KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Tomas Sadauskas

# Informacinės sistemos klasių modelio kūrimas veiklos žinių pagrindu

Magistro darbas

Darbo vadovas
Doc. dr. T. Skersys

Kaunas, 2009

# Turinys

# Paveikslėlių turinys



# Lentelių turinys

# 1. ĮVADAS

Darbe nagrinėjama *tyrimo sritis* – kompiuterizuota informacinių sistemų inžinerija. Darbo *tyrimo objektas* yra veiklos modeliais grindžiamų informacinių sistemų kūrimo metodas.

Nepakankamos informacinių sistemų projektavimo priemonių galimybės labai išplečia projektavimo proceso trukmę. Dažnai projektavimo ciklas susideda iš kelių etapų. Paprastai pradžioje aprašoma probleminė sritis (kaupiamos žinios apie veiklą), specifikuojami vartotojo reikalavimai, po to sudarinėjami projektiniai sprendimai (sistemos klasių modelis ir kt.). Projektinių sprendimų kūrimas veiklos žinių pagrindu nėra automatizuotas, dėl to šis procesas užtrunka ilgai.

Paprastai projektavimo procesas yra iteracinis (ciklinis). Dažnai tenka pataisyti projektuojamos informacinės sistemos specifikaciją. Pakeitus ar papildžius veiklos žinių bazę, tenka pakeisti ir projektinius sprendimus (pvz.: klasių modelį). Kadangi procesas nėra automatizuotas, projektuotojui tenka pakartotinai analizuoti sudarytus projektinius sprendinius ir juos pakeisti ar papildyti.

Neautomatizuotas IS projektavimas ne tik užtrunka ilgiau, bet ir įveda didesnę klaidos pasitaikymo galimybę. T.y. žmogiškas faktorius gali lemti, kad bus atsižvelgta ne į visus veiklos žinių bazėje saugomus įrašus. Įrašai taip pat gali būti ne tinkamai interpretuojami ir t.t.

Numatomas sprendimas – metodas, kuris iš žinių bazėje saugomų žinių (probleminės srities veiklos aprašo), generuos informacinės sistemos projektinį sprendinį (konkrečiai, sistemos klasių modelį nuo platformos (kompiuterio, OS, realizuojamos IS platformos) nepriklausomam lygmenyje). Iš visų projektinių sprendinių, klasių modelis yra vienas iš svarbiausių.

Atsižvelgiant į šiuos faktorius atsiranda poreikis optimizuoti informacinės sistemos kūrimo procesą. Norint tai pasiekti reikia atlikti išsamią situacijos analizę. Tuo tikslu apsibrėžiami tiriamojo darbo tikslas bei uždaviniai.

*Tyrimo tikslas* yra suteikti informacinės sistemos projektuotojui galimybę automatizuotai generuoti informacinės sistemos (toliau IS) klasių modelį (vieną svarbiausių IS proj. sprendinių) nuo platformos nepriklausomam lygmenyje pagal turimas veiklos žinias (veiklos modeliai).

Tyrimo tikslas apibrėžia siekį optimizuoti IS kūrimo procesą jį automatizuojant. Konkrečiu atveju optimizavimas galimas automatizuojant perėjimą nuo veiklos modelių (nuo skaičiavimų nepriklausomi modeliai) prie IS projektinių sprendinių, konkrečiai klasių modelio (nuo platformos nepriklausomi modeliai). Tam reikalinga esamos IS projektavimo srities situacijos analizė siekiant surasti efektyvius metodus užsibrėžto tikslo įgyvendinimui.

Nėra plačiai žinomų realizuotų metodų ar priemonių, kurios leistų atlikti automatizuotą klasių modelio sudarymą pagal veiklos žinias (veiklos modelį). Tyrimo metu analizuojama esamų IS modeliavimo priemonių dokumentacija siekiant nustatyti metodo realizavimo galimybes. Taip pat analizuojami tyrimai šioje srityje (knygos, straipsniai, kitos publikacijos...).

Atsižvelgiant į užsibrėžtą tikslą, galima suformuluoti atliekamo tyrimo uždavinius, kuriais bus vadovaujamasi tolimesniuose analizės, projektavimo bei realizacijos darbuose. *Pagrindiniai tiriamojo darbo uždaviniai*:

1. Išanalizuoti esamą situaciją, kaip veiklos modeliai yra taikomi IS kūrimo procese.
2. Išanalizuoti Informacinių sistemų katedros (ISK) vykdomus tyrimus veiklos žiniomis grindžiamų IS kūrime.
3. „MagicDraw UML“ paketo analizė, siekiant išsiaiškinti paketo galimybes palaikyti tiriamajame darbe pateiktus pasiūlymus bei realizacijas.
4. Suprojektuoti metodo, realizuojančio IS klasių modelio kūrimo veiklos žinių pagrindu, sistemos prototipą.
5. Realizuoti suprojektuoto metodo prototipą bei jį ištestuoti (eksperimentinio pavyzdžio pagrindu).
6. Atlikti ISK vykdomų tyrimų veiklos žiniomis grindžiamų modelių sudarymo srityje analizę.
7. Atliktos galimų metodų (IS klasių modelio kūrimo veiklos žinių pagrindu) analizės pagrindu, pasiūlyti papildymus, pakeitimus pasiūlytiems metodams. Tai apima papildymų bei rekomendacijų pasiūlymus esamoms veiklos modelių saugyklų koncepcijoms (modeliams) bei esamiems algoritmams.

Tyrimo metu įgyvendinti visi užsibrėžti darbo tikslai. Taip pat realizuoti naujai, analizės bei projektavimo metu užsibrėžti reikalavimai tokie kaip: informacinės veiklos atskyrimas nuo materialios veiklos. Tokie sprendimai leidžia optimizuoti IS kūrimą.

Pagrindinis tyrimo rezultatas yra realizuotas metodo prototipas bei veiklos žinių bazė. Prototipas veikia „MagicDraw UML“ pakete, kaip laisvai pasirenkamas įskiepis. Tačiau įskiepio panaudojimui privaloma tūrėti darbo metu sudarytą profilį. Priešingu atveju klasių modelis nebus generuojamas dėl privalomų elementų stygiaus (stereotipizuotų klasių stygiaus).

Taip pat realizuota modeliavimo aplinka, kuri palengvina neautomatizuotą klasių modelio sudarymą. Palengvinama tai, kad vartotojui pateikiami klasių stereotipai ir jiems tinkami objektai (ryšiai, ribojimai). Tