingsnį. Jei visi faktai susiję su K(i) peržiūrėti, einama į 11-ą žingsnį.

**Žingsnis 11:** Tikrinama, ar dar yra neperžiūrėtų K() klasių. Jei yra, grįžtama į 4-ą žingsnį. Jei visos klasės peržiūrėtos, tuomet yra baigiamas algoritmas.

Kadangi šis algoritmas ne tik papildo klasių modelį, bet ir gali jį sugeneruoti (kuria klases), tai jo funkcionalumas yra praplečiamas ir jis galės atskirai funkcionuoti be išanalizuoto klasių modelio generavimo veiklos modelio pagrindu. Kadangi šis algoritmas nebus priklausomas nuo veiklos modelio, tai juo galės naudotis ne tik įmonės veiklą modeliuojantys specialistai, bet ir kiti informacinių sistemų kūrėjai. Kadangi veiklos taisyklėmis yra labai patogu surinkti kuriamos sistemos reikalavimus, o panaudojus sukurtą algoritmą iš jų galima automatiškai gauti klasių modelį, tai šis algoritmas turėtų ženkliai sumažinti informacinės sistemos kūrimo kaštus bei trukmę. Taip pat šiame algoritme atsirado klasių modelio realizuoto MagicDraw UML konkrečios klasės dalyvaujančios veiklos taisyklėje „BusinessRule“ atributų užpildymas. Šis atributas atsirado praplėtus klasės atributą *Constraint*. Pasiūlytame algoritme nėra ryšio kardinalumo patikrinimo, tačiau nagrinėjant tik fakto ir rolės tipo veiklos taisykles, kardinalumas nėra gaunamas. Šis algoritmas išplečia klasių modelį, tačiau jo neapriboja – nesukuriami ryšių kardinalumai.

## *4.3. VT SAUGYKLOS DUOMENŲ BAZĖS PROJEKTAS*

Veiklos taisyklių saugojimui yra sukurta duomenų bazė, kuri bus analizuojama algoritmo vykdymo metu. Ši veiklos taisyklių saugyklos dalis yra paimta iš ISK mokslininkų pasiūlytos veiklos taisyklių saugyklos duomenų bazės [10]. Kadangi kuriamam algoritmui nereikia visos saugyklos, šiame darbe yra panaudojama tik jos dalis (pav. 12), kurios reikės algoritmo funkcionalumui.




12. pav. Loginė duomenų bazės fragmento schema

Ši duomenų bazė bus kuriama MS SQL Express 2005 duomenų bazių valdymo sistemoje. Duomenų bazėje taip pat bus realizuota ir dalis algoritmo funkcionalumo. Tai bus atlikta Stored procedūrų pagalba.

Bus sukurtos 5-ios Stored procedūros, kurių aprašymas pateiktas lentelėje nr. 3.

Lentelė Nr. 3. Stored procedūrų aprašymai

| **Pavadinimas** | **Įėjimai** | **Gražinami duomenys** | **Aprašymas** |
|---|---|---|---|
| getTermId | Term. *SystemName* | Term.Id | Gaunamas termo indentifikatorius. |
| sBROfObject | Term. *SystemName* | BusinessRule.ID, BusinessRule.Name, BusinessRule.Expression | Gaunamos objekto taisyklės. |
| sBRResWord | BusinessRule.ID | ResWord_Meaning.Meaning, ResWord_Meaning.Type | Gaunamas taisyklės rezervuotas žodis, pagal kurį nustatoma ar šia taisykle bus kuriamas ryšys ar atributas. |

| | | | |
|---|---|---|---|
| sBRSecondterm | BusinessRule.ID | Term.SystemName | Gaunamas atributo arba klsės pavadinimas su kuria yra aprašomas ryšys. |
| sObjects | | Term.SystemName | Gaunamas visų klasių sąrašas. |

Stored procedūros yra naudojamos tam, kad būtų atskirtas duomenų bazės ir programinis sluoksniai. Šios procedūros atlieka duomenų atrinkimą. Kadangi jos vykdomos duomenų bazių valdymo sistemoje, tai duomenų atrinkimas yra spartesnis, nei būtų kreipiamasi iš programinio lygmens. Naudojant šią technologiją greičiau vykdomi funkcionalumo pakeitimai ir jie yra atliekami nekeičiant programinio kodo. Procedūromis gali naudotis kelios programos ir pakeitus jų funkcionalumą visose programose bus atnaujintas funkcionalumas.

## *4.4.KURIAMO ĮSKIEPIO NEFUNKCINIAI REIKALAVIMAI*

Prieš kuriant įskiepį reikia apsibrėžti jo nefunkcinius reikalavimus tam, kad būtų apibrėžtos jo kokybės ir santykio su išoriniais veiksniais gairės.

Kuriamas produktas turi:

- nesugadinti sistemos į kurią jis yra integruojamas;
- nesugadinti duomenų veiklos taisyklių saugyklos bazėje;
- būti nesudėtingai naudojamas;
- lengvai iškviečiamas;
- turėti galimybę būti paleidžiamas kelis kartus;
- būti lengvai pašalinamas.

## *4.5. KURIAMO ĮSKIEPIO FUNKCINIAI REIKALAVIMAI*

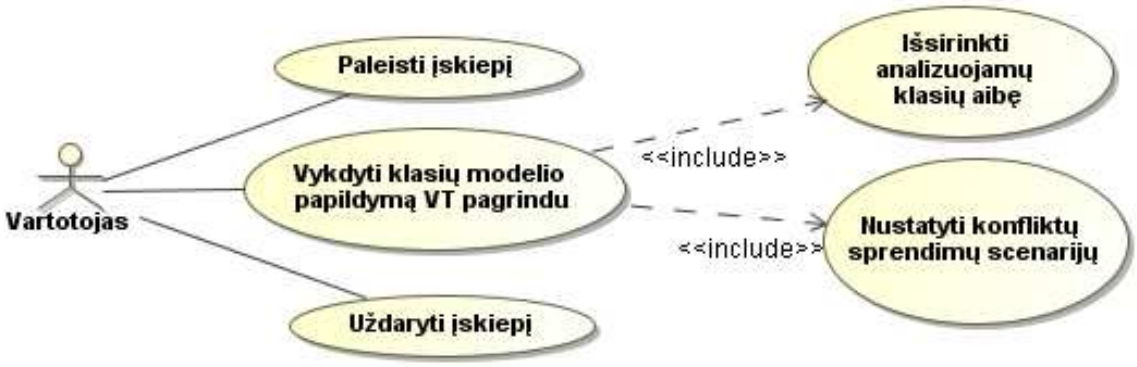
Prieš kuriant įskiepį reikia apsibrėžti jo funkcijas. Šios funkcijos aprašomos funkciniais reikalavimais. Įskiepis prvalo turėti tokias funkcijas:

- būti paleidžiamas mygtuko meniu juostoje paspaudimu;
- leisti vartotojui pasirinkti aibę klasių, kurios bus analizuojamos;
- leisti pasirinkti konfliktų sprendimo scenarijų;
- leisti pasirinkti analizuojamų klasės elementų aibes;
- atlikti algoritme nurodytus veiksmus;

## *4.6. MAGICDRAW UML ĮSKIEPIO PROJEKTINIAI MODELIAI*

### 4.6.1. ĮSKIEPIO PANAUDOJIMO ATVEJŲ DIAGRAMA

Įskiepio panaudojimas susidarys iš šešių įskiepio panaudojimo atvejų, kurie yra pateikti paveikslėlyje Nr. 13.




13. pav. Įskiepio panaudojimo atvejų diagrama

Vartotojas galės paleisti plėtinį, atsiradusiame lange pasirinkti norimą analizuoti klasių aibę bei vykdyti klasių modelio analizę ir papildymą, arba uždaryti įskiepio vykdymo langą. Panaudos atvejų aprašas yra pateiktas lentelėje nr. 4.

Lentelė Nr. 4. Panaudos atvejų aprašas

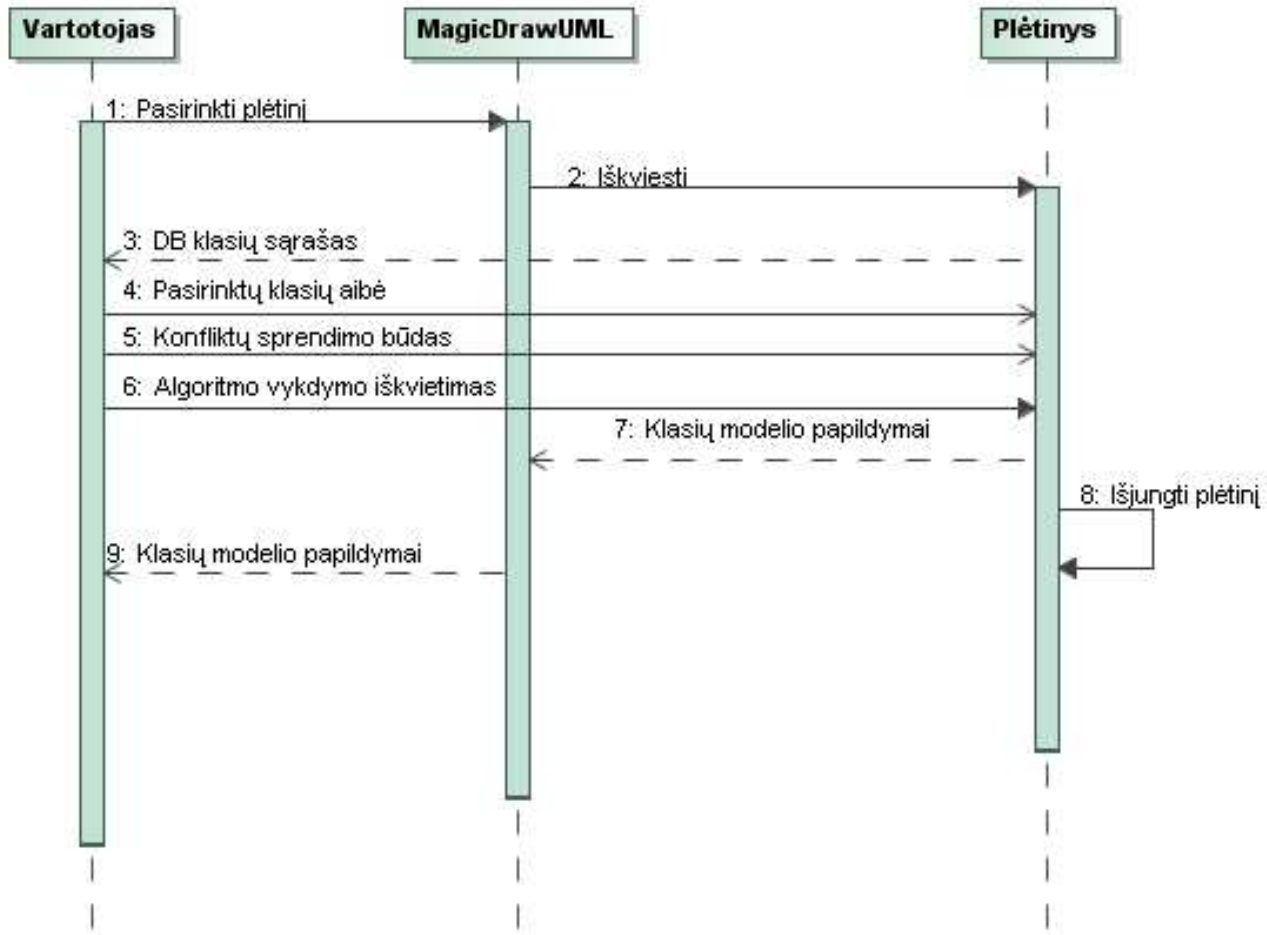
| **Panaudos atvejis** | **Aprašymas** |
|---|---|
| Paleisti įskiepį | Vartotojas meniu juostoje pasirenka ir paspaudžia įskiepio paleidimo mygtuką. Yra atidaromas įskiepio vykdymo langas. |
| Vykdyti klasių modelio papildymą VT pagrindu | Paspaudus šį įskiepio mygtuką yra paleidžiamas įskiepio algoritmas. Algoritmas yra vykdomas pasirinktai klasių aibei, naudojant pasirinktą konfliktų sprendimų scenarijų. |
| Išsirinkti analizuojamų klasių aibę | Vartotojas pasirenka VT duomenų bazėje aprašytų klasių aibę. |
| Nustatyti konfliktų sprendimų scenarijų | Vartotojas pasirenka vieną iš dviejų konfliktų sprendimų scenarijų: 1)VT yra aukštesnio prioriteto; 2)KM yra aukštesnio prioriteto; |
| Uždaryti įskiepį | Uždaromas atidarytas įskiepio vykdymo langas. |

Ateityje bus galima praplėsti įskiepio funkcionalumą, leidžiant vartotojui pasirinkti, kokius veiksmus leisti atlikti algoritmui, o kokius drausti. Taip pat turėtų būti galimybė papildyti įskiepio funkcionalumą, realizuojant abipusius mainus tarp veiklos taisyklių duomenų bazės ir MagicDraw klasių modelio.

Turėtų būti galimybė nurodyti, kokius klasių elementus analizuoti ir kaip spręsti iškilusius konfliktus.

### 4.6.2. ĮSKIEPIO PANAUDOJIMO SEKŲ DIAGRAMA

Įskiepio sekų diagrama yra pateikta paveikslėlyje Nr. 14.

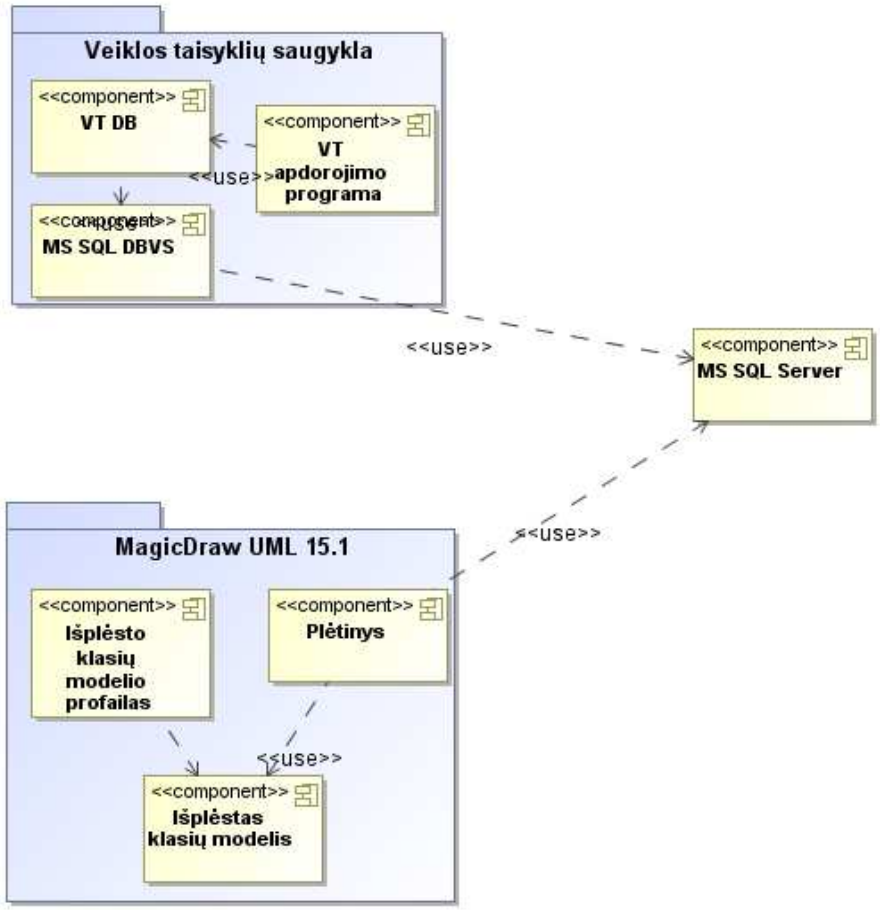



14. pav. Įskiepio panaudojimo sekų diagrama

Vartotojas galės paleisti plėtinį iš MagicDraw UML meniu pasirinkęs įskiepio pavadinimą. Paleidus plėtinį bus sukuriamas arba praplečiamas MagicDraw UML esantis klasių modelis pagal VTS‘os duomenų bazėje aprašytus objektus ir pasirinktus algoritmo funkcionalumo nustatymus.

### 4.6.3. ĮSKIEPIO KOMPONENTŲ DIAGRAMA

Kuriamas įskiepis integruos veiklos taisyklių saugyklą su išplėstu klasės modeliu. Kuriamo sprendimo pagrindiniai komponentai pateikti paveikslėlyje Nr.15.



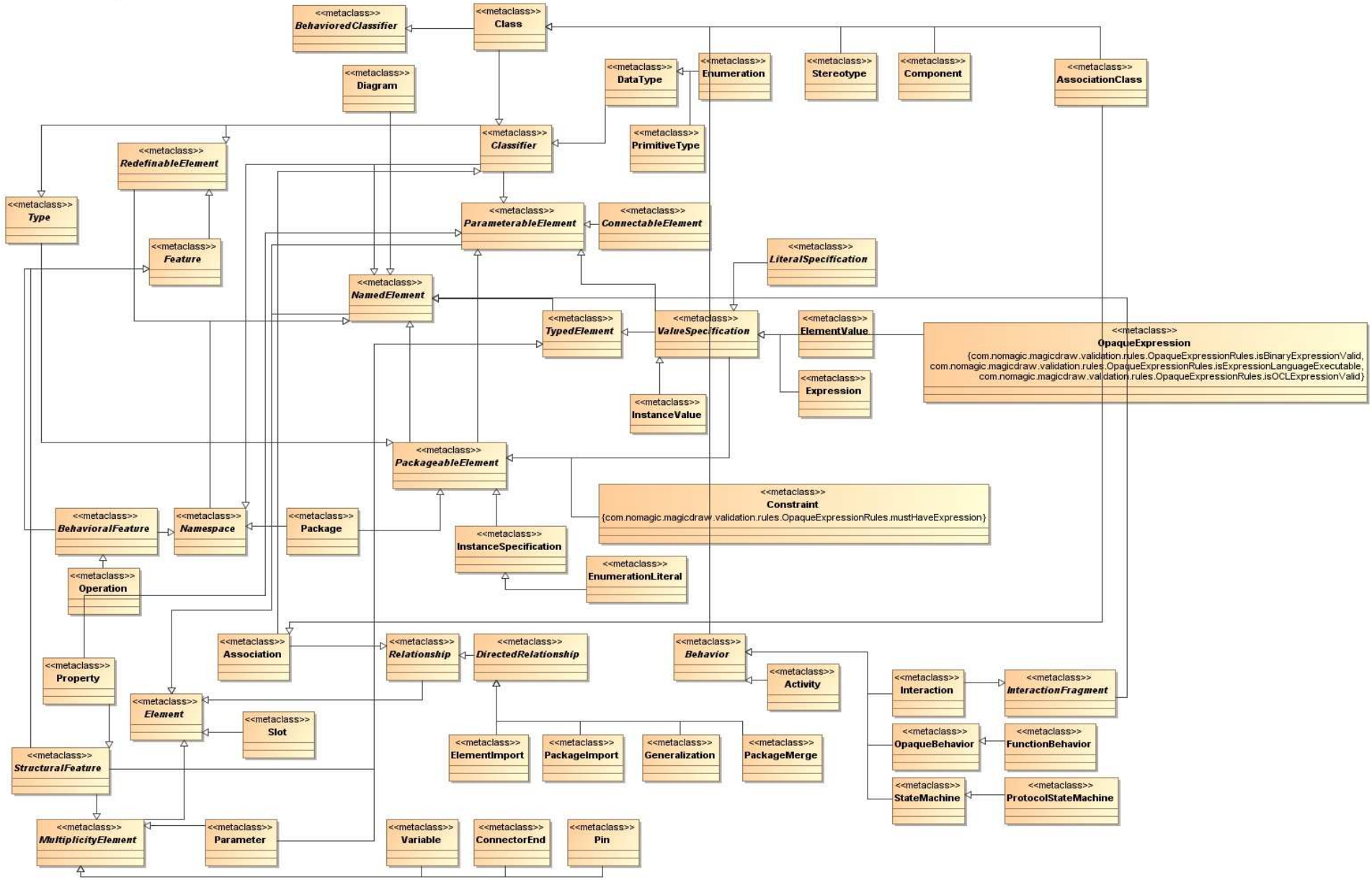

15. pav. Įskiepio vieta sistemoje

Kaip matoma iš komponentų diagramos, įskiepis sujungs dvi nepriklausomas sistemas. Taigi įskiepis atlieka veiklos taisyklių integravimo į projektinį MagicDraw UML klasių modelį. Ši integracija yra labai svarbi informacinių sistemų projektavimo remiantis žiniomis dalis.

## *4.7.MAGICDRAW UML NAUDOJAMAS KLASĖS METAMODELIS*

MagicDraw UML glaudžiai bendradarbiauja su OMG. Buvo išsiaiškinta, kad MagicDraw UML palaiko OMG pateiktą UML2 specifikaciją. MagicDraw UML projektiniai metamodeliai yra realizuoti pagal OMG specifikacijas ir palaiko standartus.

MagicDraw UML klasių metamodelio fragmentas, pateiktas paveikslėlyje Nr.16. Šis klasių metamodelis buvo sukurtas naudojant atvirkštinę inžineriją. Buvo piešiamas klasių metamodelio fragmentas pasinaudojant MagicDraw UML funkcijomis, kurios pateiktame UML2 metamodelyje parodo susijusius elementus.




16. pav. MagicDraw UML klasių metamodelio fragmentas

Atlikus dalinę atvirkštinę inžineriją buvo patvirtinta, jog MagicDraw UML pilnai palaiko OMG pateiktą UML2 klasių metamodelio specifikaciją [17].

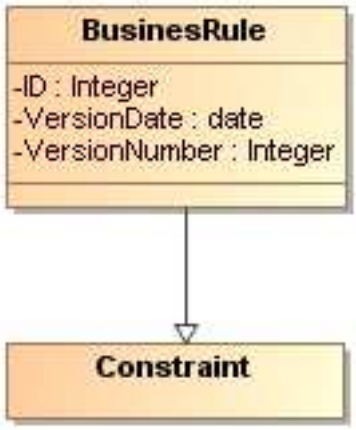
## *4.8.MAGICDRAW UML KLASIŲ METAMODELIO PRAPLĖTIMAS*

Kuriamam VT taikymo sprendimui klasių metamodelis turi būti išplėstas. Šis išplėtimas yra aprašytas skyriuje 10.5. Jis turėtų pritaikyti klasių modelį veiklos modeliavimo dalykinei sričiai ir ISK mokslininkų kuriamo algoritmo pagalba klasių modelis turėtų būti automatiškai generuojamas iš veiklos modelio.

Šiame darbe kuriamam algoritmui klasių metamodelį užtenka praplėsti tik *BusinessRules* atributu, pagal kurį bus atpažįstamos veiklos taisyklės iš duomenų bazės.

Išplėsto MagicDraw UML klasių metamodelio fragmento vaizdas yra pateiktas paveikslėlyje Nr.18.

Praplėtimas bus atliktas remiantis G. Tamulio rekomendacijomis [27], kuriame aprašyta MagicDraw UML praplėtimo metodika. *Constraint* elementas bus praplėstas papildomais elementais. Naujasis *BusinesRule* elementas paveldės visus *Constraint* elementus ir bus praplėstas naujais atributais, kurie yra matomi paveikslėlyje Nr. 17.

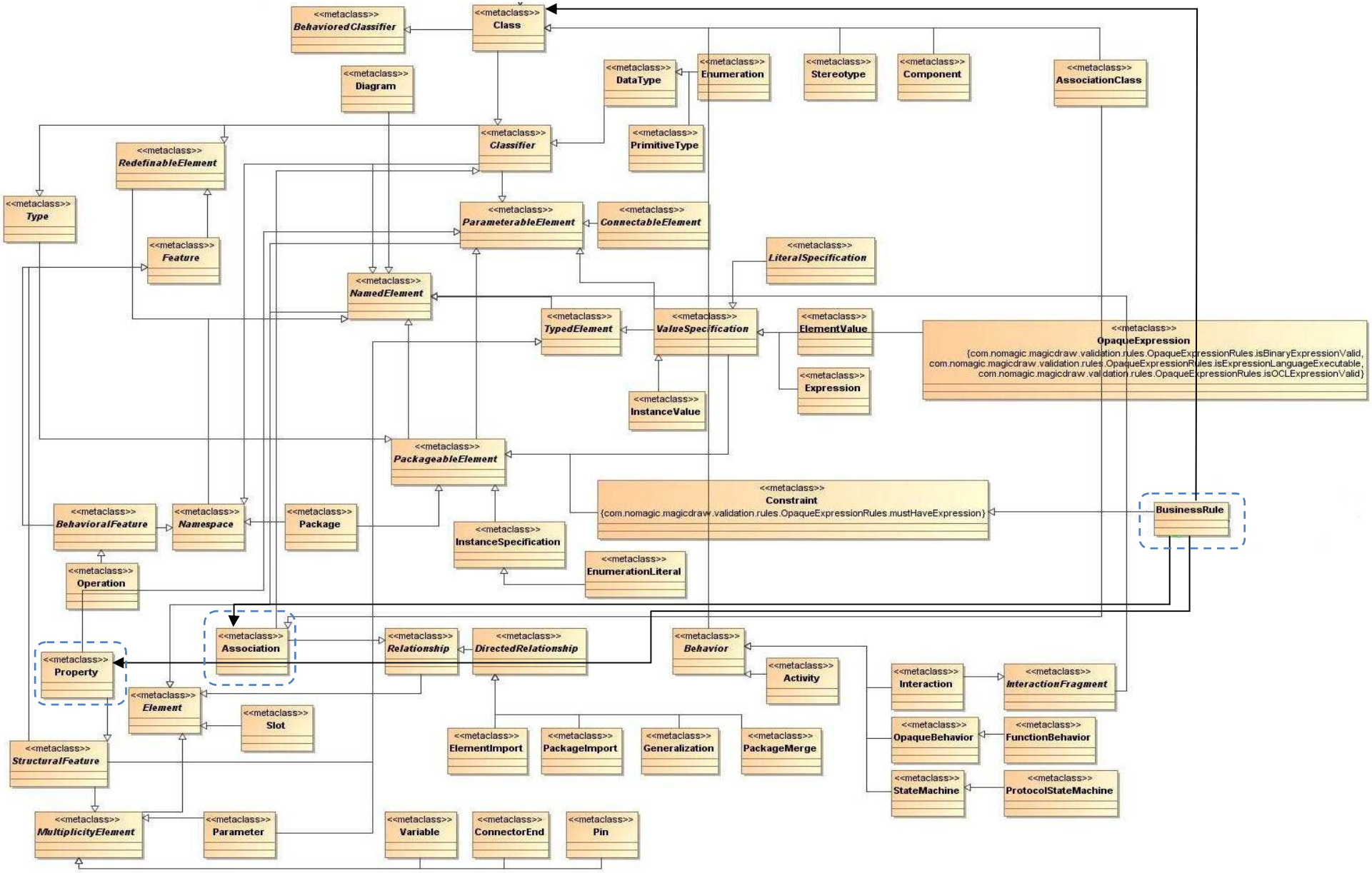



17. pav. *BusinessRule* elemento sandara

Naujas elementas turės tokius laukus:

- *ID* – veiklos taisyklės identifikatorius duomenų bazėje;
- *VersionDate* – naudojamos veiklos taisyklės versijos sukūrimo data;
- *VersionNumber* - naudojamos veiklos taisyklės versijos numeris.

Kitus laukus *BusinessRule* elementas paveldės iš *Constraint* elemento.




18. pav. MagicDraw UML klasių metamodelio praplėtimas

Išanalizavus esamą UML2 metamodelį, buvo išsiaiškinta, jog tokie elementai, kaip Event, Action ir su jais susiję (su jais paveldimi) elementai, kurie yra pateikti ISK mokslininko pasiūlytame išplėstame klasių metamodelyje, nepriklauso klasių metamodeliams. Šie elementai yra iš Activity modelio ir jie su klasių modeliu gali būti siejami tik per *Behavior* žymę, kuri aprašo klasės objekto elgseną. Tačiau elgsenos aprašymui taip pat naudojamos diagramos – medeliai, tokie kaip: Activity, StateMachine ir kt.

Klasių metamodelio praplėtimas daromas, kad veiklos taisyklių pagalba galima būtų papildyti klasių modelį. Šis praplėtimas išplečia klasių modelio panaudojimo lankstumą ir pritaiko klasių modelį automatiškam jo generavimui iš surinktų dalykinės srities konkrečių veiklos taisyklių.

Šiame darbe bus reikalingas tik klasių metamodelio praplėtimas, kuris išplės tik klasės tipo objektų savybes – *BusinessRule* elementu.

## *4.9.MAGICDRAW UML ĮSKIEPIO KŪRIMAS*

Kadangi KTU universitetas bendradarbiauja su įmone No Magic, kuri kuria projektavimo įrankį MagicDraw UML, kuris jau kelis metus iš eilės yra pripažįstamas geriausiu savo srityje.

Kuriamas algoritmas bus realizuotas MagicDraw UML programiniam paketui. Gamintojai šiam paketui paliko galimybę susikurti paketo plėtinį, kurį galima integruoti su visu programinio paketo funkcionalumu. Aprašymas, kaip sukurti plėtinį yra kiekviename kompiuteryje, kuriame yra įdiegtas MagicDraw UML paketas [25].

## *4.10. MAGICDRAW UML METAMODELIŲ PRAPLĖTIMAS*

MagicDraw UML projektavimo pakete yra realizuotas labai lankstus praplėtimo ir pritaikymo mechanizmas, kuris gali labai išplėsti MagicDraw UML panaudojimo galimybes. Nuo egzistuojančių modelių praplėtimo iki naujų modelių kūrimo naudojantis esamų modelių pagrindu.

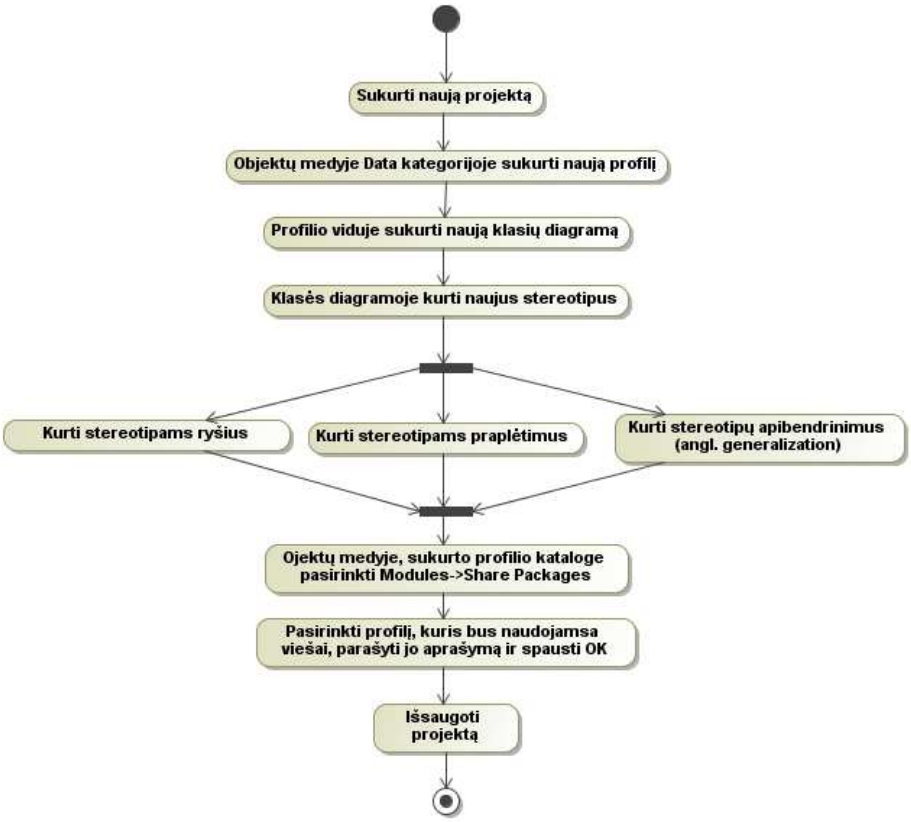
MagicDraw UML praplėtimo mechanizmas susideda iš:

- Profilių mechanizmo, kuris susideda iš:
    - Stereotipų;
    - Elementų;
    - Ribojimų.
- Custom diagramų kūrimo mechanizmo.

Taigi MagicDraw UML modelius galima modifikuoti ir pritaikyti reikiamiems tikslams.

### 4.10.1. MAGICDRAW UML PROFILIO KŪRIMAS

Profilis gali būti sukurtas bet kuriame darbiniame projekte, tačiau dažniausiai vienas profilis yra naudojamas daugelyje projektų. Profilio pakartotinam panaudojimui yra būtina jį sukurti su viešais duomenimis, vadinamais moduliais. Profilio kūrimo algoritmas pateiktas paveikslėlyje Nr. 19.




19. pav. MagicDraw UML profilio kūrimas

Kuriamame profilyje dažniausiai yra kuriami konkrečių modelio elementų praplėtimai – stereotipai.

Kuriami nauji, stereotipizuoti modelio elementai gali turėti:

- papildomus atributus;
- gali būti paslėpti esantys standartiniai atributai;
- specifinį elemento grafinį žymėjimą;
- specifinių elementų ribojimus (elementų jungimo taisykles).

Sukurti profiliai yra saugojami ir gali būti užkraunami tik kai reikalingas specifinis modelis. Šis profilis gali būti panaudojamas daug kartų.

### 4.10.2. MAGICDRAW UML SPECIALIZUOTOS DIAGRAMOS KŪRIMAS

Kiekvienam naujam profiliui, dažniausiai yra kuriama specializuota diagrama. MagicDraw UML pakete yra realizuotas diagramų (modelių) kūrimo įrankis, kurio pagalba yra sukuriamos atitinkamos įrankių juostos, kuriose yra įdedami specifiniai, profilyje aprašyti objektai, modelio elementai.

Naudojantis sukurtomis įrankių juostomis realizuojamas vizualus specifinio profilio pateikimas, kuris pagerina modelio supratimą ir naudojamumą. Patogus ir aiškus modelio skaitymas ir sudarymas lemia vartotojų palankumą sukurtam modeliui ir daro jų darbą efektyvesnį.

## *4.11. PROJEKTINĖS DALIES IŠVADOS*

1. Šioje dalyje buvo suprojektuotas papildytas algoritmas kuris turėtų būti funkcionalesnis ir teikti didesnę naudą veiklos taisyklių taikymui.
2. Buvo išanalizuota pasiūlyta veiklos taisyklių saugyklos duomenų bazė ir atrinkta tik šiam darbui reikalinga saugyklos dalis. Taip pat buvo suprojektuotos *Stored procedūros*, kuriomis duomenų bazių valdymo sistemoje bus atliekama dalis algoritmo funkcionalumo, susijusio su duomenų bazės analize.
3. Šioje dalyje buvo apsibrėžti funkciniai ir nefunkciniai kuriamo įskiepio reikalavimai. Buvo suprojektuotos įskiepio panaudos atvejų, sekų, komponentų ir diegimo diagramos.
4. Kadangi KTU universitetas bendradarbiauja su įmone No Magic, kuri kuria projektavimo įrankį MagicDraw UML. Buvo išanalizuota šio paketo praplėtimo galimybė. Projektinės dalies metu buvo išanalizuotos MagicDraw UML plėtinio kūrimo ypatybės. Buvo išsiaiškintos šio paketo metamodelių praplėtimo galimybės ir mechanizmas.
5. Atlikus projektinę dalį, didžioji laiko dalis buvo skirta klasių metamodelio analizavimui ir jo praplėtimo galimybės pagal pateiktą teorinį metamodelį. Atlikus šią analizę, buvo nustatyta, jog pilnai realizuoti teorinį klasių metamodelio praplėtimą yra labai sudėtinga, kadangi šiame modelyje yra panaudoti kitų modelių elementai. Tačiau kuriamam algoritmui užtenka tik papildyti klasių modelio metamodelį *BusinessRule* elementu ir juo papildyti *Class*, *Property* ir *Association* klasės modelio elementus.

# 5. REALIZACINĖ IR EKSPERIMENTINĖ DALIS

## *5.1.REALIZACIJOS UŽDAVINIAI*

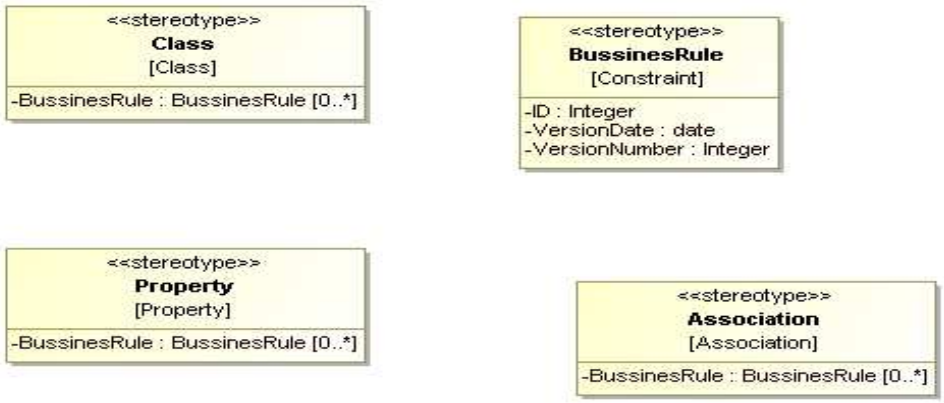
Kadangi darbo laikas yra ribojamas, tai buvo nuspręsta, jog bus realizuoti tik 5 iš pasiūlytų punktų. Buvo nuspręsta atsisakyti rolės tipo veiklos taisyklių analizės.

Algoritmo realizacijai reikia atlikti tokius veiksmus:

- sukurti MagicDraw UML profilį;
- sukurti duomenų bazę ir užpildyti ją reikiamais pradiniais duomenimis;
- sukurti nesudėtingų veiklos taisyklių įvedimo įrankį ir jo pagalba suvesti testavimo duomenis ;
- realizuoti algoritmą;
- ištestuoti algoritmą – atlikti eksperimentą.

## *5.2.MAGICDRAW UML SUKURTAS PROFILIS*

Kadangi algoritmo funkcionalumo realizavimui reikalingas klasių metamodelio praplėtimas, tai pirmiausiai reikia atlikti šį praplėtimą. Pasinaudojus G. Tamulio [27] sukurta metodika, buvo sukurtas klasių metamodelio praplėtimas, kuris klasės tipo objektą praplėtė *BusinessRule* elementu, paveldinčiu visas *Constraint* elemento savybes. Šiam praplėtimui MagicDraw UML yra naudojamas stereotipų ir profilių mechanizmas. Taigi buvo sukurtas profilis, kuriame realizuotas šis praplėtimas (pav. 20).




20. pav. Profilyje praplėsti elementai

Kaip matoma iš paveikslėlio Nr. 20, buvo sukurtas naujas elementas *BussinesRule,* kuris sukurtas *Constraint* elemento pagrindu. Jis yra praplėstas trimis laukais. Taip pat papildyti visi klasės elementai, kurie yra praplečiami naujuoju elementu.

Norint pasileisti sukurtą algoritmą, pradžioje atidarytame MagicDraw UML projekte turi būti įkeltas specializuotas profilis.

## *5.3. MAGICDRAW ĮSKIEPIO KŪRIMO ALGORITMAS*

Plėtinį kuriant, reikia atlikti šiuos veiksmus :

***Žingsnis. 1.*** Sukurti įskiepio direktoriją „plugins“ direktorijoje;

***Žingsnis. 2.*** Parašyti įskiepio kodą. Įskiepis turi turėti bent klasę, kuri paveldi com.nomagic.magicdraw.pluginsPlugin klasę

***Žingsnis. 3.*** Sukompiliuoti ir supakuoti plėtinį į jar failą;

Kodo kompiliavimui turi būti įtrauktos klasės į sisteminius nustatymus *java classpath*

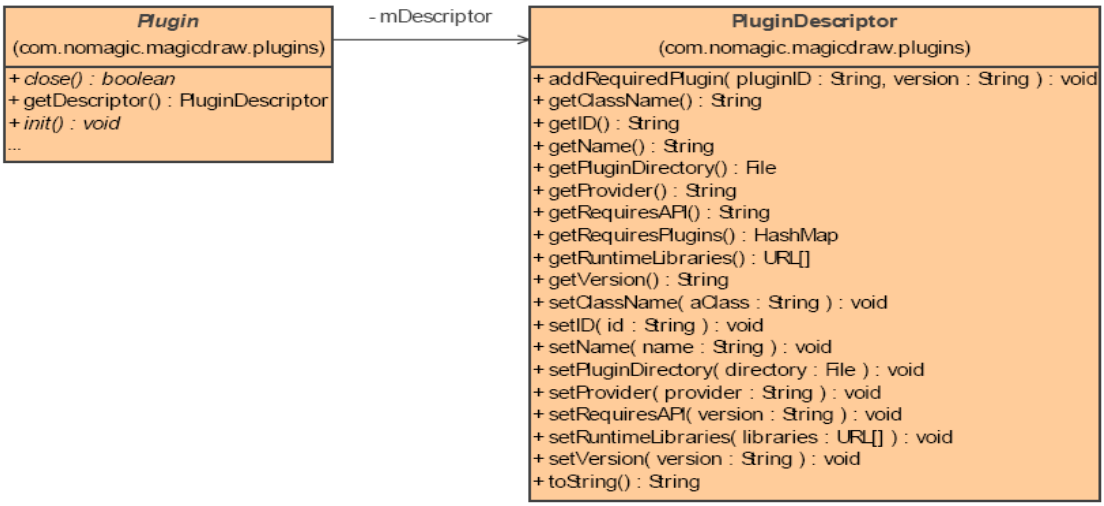
Visos reikalingos MagicDraw klasės yra supakuotos šiose bibliotekose:

- <MagicDraw installation directory>/lib/md_api.jar
- <MagicDraw installation directory>/lib/md_common_api.jar plėtinių klasės (iš „com.nomagic.magicdraw.*" paketo) yra supakuoti konkrečiame įskiepio jar faile
- <MagicDraw installation directory>/lib/jide_action.jar
- <MagicDraw installation directory>/lib/jide-common.jar
- <MagicDraw installation directory>/lib/jide-components.jar
- <MagicDraw installation directory>/lib/jide-dock.jar
- <MagicDraw installation directory>/lib/jide-grids.jar
- <MagicDraw installation directory>/lib/uml2.jar
- <MagicDraw installation directory>/lib/javax_jmi-1_0-fr.jar
- <MagicDraw installation directory>/lib/cmof14.jar
- <MagicDraw installation directory>/lib/y.jar

Sukompiliuotas kodas turi būti supakuotas į jar failą. Šio paketo formavimui reikia naudoti jar komandą „plugins“ direktorijoje (pvz. jar -cf myplugin\myplugin.jar myplugin *.class)

***Žingsnis. 4.*** Parašyti įskiepio aprašymą; Įskiepio aprašymas turi būti faile plugin.xml. Šis failas turi būti patalpintas kuriamo įskiepio direktorijoje.

Rašant įskiepio kodą būtina panaudoti bent vieną apibrėžtą klasę iš C*om.nomagic.magicdraw.-plugins.Plugin* klasių (pav. Nr. 21).




21. pav. Įskiepio klasės [25]

*Plugin* yra bazinė abstrakti klasė visiems MagicDraw įskiepiams (pav. 21). Kuriamas įskiepis turi paveldėti šią klasę. Kiekvienas įskiepis turi savo aprašymą, kurį nuskaito įskiepių valdytojas (angl. Plug-in Manager). Įskiepis turi tris specialius metodus:

- *public abstract void init();*

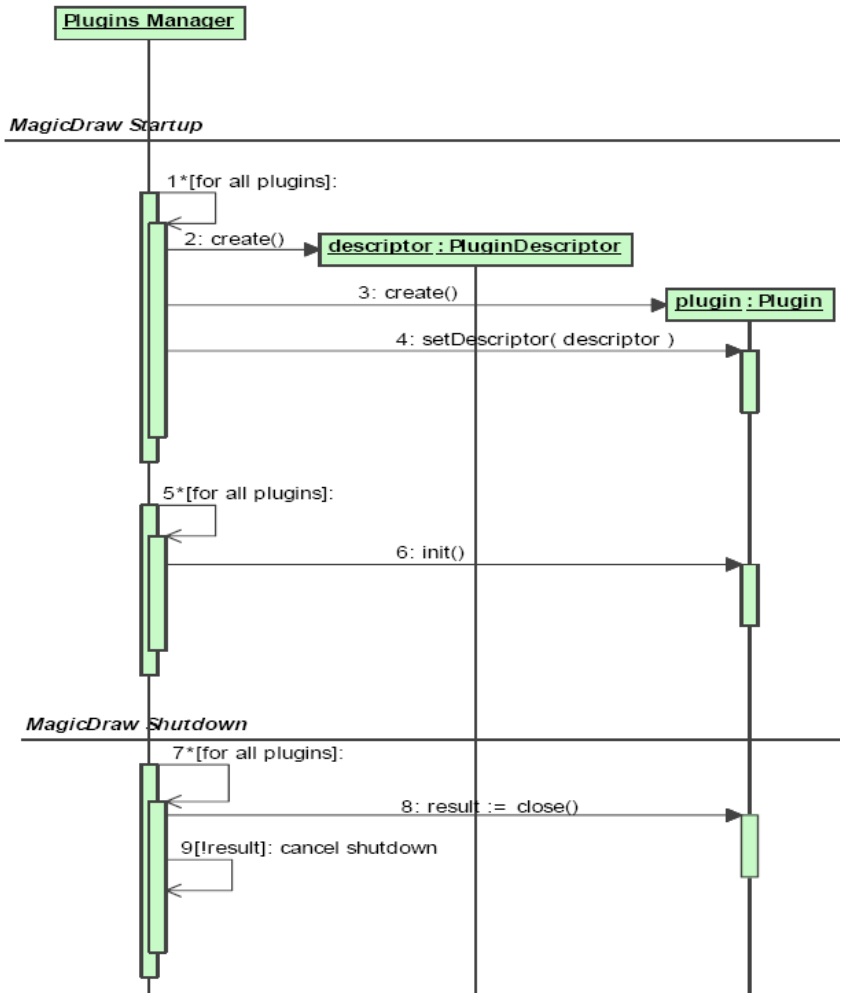
  Metodas yra vykdomas kraunantis MagicDraw programai (pav. 14). Įskiepis turi perimti šį metodą ir realizuoti savo funkcionalumą.

- *public abstract boolean close();*

  Metodas yra vykdomas išsijungiant MagicDraw programai (pav. 10). Įskiepis turi perimti šį metodą ir gražinti teigiamą reikšmę (*true*), jei įskiepis yra pasiruošęs išjungimui arba neigiamą reikšmę (*false*) kitu atveju ir programos išjungimas bus atšauktas.

- *public abstract boolean isSupported();*

  Metodas yra vykdomas prieš įskiepio *init()* metodą. Įskiepis nebus įtraukiamas, metodui gražinant neigiamą reikšmę (*false*).




22. pav. Įskiepio veikimo seka [25]

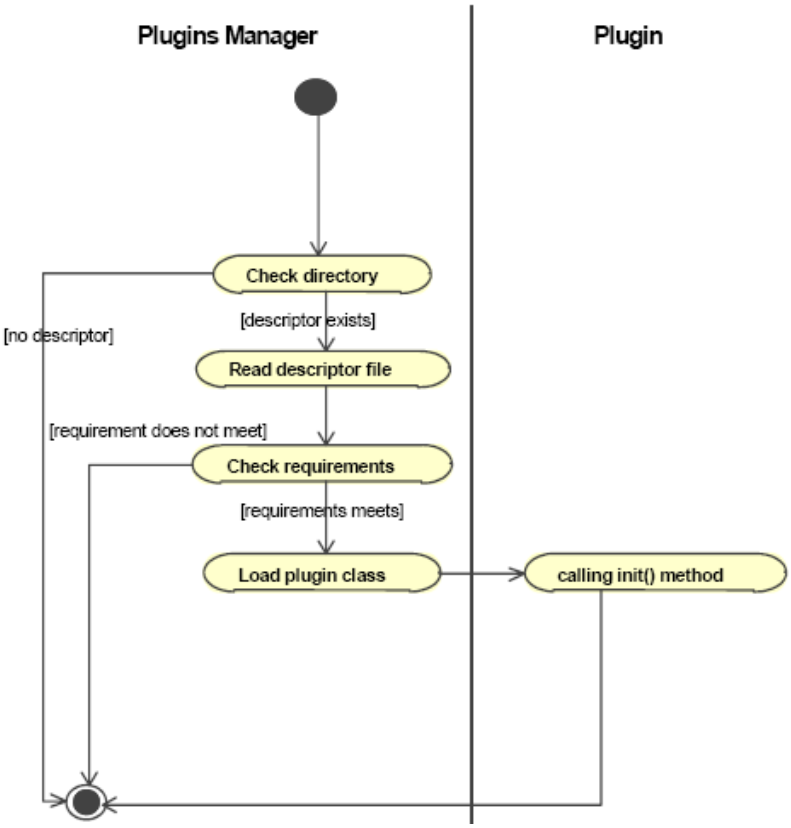
Šiame paveikslėlyje (pav. 22) yra pavaizduoti veiksmai, kurie yra atliekami leidžiant/vykdant bet kurį plėtinį.

### 5.3.1. MAGICDRAW ĮSKIEPIO VEIKIMAS

MagicDraw kiekvieno paleidimo metu peržiūri „plug-ins“ katalogą ir jame ieško pakatalogių:

- Jei pakatalogyje yra įskiepį aprašanti rinkmena, tai įskiepių valdymo agentas skaito šią rinkmeną.
- Jei reikalavimai, apibrėžti aprašymo rinkmenoje, yra įvykdyti, įskiepių agentas užkrauna apibrėžtą klasę (apibrėžta papildinio klasė turi būti *com.nomagic.magicdraw. plugins. Plugin* poklasis). Tada kreipiamasi į užkrautos klasės metodą init (). Kuris metodas kviečiamas tik jei isSupported () sugrąžina teigiamą (*true*) reikšmę. Šis metodas gali pridėti grafinės vartotojo sąsajos komponentus, naudodamas veiksmų architektūrą ar padaryti kitą veiksmą ir sugrįžti iš metodo.

Paveikslėlyje Nr. 23 yra matomas MagicDraw UML programinio paketo įskiepių valdytojo veikimo schema.

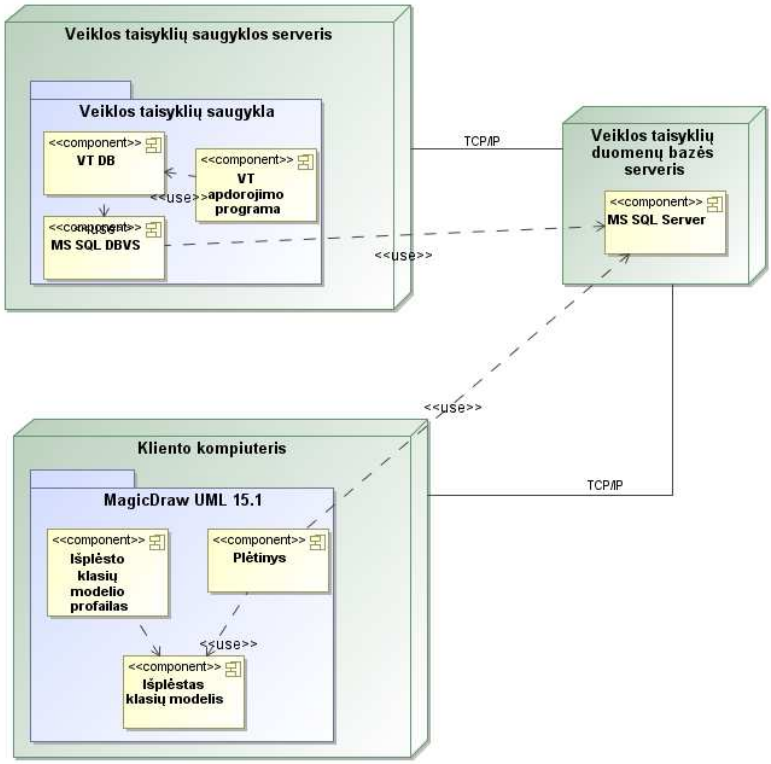



23. pav. MagicDraw įskiepio panaudojimas [25]

MagicDraw UML paketas yra pritaikytas plėtiniams, kurie leidžia šiam paketui tobulėti ir populiarėti dėl lankstumo ir galimybės individualiai prasiplėsti paketo funkcionalumą. Nors ir yra pateikta minimali dokumentacija, apie įskiepių mechanizmus, tačiau jų kūrimui reikalingos specifinės žinios apie MagicDraw UML paketo savybes.

## *5.4.ĮSKIEPIO DIEGIMO DIAGRAMA*

Norint parodyti, kad kuriamas sprendimas yra lankstus ir mobilus yra pateikiama diegimo diagrama (pav. 24).

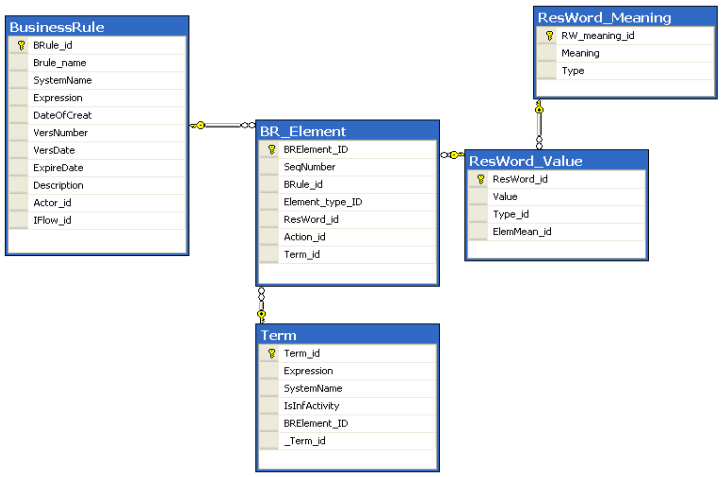



24. pav. Sistemos diegimo diagrama

Diegimo diagramoje matome, jog trys pagrindinės sistemos dalys yra išskaidytos į tris fizinius serverius, kurių naudojamos technologijos ir funkcijos skiriasi. Kuriamas įskiepis integruos dvi skirtingomis technologijomis parašytas sistemas ir jas apjungs į vieningą sistemą. Visos sistemos sudedamosios dalys gali funkcionuoti viena nuo kitos nepriklausomai, tačiau, nors, vieną sistemos dalį išjungus, sistema praras kuriamą funkcionalumą.

## *5.5.REALIZUOTA VT SAUGYKLOS DUOMENŲ BAZĖ*

Algoritmo realizacijai buvo sukurta MS SQL Express duomenų bazė, kuria bus emuliuojama veiklos taisyklių saugyklos duomenų bazės dalis, kuri bus analizuojama sukurto algoritmo realizavimo metu. Sukurtos fizinės duomenų bazės schema yra pateiktas paveikslėlyje Nr. 25




25. pav. Fizinės duomenų bazės schema

Taip pat šioje duomenų bazėje buvo sukurtos duomenų bazėje saugomos procedūros, kurių funkcionalumas yra aprašytas skyriuje 12.3.

Duomenų bazės valdymo sistemos pagalba buvo įvesti reikiami pradiniai duomenys. Kadangi veiklos taisyklės yra užrašomos natūralios kalbos pagalba, tai reikia susikurti atitinkamus *ResWord_Value* lentelės šablonus, kurie turėtų atitinkamą reikšmę *ResWord_Meaning* lentelėje. Šis pradinis užrašymas yra pateiktas lentelėje Nr .5.

Lentelė Nr. 5. ResWord išraiška ir reikšmės

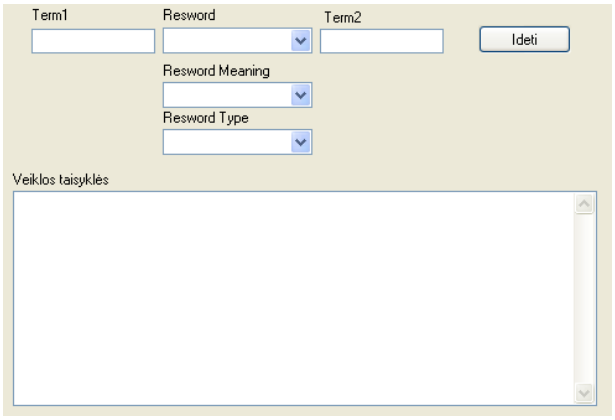
| **Lentelė** / **Tipas** | **ResWord_Meaning** | | **ResWord_Value. Value** |
|---|---|---|---|
| | **Meaning (reikšmė)** | **Type (tipas)** | **Išraiška lietuvių kalba** |
| *Atributas* | Feature | Null | Turi savybę,<br>pasižymi |
| *Rolė* | Role | Null | Vaidina vaidmenį,<br>Vaidmuo fakte |
| *Ryšys* | Relationship | Generalization | Yra |
| | | Association | Naudoja |
| | | Aggregation | Neegzistuoja be |
| | | Composition | Turi turėti,<br>gali turėti,<br>susideda iš, |

Pagal šiuos rezervuotų žodžių šablonus bus identifikuojamos jų reikšmės ir tipai. Naudojant šiuos natūralios kalbos šablonus galime kuriamą algoritmą pritaikyti bet kokiai kalbai. Tam tereikia pakeisti veiklos taisyklių surinkimo programoje atitinkamos kalbos šablonus, kurie duomenų bazės lentelės ResWord_Value užpildys Value reikšmes atitinkamai kalbai.

## *5.6.SUKURTAS ĮRANKIS TESTINIAMS DUOMENIMS SUVESTI*

Į duomenų bazę įvedant vieną veiklos taisyklę yra pildomos 5 lentelės. Pildant šias lenteles rankiniu būdu, yra labai didelė galimybė klaidingai įvesti duomenis ir dėl šios priežasties gali nekorektiškai veikti sukurtas algoritmas. Kadangi bus testuojamas algoritmas su keliomis dešimtimis taisyklių, tai yra tikslinga susikurti nesudėtingą įrankį, kuris visuomet teisingai pildytų duomenų bazės lenteles.

Veiklos taisyklėms suvedinėti buvo sukurta programa, kurios vartotojo sąsaja pateikta paveikslėlyje Nr.26. Šis įrankis buvo sukurtas .Net technologijų pagalba, tad jo naudojimuisi reikalingos .Net bibliotekos.




26. pav. Veiklos taisyklių įvedimo įrankis

Naudojantis šiuo įrankiu buvo suvesti testiniai duomenys, kurie pateikti skyriuje 6.8.3. Šiame įrankyje tereikia užpildyti fakto tipo laukus. *Term1* – tai pirmo termo, dalyvaujančio taisyklėje, reikšmė, *ResWord* yra pasirenkami, iškrentančiame sąraše yra užpildytos galimos reikšmės, kurios yra jau užpildytos duomenų bazėje (lentelė nr.4. ->Išraiška lietuvių kalba ). Pasirinkus atitinkamą *ResWord* reikšmę, automatiškai yra užpildomi *ResWordMeaning* ir *ResWordType* laukai. Šie laukai yra skirti tik priminti vartotojui, kokio tipo ResWord pasirinko, ir ką jis reiškia. *Term2* – yra vedamas antrojo termo, dalyvaujančio veiklos taisyklėje, reikšmė. Paspaudus mygtuką *Įdėti,* taisyklė yra įdedama į duomenų bazę ir jos išraiška atspausdinama *Veiklos taisyklė* lauke.

## *5.7.ALGORITMO FUNKCIONALUMO REALIZACIJA*

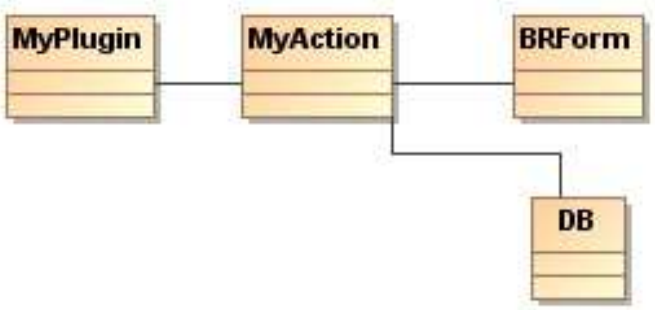
Algoritmo funkcionalumo realizacijai MagicDraw UML pakete buvo naudojamasi jų pateikta įskiepių kūrimo dokumentacija.

Algoritmo realizacijai yra reikalinga:

- MagicDraw UML;
- Eclipse arba kita Java kalbos programavimo aplinka;
- Microsoft SQL Server 2005 JDBC tvarkyklės;
- Microsoft SQL Express Server duomenų bazės valdymo sistema;

- Sukurto pagalbinio įrankio paleidimui reikia.NET Framework bibliotekų.

Realizuotame algoritme yra 3 klasės, kurių tarpusavio ryšiai yra pateikti paveikslėlyje Nr. 27




27. pav. Algoritmo klasės

*MyPlugin* klasė paveldi *com.nomagic.magicdraw.plugins.Plugin* standartinio įskiepio klasę, kurioje aprašyti visi reikiami metodai būtini įskiepiui. *MyPlugin* klasėje yra inicijuojamas įskiepis ir yra sukuriamas *MyAction* klasės tipo objektas.

*MyAction* klasė paveldi *com.nomagic.magicdraw.actions.MDAction* standartinę veiksmo klasę, kurioje yra aprašyti galimi įskiepio paleidimo būdai. Visas algoritmo funkcionalumas yra aprašytas šioje klasėje. Kai užfiksuojamas mygtuko nuspaudimo įvykis yra iškviečiama *BRForm* klasės tipo objektas.

*DBForm* klasė yra programos lango atvaizdavimo klasė, kurioje yra pateikiamos pradinės algoritmo paleidimo sąlygos. Kai yra paspaudžiamas mygtukas, kuris inicijuoja algoritmo vykdymą, iškviečiamas *MyAction* klasės metodas, kuriame aprašytas pagrindinis funkcionalumas. Kai metodas būna įvykdytas *BRForm* klasė yra panaikinama.

*DB* klasėje yra aprašytas prisijungimo prie duomenų bazės algoritmas ir prisijungimas. Į šia klasę kreipiasi *MyAction* klasė.

## *5.8.EKSPERIMENTO EIGA*

### 5.8.1. EKSPERIMENTO UŽDUOTIS

Duotą dalykinės srities ekspertų dalies sistemos objektų aprašymą natūralia kalba užrašyti struktūrizuotomis veiklos taisyklėmis. Remiantis natūralios kalbos šablonais, identifikuoti VT dalyvaujančius objektus. Remiantis veiklos taisyklių saugyklos aprašymu, užpildyti VTS duomenų bazę. Paleidus MagicDraw UML paketą verifikuoti algoritmą.

### 5.8.2. DALYKINĖS SRITIES APRAŠYMAS NATŪRALIA KALBA IR VEIKLOS TAISYKLĖMIS

Dalykinė sritis yra pasirinkta daugumai gerai pažįstama ir kasdien sutinkama – automobilis. Algoritmo veikimui nustatyti užtenka keletos objektų aprašymų, kad būtų matoma, kaip yra sugeneruojamas arba patikslinamas jau sukurtas klasių modelis.

Realizuoto algoritmo testavimui buvo panaudotas toks dalykinės srities aprašymas natūralia kalba:

Asmens savybės yra:
- Vardas;
- Pavardė;
- Amžius.

Vairuotojas turi savybes:
- Stažas;
- kategorija.

Vairuotojas yra asmuo.
Asmuo naudoja automobilį.
Asmens vaidmuo fakte „Asmuo naudoja automobilį“ - vairuotojas.
Transporto priemonė turi savybes:
- Ratų skaičių;
- Tipas.

Automobilis yra transporto priemonė.
Automobilio savybės yra:
- spalva;
- durų skaičius;
- markė;
- kėbulo tipas.

Automobilis turi turėti variklį.
Automobilio variklio savybės yra:
- kuro tipas;
- cilindrų skaičius;
- V forma;
- Gamintojas.

Automobilis turi turėti greičių dėžę.
Greičių dėžė turi savybes:
- Tipas;
- Pavarų skaičius;

Automobilis gali turėti audio sistemą.
Audio sistema turi savybes:
- Grotuvo tipas
- Garsiakalbių skaičius
- Laisvų rankų įranga

### 5.8.3. DALYKINĖS SRITIES APRAŠYMAS VEIKLOS TAISYKLĖMIS

Kadangi algoritme analizuojamos tiktai fakto ir rolės tipo veiklos taisyklės, tai veiklos taisyklių išraiška natūralia kalba ir jų struktūra pateikta lentelėje Nr. 6.

Lentelė Nr. 6. Veiklos taisyklių struktūrizavimas

| **Tipas** | **VT išraiška natūralia kalba** | **Struktūrizuotos VT išraiška** |
|---|---|---|
| *Faktas* | Asmuo turi savybę Vardas. | Asmuo *turi savybę* Vardas. |
| *Faktas* | Asmuo turi savybę Pavardė. | Asmuo *turi savybę* Pavardė. |
| *Faktas* | Asmuo turi savybę Amžius. | Asmuo *turi savybę* Amžius. |
| *Faktas* | Vairuotojas turi savybę stažas. | Vairuotojas *turi savybę* stažas. |
| *Faktas* | Vairuotojas turi savybę kategorija. | Vairuotojas *turi savybę* kategorija. |
| *Faktas* | Vairuotojas yra Asmuo. | Vairuotojas *yra* Asmuo. |
| *Faktas* | Asmuo naudoja automobilį. | Asmuo *naudoja* automobilį. |
| *Faktas* | Transporto priemonė turi savybę ratų skaičius. | Transporto priemonė *turi savybę* ratų skaičius. |
| *Faktas* | Transporto priemonė turi savybę tipas. | Transporto priemonė *turi savybę* tipas. |
| *Faktas* | Automobilis yra transporto priemonė. | Automobilis *yra* transporto priemonė. |
| *Faktas* | Automobilis turi savybę spalva. | Automobilis *turi savybę* spalva. |
| *Faktas* | Automobilis turi savybę durų skaičius. | Automobilis *turi savybę* durų skaičius. |
| *Faktas* | Automobilis turi savybę markė. | Automobilis *turi savybę* markė. |
| *Faktas* | Automobilis turi savybę kėbulo tipas. | Automobilis *turi savybę* kėbulo tipas. |
| *Faktas* | Automobilis turi turėti variklį. | Automobilis *turi turėti* variklį. |
| *Faktas* | Variklis turi savybę kuro tipas. | Variklis *turi savybę* kuro tipas. |
| *Faktas* | Variklis turi savybę cilindrų skaičius. | Variklis *turi savybę* cilindrų skaičius |
| *Faktas* | Variklis turi savybę V forma. | Variklis *turi savybę* V forma. |
| *Faktas* | Variklis turi savybę gamintojas. | Variklis *turi savybę* gamintojas. |
| *Faktas* | Automobilis turi turėti greičių dėžė. | Automobilis *turi turėti* greičių dėžė. |
| *Faktas* | Greičių dėžė turi savybę tipas. | Greičių dėžė *turi savybę* tipas. |
| *Faktas* | Greičių dėžė turi savybę pavarų skaičius. | Greičių dėžė *turi savybę* pavarų skaičius. |
| *Faktas* | Automobilis gali turėti audio sistemą. | Automobilis *gali turėti* audio sistemą. |
| *Faktas* | Audio sistema turi savybę grotuvo tipas. | Audio sistema *turi savybę* grotuvo tipas. |
| *Faktas* | Audio sistema turi savybę garsiakalbių skaičius. | Audio sistema *turi savybę* garsiakalbių skaičius. |
| *Faktas* | Audio sistema turi savybę laisvų rankų įranga. | Audio sistema *turi savybę* laisvų rankų įranga. |
| *Rolė* | Asmens vaidmuo fakte „Asmuo naudoja automobilį“ - vairuotojas. | Asmens *vaidmuo fakte* „Asmuo naudoja automobilį“ - vairuotojas. |

**Legenda:** Terminas, ***Rezervuotas žodis*****,** Factas

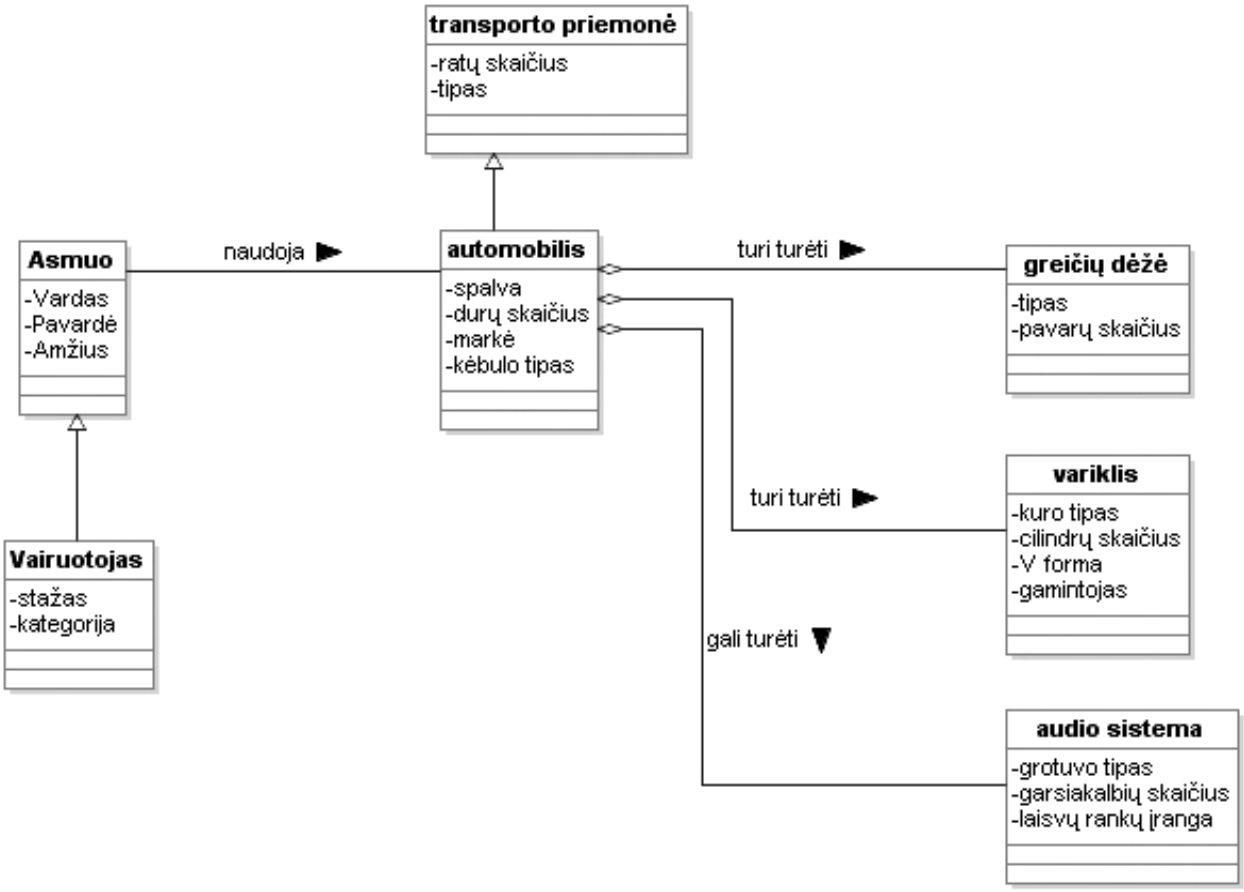
Remiantis natūralios kalbos šablonais žinome, jog Fakto ir Rolės VT sudaromos taip (žiūrėti 9.4. skyrių):

- <Template_Fact>::= <Term> <ResWord> <Term>
- <Template_Role>::=<Term><ResWord><Fact>”-“<Term>.

Kadangi buvo nuspręsta, jog rolės tipo taisyklių analizė nebus realizuota, tai eksperimente pas kutinė veiklos taisyklė nedalyvaus.

### Laukiamas klasių modelis

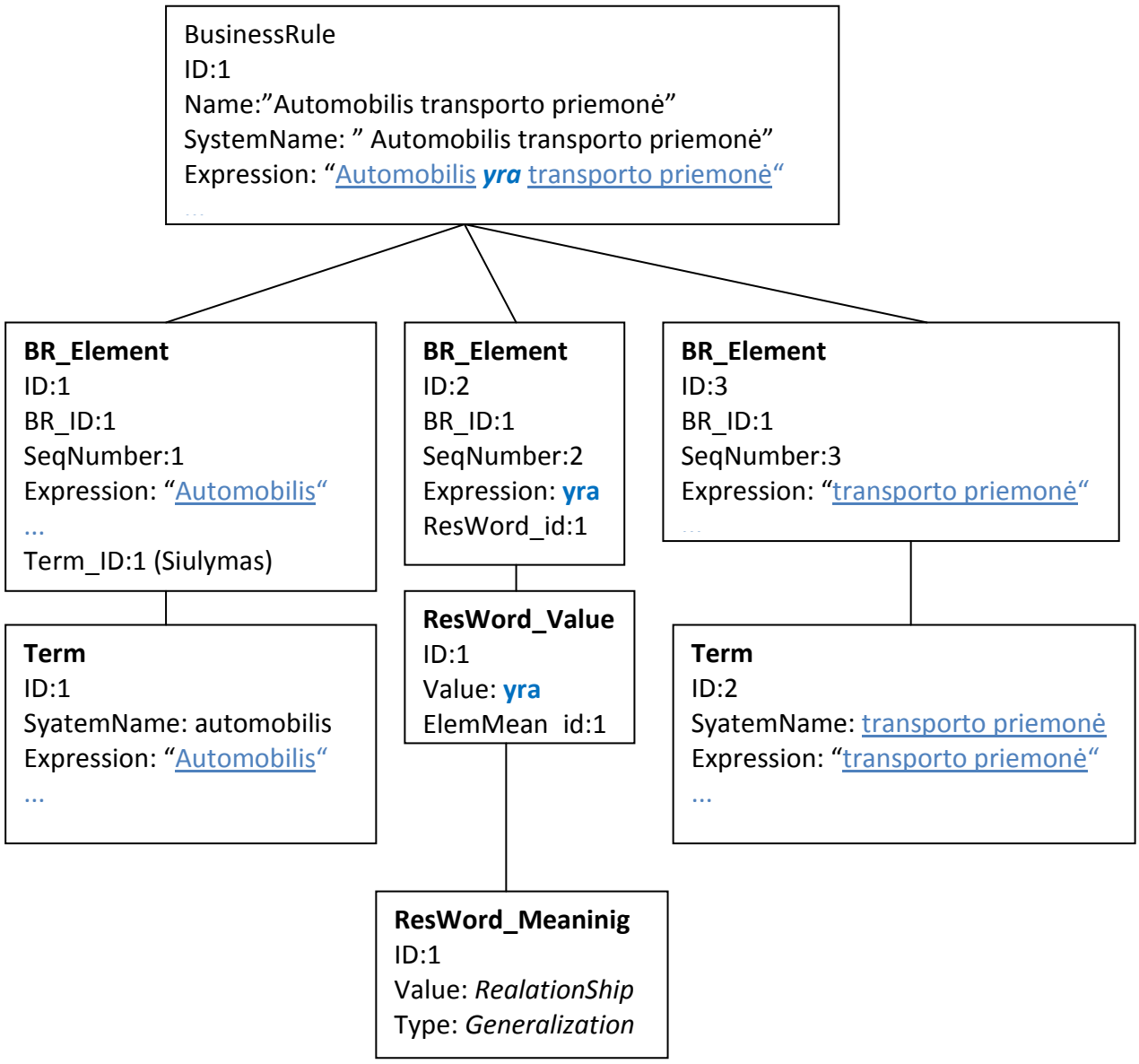
Laukiamas klasių modelis pagal aprašymą yra pateiktas paveikslėlyje Nr. 28.




28. pav. Laukiamas klasių modelio vaizdas.

## Duomenų bazės užpildymo pavyzdžiai

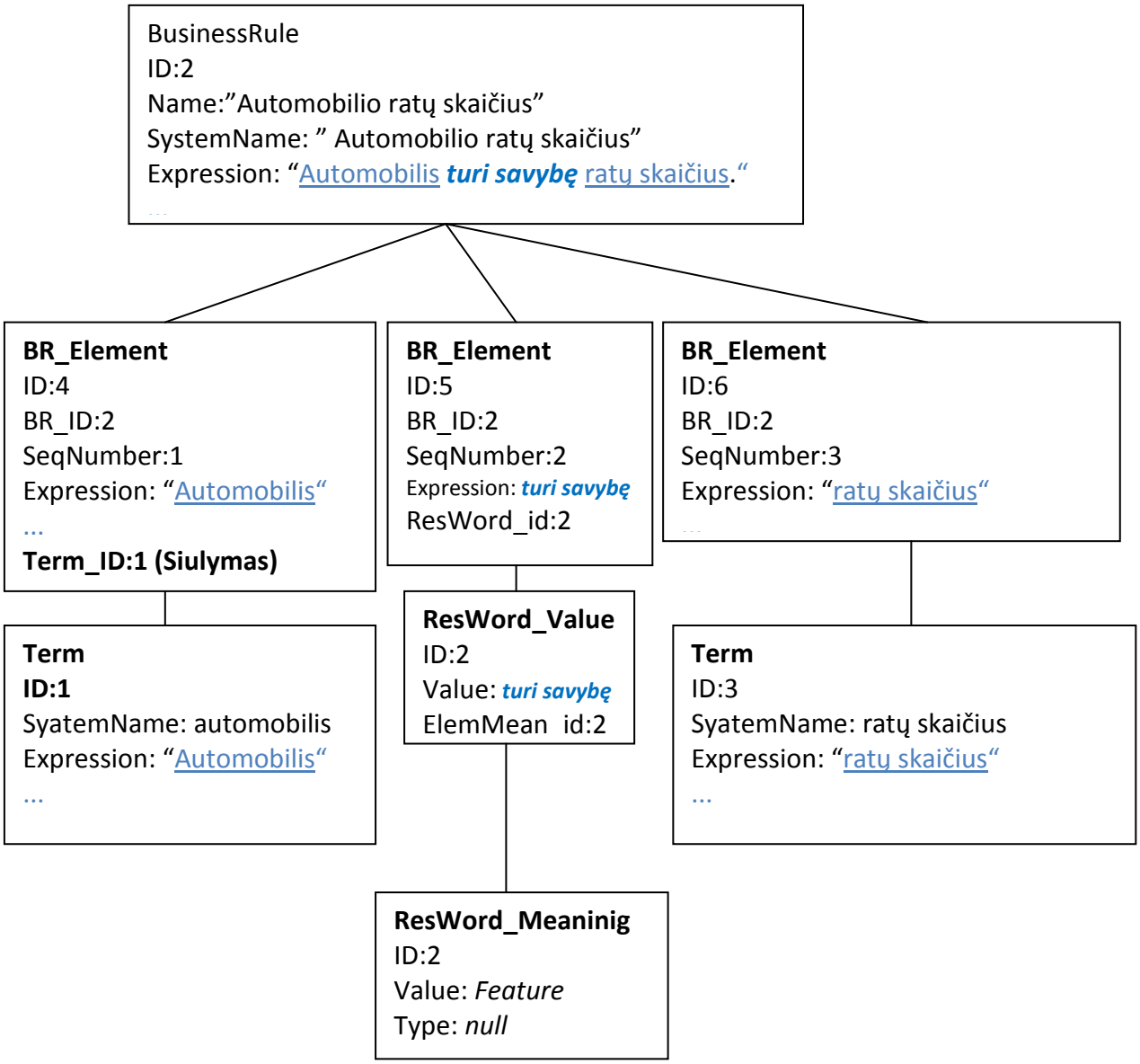
Pateikiami keli pavyzdžiai, kaip yra užpildomos duomenų bazės atitinkamos lentelės duomenimis.




29. pav. VT „Automobilis yra transporto priemonė" DB užpildymo pavyzdys.

Paveikslėlyje Nr 29. Yra pateikiamas pavyzdys, kaip užpildomos VTS duomenų bazės lentelės, kai veiklos taisykle yra aprašomas klasės atributas.

Paveikslėlyje Nr. 30. Yra pateiktas pateikiamas pavyzdys, kaip užpildomos VTS duomenų bazės lentelės, kai veiklos taisykle yra aprašomas ryšys tarp klasių.

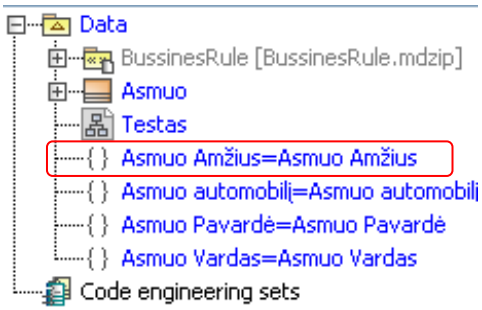



30. pav. VT "Automobilis turi savybę ratų skaičius." DB užpildymo pavyzdys

## *5.9.IŠPLĖSTŲ KLASĖS MODELIO ELEMENTŲ REALIZACIJŲ ILIUSTRACIJOS*

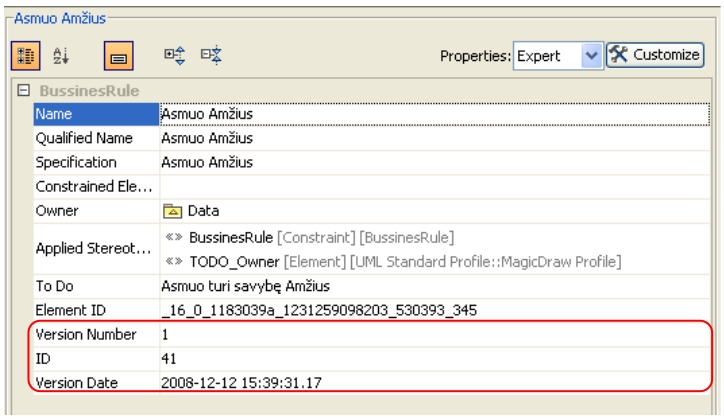
### 5.9.1. SUKURTO BUSSINESRULE ELELEMTO ILIUSTRACIJA

Visų projektinių modelių pagrindinis modelis yra *Data,* jam priklauso visos klasės, diagramos, asociacijos tipo ryšiai ir taip pat BusinessRule. BussinesRule elemento žymėjimas modelyje yra pateiktas paveikslėlyje Nr. 31.




31. pav. BussinesRule elemento vieta modelyje

Šiame paveikslėlyje (pav. 31) *BussinesRule* elementas yra apibrauktas raudonu stačiakampiu. Šis elementas turi visus *Constraint* elemento savybes ir yra praplėstas keliais specifiniais atributais – ID, Version Number, Version Date (apibraukta raudonu stačiakampiu pav. 32.).

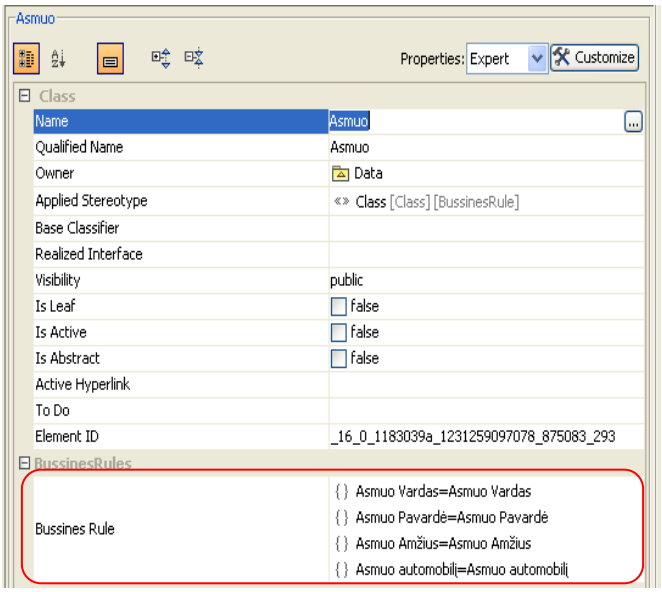



32. pav. BussinesRule elemento savybių iliustracija

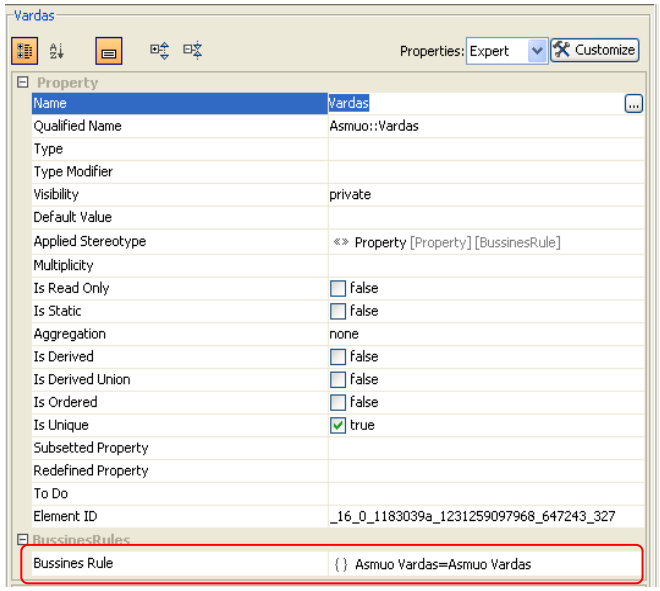
Paveikslėlyje nr. 32 yra pateiktas *BussinesRule* elemento savybių langas su užpildytais testo metu naudotais duomenimis. Šį elementą gali turėti keli klasių modelio elementai. Šį elementą turi tie elementai, kurie yra aprašomi šios veiklos taisyklės. Konkrečiu atveju BussinessRule elementą, kurio pavadinimas yra „Asmuo Amžius“, turės klasė *Asmuo* ir atributas *Amžius*.

### 5.9.2. IŠPLĖSTŲ CLASS, PROPERTY IR ASSOCIATION ELEMENTŲ ILIUSTRACIJOS

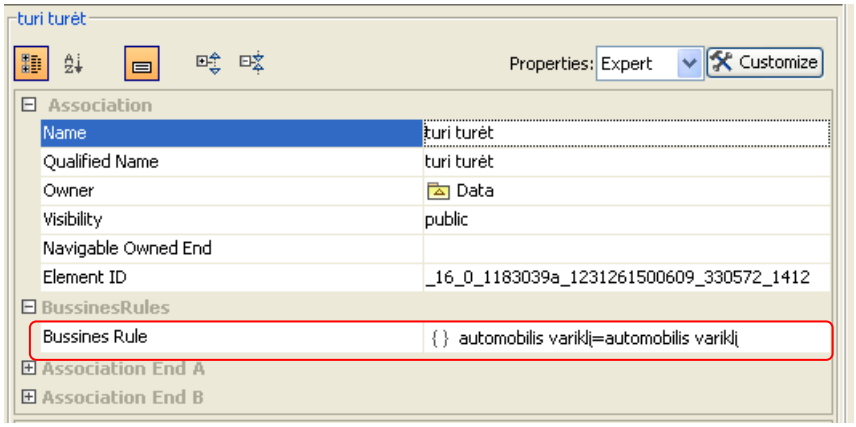
Išpėsti klasės, atributo ir asociacijos modelio elementai turi atitinkamai visus standartinius *Class, Property ir Association* tipo elementų atributus ir yra praplėsti *BussinesRule* atributu, kuriame yra priskiriamos veiklos taisyklės, kuriose dalyvauja konkretūs elementai.




33. pav. Praplėsto *Class* elemento savybių iliustracija




34. pav. Praplėsto *Property* elemento savybių iliustracija



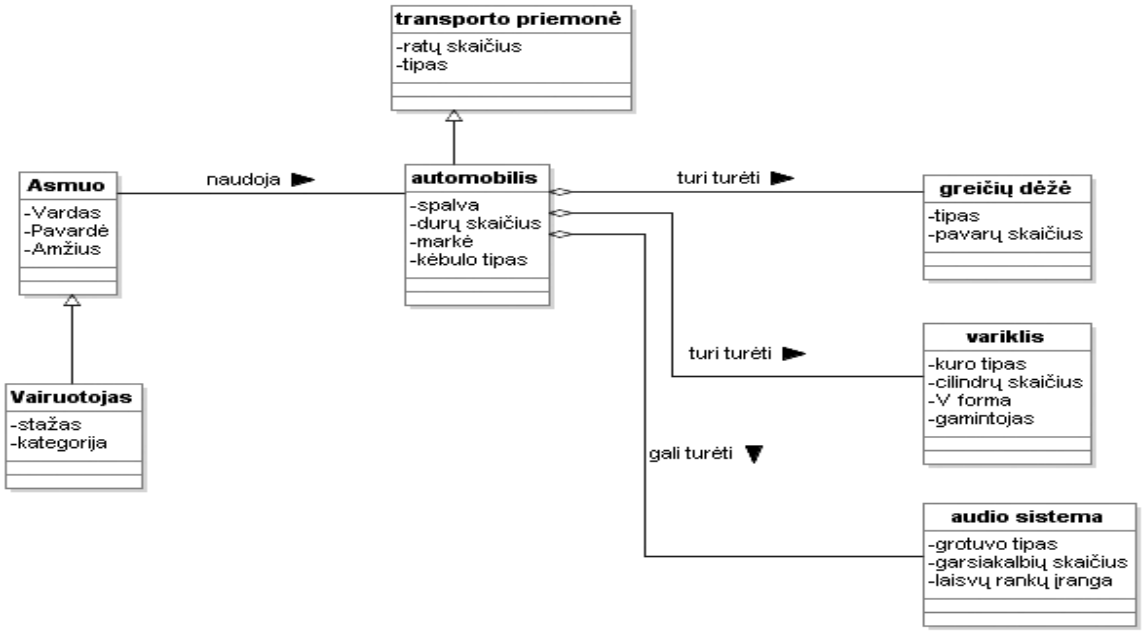

35. pav. Praplėsto Association elemento savybių iliustracija

Paveikslėliuose Nr. 33, 34, 35 yra pateikti klasės modelio praplėstų elementų savybių langų fragmentai, kuriuose raudonu stačiakampiu yra išskirtas elementų atributas, kuriuo jie buvo visi papildyti.

## *5.10. EKSPERIMENTO ATVEJAI*

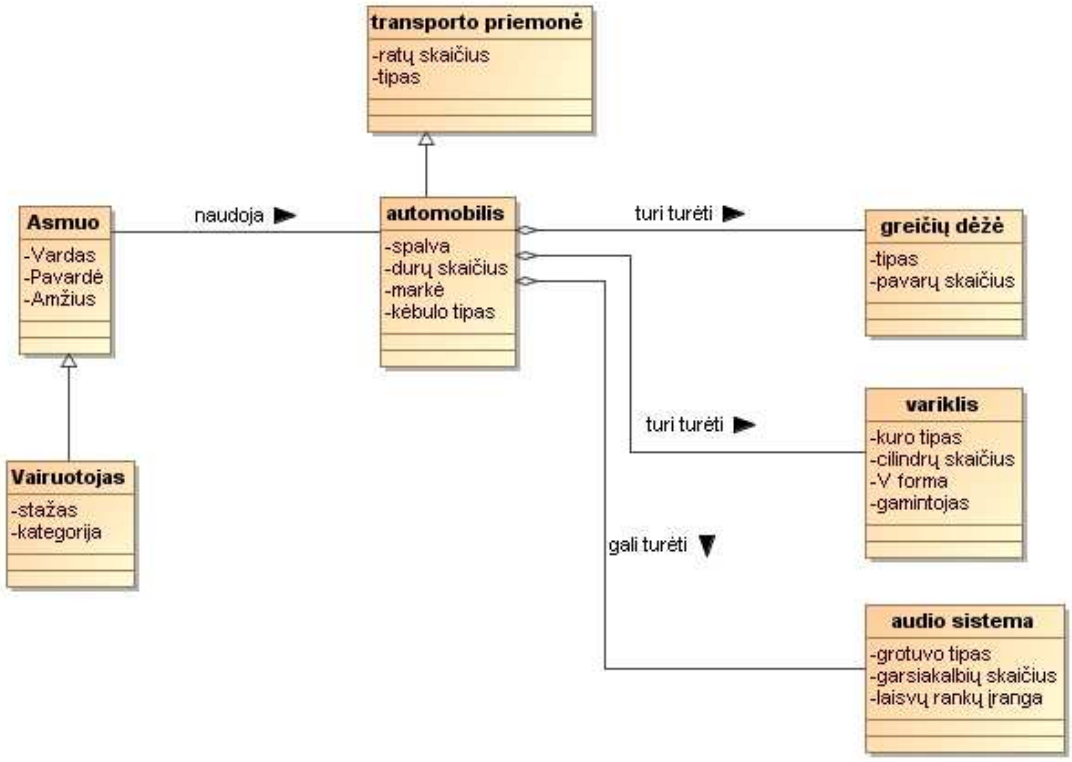
**Atvejis1:** MagicDraw Uml yra sukurtas tuščias klasių modelis ir yra pažymėtos visos klasės ir jų elementai.

### Tikimasis rezultatas




36. pav. Atvejis1. Tikimasis rezultatas įvykdžius algoritmą.

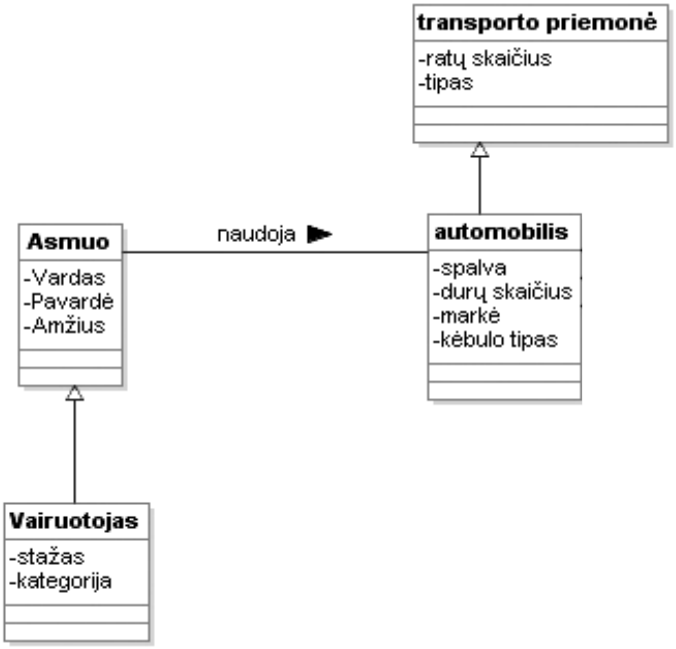
### Gautas rezultatas




37. pav. Atvejis1. Gautas rezultatas įvykdžius algoritmą.

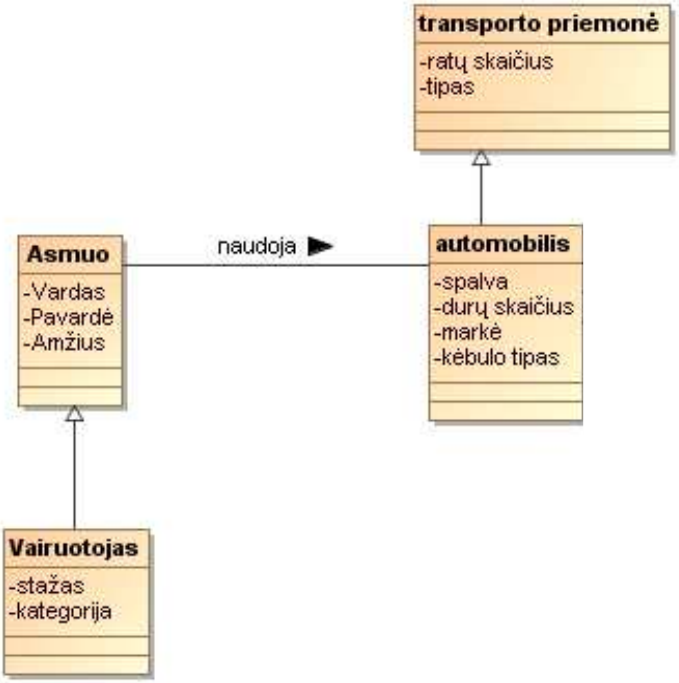
**Atvejis2:** MagicDraw UML yra sukurtas tuščias klasių modelis ir yra pažymėtos tik 4 klasės ir jų elementai.

### Tikimasis rezultatas




38. pav. Atvejis2.Tikimasis rezultatas įvykdžius algoritmą.

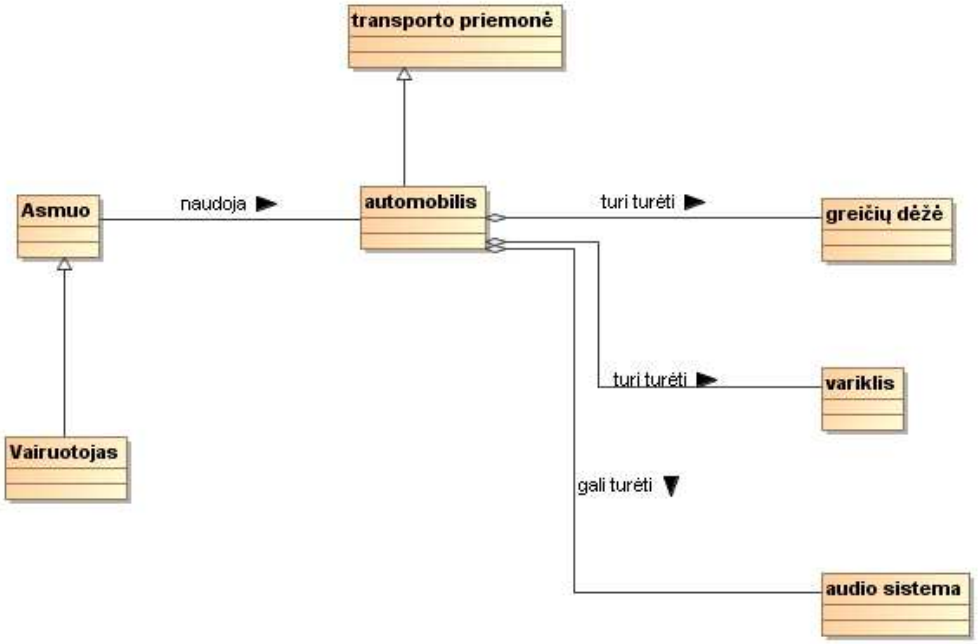
### Gautas rezultatas




39. pav. Atvejis2. Gautas rezultatas įvykdžius algoritmą.

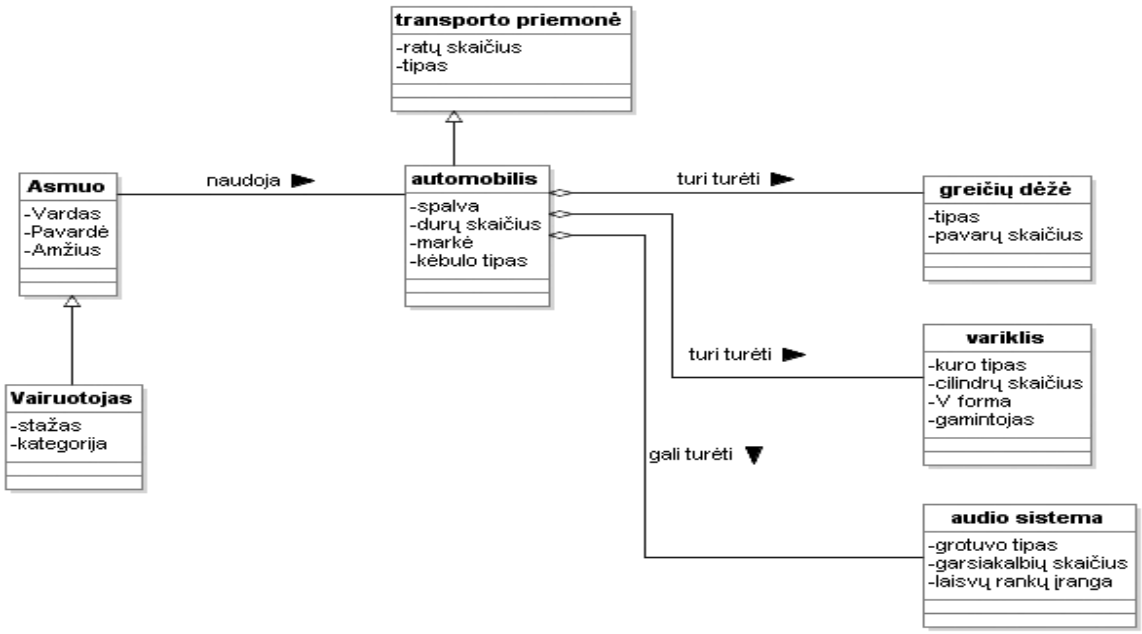
**Atvejis 3:** MagicDraw UML yra sukurtas klasių modelis su klasėmis be atributų.

**Pradiniai duomenys:**



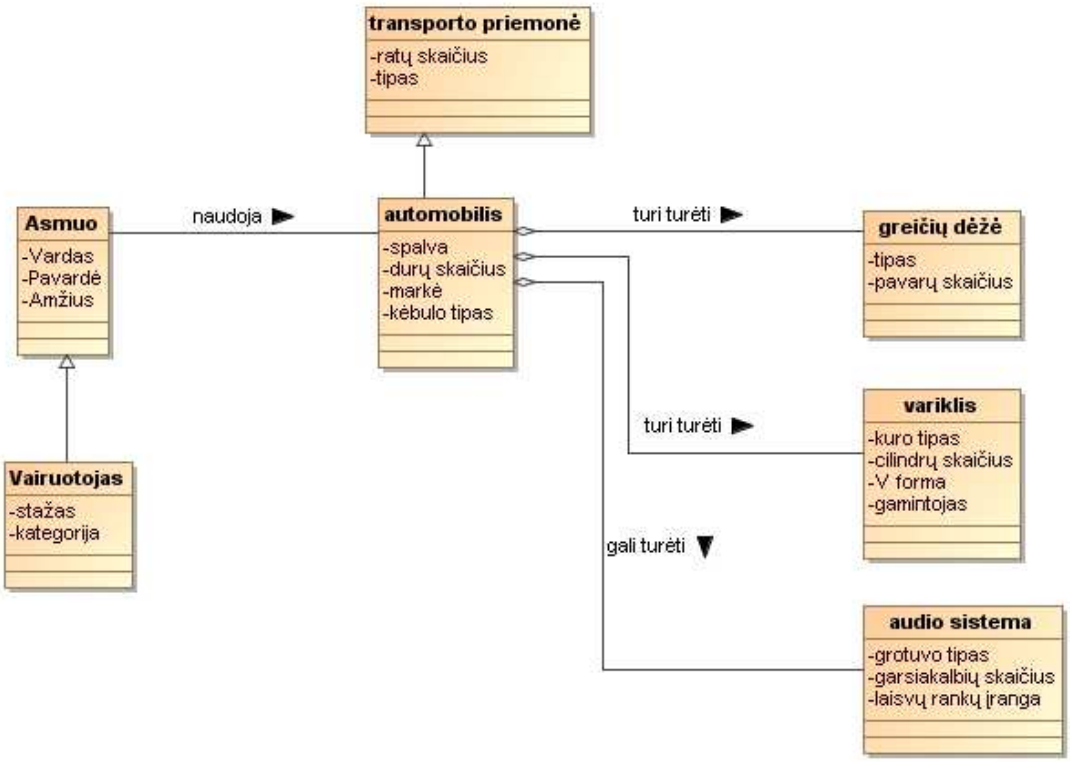

40. pav. Atvejis 3. Pradiniai atvejo duomenys

**Tikimasisi rezultatas**




41. pav. Atvejis 3.Tikimasis rezultatas įvykdžius algoritmą.

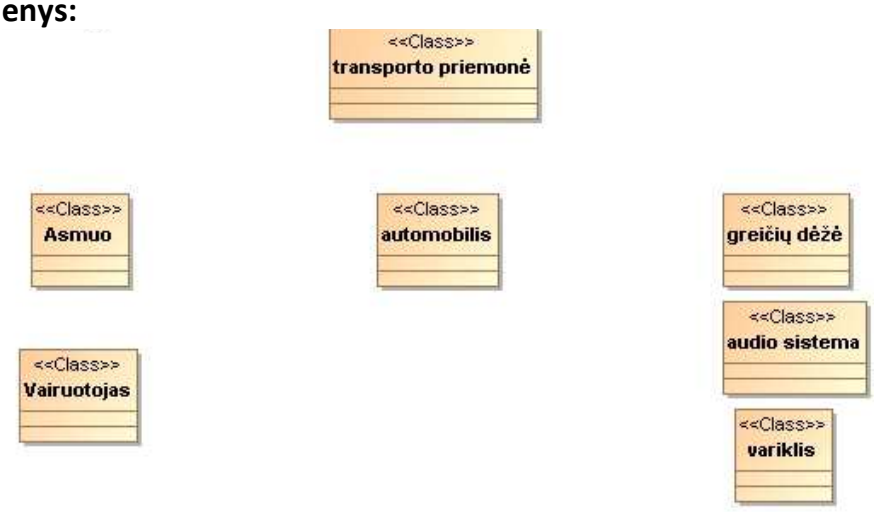
**Gautas rezultatas**




42. pav. Atvejis 3.Gautas rezultatas įvykdžius algoritmą.

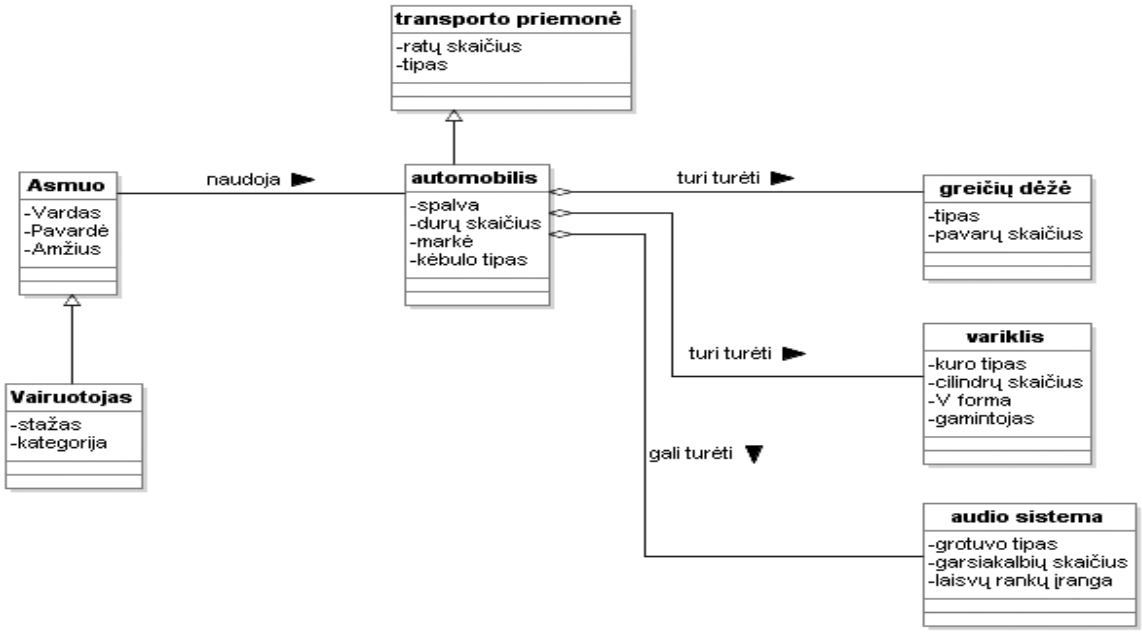
**Atvejis 4:** MagicDraw UML yra sukurtas klasių modelis su klasėmis be atributų ir be ryšių.

**Pradiniai duomenys:**



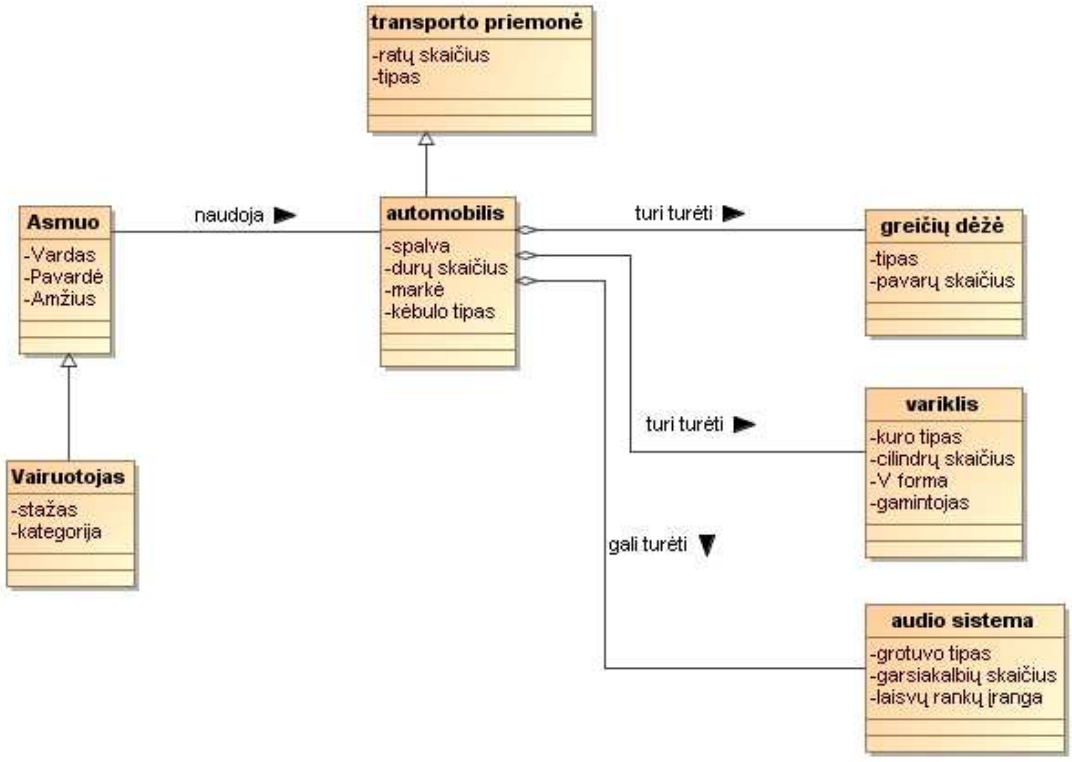

43. pav. Atvejis 4. Pradiniai duomenys

## Tikimasis rezultatas




44. pav. Atvejis 4. Tikimasis rezultatas įvykdžius algoritmą.

## Gautas rezultatas




45. pav. Atvejis 4. Gautas rezultatas įvykdžius algoritmą.

Ištestuoti atvejai parodė, jog realizuotas algoritmas veikia korektiškai ir patenkina lūkesčius.

## *5.11. REALIZACIJOS IR EKSPERIMENTO IŠVADOS*

1. Šio darbo realizavimo metu buvo praplėstas klasių metamodelis su *BussinesRule* elementu, kuris paveldėjo visas *Constraint* elemento savybes. Šis praplėtimas MagicDraw UML pakete yra atliekamas kuriant profilį ir išplečiant jau esančius modelio elementus papildomais elementais arba kuriant jiems ribojimus.
2. Realizacijai buvo sukurta ir naudota reliacinė duomenų bazė. Duomenų bazė buvo užpildyta pradiniais duomenimis, kurie yra reikalingi veiklos taisyklių tipams atpažinti. Taip pat buvo sukurtos procedūros, kurios realizuoja dalį algoritmo funkcijonalumo, kuris susijęs su duomenų bazėje esančių duomenų atrinkimu.
3. Kad būtų korektiškai suvestos veiklos taisykles į duomenų bazę buvo sukurtas programinis įrankis, kurio pagalba buvo žymiai pagreitintas testinių duomenų suvedimas.
4. Kadangi buvo ribotas darbo kūrimo laikas, buvo nuspręsta neatlikinėti rolės ir ribojimo tipų veiklos taisyklių. Nerealizuota analizuojamų klasės elementų pasirinkimo galimybė. Nebuvo realizuotas konfliktų sprendimas.
5. Atlikus keletą testų su realizuotu algoritmu, buvo įsitikinta, jog realizuotas MagicDraw UML įskiepis veikia korektiškai. Veikimo laikas nebuvo matuojamas, tačiau galima teigti jog šio metodo taikymas pagreitina veiklos taisyklių integravimą į klasių modelį, nes tai atliekama automatiškai. Šis sukurtas algoritmas gali ne tik papildyti klasių modelį veiklos taisyklių pagrindu, bet ir generuoti klasių modelį. Ši savybė labai praplečia šio algoritmo panaudojimą, kadangi galima iš surinktų reikalavimų natūralios kalbos šablonų pagalba automatiškai generuoti sistemos klasių modelį.

# 6. IŠVADOS

1. Atlikus veiklos taisyklių taikymo informacinių sistemų kūrime analizę, galima teigti, kad veiklos taisyklės dažniausiai yra naudojamos dalykinės srities specifikavimo etape ir jų pagrindu yra kuriama taikomųjų uždavinių logika bei ribojimai.
2. Išanalizuotas veiklos taisyklių užrašymas ir struktūrizavimas natūralios kalbos šablonų pagrindu. Natūralios kalbos šablonai yra įvairūs skirtingiems veiklos taisyklių tipams, tačiau pagrindinius jų elementus sudaro rezervuoti žodžiai, dalykinės srities terminai, faktai ir reikšmės.
3. Remiantis ISK mokslininkų darbais veiklos taisyklių taikymo projektinių modelių patikslinimo srityje, buvo sukurtas ir realizuotas algoritmas, kuriame yra naudojami veiklos taisyklių duomenų bazės analizavimo algoritmai. Sukurtas algoritmas papildomai leidžia apriboti nagrinėjamų klasių aibę, pasirinkti susijusių su ryšių kurimu konfliktų sprendimo būdą, kurti naujus klasės tipo elementus, kurti *BusinessRule* elementus, analizuoti rolės tipo taisykles.
4. Praktiniam algoritmo taikymui buvo sukurtas MagicDraw UML programinio paketo klasių metamodelį praplečiantis profilis ir įskiepis. Jį kuriant, išanalizuotas programinio paketo klasių metamodelis, jo praplėtimo galimybė, įskiepio kūrimo mechanizmas bei sukurta algoritmo, papildančio klasių modelį veiklos taisyklių pagrindu, projektinė specifikacija.
5. Galima teigti, kad sukurtas algoritmas yra bazinis klasių modelio patikslinimo veiklos taisyklių pagrindu algoritmas, kadangi jis analizuoja ir papildo arba kuria klasės modelio struktūrą. Ateityje algoritmą planuojama papildyti metodais, kurie analizuotų veiklos taisykles, aprašančias ribojimus bei logiką, kas leistų gauti tikslesnį klasių modelį.

# SUTRUMPINIMŲ IR TERMINŲ ŽODYNAS

**OMG** - (angl. Object Management Group) tarptautinis, atviras, ne pelno siekianti kompiuterių pramonės konsorciumas, kuriantis integracijos standartus.

**MDA** - (angl. model drivea- aproach) modeliai grindžiamas sistemų kūrimo principas.

**CIM** - (angl. computation independent model) nuo skaičiavimų nepriklausomas modelis.

**PIM** - (ang. Platform independent model) nuo platformos nepriklausomas modelis.

**PSM** - (angl. platform specific model) nuo platformos priklausantis modelis.

**EBNF** – (angl. Extended Backus-Naur) notacija (labiau žinoma kaip EBNF ar Extended Backus-Naur Form) tai formalus matematinis kalbos apibrėžimas, kuris yra sukurtas John Backus (kaip žinoma, gali būti ir Peter Naur) aprašyti Algol 60 programavimo kalbos sintaksei.

**VT** – veiklos taisyklė.

**VTS** – veiklos taisyklių saugykla.

**VTVS** – veiklos taisyklių valdymo sistema.

**CASE** – (angl. computer aided system engeneering) kompiuterizuota sistemų inžinerija.

## SANTRAUKA ANGLŲ KALBA

# BUSINESS RULES-BASED AUGMENTATION OF THE INFORMATION SYSTEM CLASS MODEL

*Summary*

Software development process of the collection and analysis phases are the key, since they belong to the software development phases of implementation. Information systems are the most used in the design specification language (UML, Z, DFD, OCL, etc.), understanding the need for specific knowledge. Customer's design language is often minimal, so he is confident in the knowledge the designer and the requirements of the interpretation. The design phase of the customer due misunderstood formally specified language, system design models, inaccuracies, often become apparent only in the realization phase. Erroneous interpretation of the IT professional knowledge effects editing is expensive and prolonged the duration of software product development.

This misunderstood was proposed to avoid the subject area specificate operational rules of natural-language ralios pattern. Activities are structured and unambiguous interpretation of the IT professionals and business representatives. The Subject field of the description method is a very recent and has not established standards for its application.

This work will be developed in the algorithm, the business rules-based adjustments to the calculations of independent (or platform-independent) level of data models. Meanwhile, prepare their own models for accelerating the development of design automation and to verify rules and model of class analysis.

# LITERATŪRA


[1.] J.Miller, J. Mukerji, June, 2003: „MDA Guide Version 1.0.1“, [žiūrėta 2007-10-03]Prieiga per internetą:<http://www.omg.org/cgi - bin/apps/doc?omg/03-06-01.pdf>.

[2.] Neville Haggerty, Grace Van Etten, Janet Wall, April 1, 2001:“Defining the Requirements for a Business Rule Repository“.

[3.] T. Skersys, S. Gudas: “Enterprise Model-based Generation of the Class Model*” Kaunas, 2007 birželis.

[4.] Taylor, J., March 22, 2006: „Why would I use a Business Rules Management System?“, [žiūrėta 2007-12-04] Prieiga per internetą: <http://oxygen.informatik.tucottbus.de/RealRules /?q= node /14>.

[5.] Business Rules Group: „Defining Business Rules ~ What Are They Really?“ , [žiūrėta 2007-11-14] Prieiga per internetą: <http://www.businessrulesgroup.org /first_paper/ br01c3.htm#s3b>.

[6.] IBM : “Different types of business rules“, [žiūrėta 2007-11-14] Prieiga per internetą:<http://publib.boulder. ibm.com/infocenter/wasinfo/v4r0/index.jsp?topic=/com. ibm.websphere.v4.doc/wasee_content/brb/concepts/crtypes.htm>.

[7.] Tomas Skersys. Kaunas, 2006:“ Informacijos sistemų kompiuterizuoto kūrimo metodas, grindžiamas veiklos taisyklėmis papildytu veiklos modeliu“ [Rankraštis]: daktaro disertacija: fiziniai mokslai, informatika (09P) 186 lap.

[8.] Ronald G. Ross and Gladys S. W. Lam, 2001:“RuleSpeakTM Sentence Templates, Developing Rule Statements Using Sentence Patterns“.

[9.] B.Theodoulidis, A.Youdeowei, 2000 :„Business Rules: Towards Effective Information Systems Development“.

[10.] Skersys T., Gudas S.: „The Enhancement of Class Model Development Using Business Rules. In: Lecture Notes in Computer Science: The Tenth Panhellenic Conference on Informatics „ (PCI’2005). Springer, Berlin (2005) 480-490.

[11.] OMG Available Specification, January, 2008 :“ Semantics of Business Vocabulary and Business Rules „.

[12.] Dr. Paul Dorsey ‚July 27, 2002:“The Business Rules Approach to Systems Development“.

[13.] Matignon, Carole-Ann, November 15, 2005:“Fix the requirements mess - use business rules“, [žiūrėta 2008-10-07] Prieiga per internetą: <http://www.edmblog.com/weblog/2005 /11/ fix_the_require.html>.

[14.] Anthony Rowe1, Susie Stephens2, Yike Guo1 :“The use of Business Rules with Workflow Systems“.

[15.] James Taylor on September 14, 2006 :„The magnificent 7 or 7 ways to use business rules in a modern software architecture“, [žiūrėta 2008-10-07] Prieiga per internetą: <http://www.ebizq.net/blogs/decisionmanagement/2006/09/the_magnificent_7_or_7_ways_to.php>

[16.] Blasum, Robert, DM Review Magazine, April, 2007: “Business Rules and Business Intelligence“, [žiūrėta 2008-10-09] Prieiga per internetą: <http://www.dmreview.com/issues/ 20070401/1079638-1.html>.

[17.] OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2T., [žiūrėta 2008-03-17] Prieiga per internetą:<http://www.omg.org/technology/documents/formal/uml.htm>.

[18.] Herbst, H., Myrach, T., 1997: „A repository system for business rules, Database Application Semantics“, R. Meersman, L. Mark (Eds.), London, pp. 119 - 138.

[19.] Kapocius, K., Butleris, R., 2002: „The business rules repository for information systems design, in: Research Communications of 6th East European Conference‘‘ (ADBIS’02), Bratislava, p. 64-77.

[20.] November, 2005: „Business Rules and Web Architecture: W3C Creates Rule Interchange Format WG“, [žiūrėta 2008-02-24] Prieiga per internetą: <http://xml.coverpages.org/ni2005-11-09-a.html>.

[21.] About the Object Management Group. [žiūrėta 2007-12-04] Prieiga per internetą: <http://www.omg.org/gettingstarted/gettingstartedindex.htm>.

[22.] „Business Rules Life Cycle“, [žiūrėta 2007-12-14] Prieiga per internetą: <http://www.yasutech.com/qrdn/brlc.htm>.

[23.] Lars Marius Garshol „BNF and EBNF: What are they and how do they work?“ [žiūrėta 2008-03-04] Prieiga per internetą:<http://www.garshol.priv.no/download/text/ bnf.html #id2.3>.

[24.] „System Requirements CASE Tool for Requirements Management and RAD“, [žiūrėta 2008-05-04] Prieiga per internetą: <http://www.leapse.com/models.htm>.

[25.] „MagicDraw OpenAPI“, MagicDraw UML install directory\openapi\docs\ UserGuide.pdf.

[26.] Tomas Skersys, Vaidas Pečiulis, Rimvydas Simutis, 2008.02: “Business Rules Specification using Natural Language-based Templates: Approach and Implementation”.

[27.] Tamulis, G., 2009: „Dalykinės srities kalbų kūrimo metodika UML MagicDraw sistemai“.

# PRIEDAI

## *6.1.PRIEDŲ TURINYS*

**Priedas Nr.1. Kompaktinis diskas su ataskaita, pristatymu ir programiniu kodu**

## Priedas Nr.2. Stored procedūrų programinis kodas

```
USE [BR]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sBROfObject]
@name nvarchar(50)
as
begin
SELECT   br.brule_id, br.brule_Name, br.expression, t.SystemName
from br_element as bre left join
 term as t on t.term_id = bre.term_id left join
businessrule as br on bre.brule_id = br.brule_id
where upper(t.systemname) = upper(@name)and bre.seqNumber = 1
end
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sBRResWord]
      @id bigint
AS
BEGIN
      select rwm.meaning, rwm.type
from br_element as bre, resword_value as rwv, resword_meaning as rwm
where bre.brule_id = @id and bre.resword_id = rwv.resword_id and rwv.elemmean_id =
rwm.rw_meaning_id
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sBRSecondTerm]
      @id bigint
AS
BEGIN
select t.systemname
 from br_element as bre, Term as t
  where bre.brule_id = @id and bre.seqNumber = 3 and bre.Term_id = t.term_id
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sFeatureOrRelationship]
      -- Add the parameters for the stored procedure here
      @type nvarchar(50)
-- jei relationship, tai sytemName reiskia rysio klases varda i kuria nukreiptas
rysys
-- jei feature, tai systemName grazina atributo pavadinima
AS
BEGIN
      SELECT     t.SystemName, F.Meaning, F.BRule_id, F.type
FROM         (SELECT     TOP (100) PERCENT rwm.Meaning,rwm.type, br.BRule_id,
bre.SeqNumber, bre.BRElement_ID
     FROM          ResWord_Value AS rwv left join
```

```
      ResWord_Meaning AS rwm ON rwv.ElemMean_id = rwm.RW_meaning_id LEFT OUTER
JOIN BR_Element AS bre ON bre.ResWord_id = rwv.ResWord_id LEFT OUTER
JOIN BusinessRule AS br ON br.BRule_id = bre.BRule_id
WHERE      (UPPER(rwm.Meaning) = UPPER(@type))
ORDER BY rwm.Meaning) AS F LEFT OUTER JOIN

BR_Element AS bre ON F.BRule_id = bre.BRule_id AND bre.SeqNumber = F.SeqNumber + 1
LEFT OUTER JOIN AS t ON bre.Term_id = t.Term_id
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sObjects]
AS
BEGIN
SELECT .SystemName
FROM SELECT     TOP (100) PERCENT rwm.Meaning,rwm.type, br.BRule_id,
bre.SeqNumber, bre.BRElement_ID
FROM ResWord_Value AS rwv left join
ResWord_Meaning AS rwm ON rwv.ElemMean_id = rwm.RW_meaning_id LEFT OUTER
JOIN BR_Element AS bre ON bre.ResWord_id = rwv.ResWord_id LEFT OUTER
JOIN BusinessRule AS br ON br.BRule_id = bre.BRule_id
WHERE (UPPER(rwm.Meaning) = UPPER('relationship'))and br.brule_id is not null
ORDER BY rwm.Meaning) AS F LEFT OUTER
JOIN BR_Element AS bre ON F.BRule_id = bre.BRule_id AND bre.SeqNumber = 1 LEFT
OUTER JOIN Term AS t ON bre.Term_id = t.Term_id
group by systemname
union
      SELECT     t.SystemName
FROM (SELECT     TOP (100) PERCENT rwm.Meaning,rwm.type, br.BRule_id,
bre.SeqNumber, bre.BRElement_ID
FROM ResWord_Value AS rwv left
join ResWord_Meaning AS rwm ON rwv.ElemMean_id = rwm.RW_meaning_id LEFT OUTER
JOIN BR_Element AS bre ON bre.ResWord_id = rwv.ResWord_id LEFT OUTER
JOIN BusinessRule AS br ON br.BRule_id = bre.BRule_id
WHERE (UPPER(rwm.Meaning) = UPPER('feature')) and br.brule_id is not null
ORDER BY rwm.Meaning) AS F LEFT OUTER
JOIN BR_Element AS bre ON F.BRule_id = bre.BRule_id AND bre.SeqNumber = 1 LEFT
OUTER JOIN AS t ON bre.Term_id = t.Term_id
group by systemname
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[getTermID]
      @name nvarchar(250)
AS
BEGIN
declare @id bigint

   set @id = (select Term_id from term where Upper(systemname)=upper(@name));
   if @id is null
     begin
        insert into term (systemname, expression) Values(@name, @name);
            set @id = (select Term_id from term where
Upper(systemname)=upper(@name));
      end
select @id
END
```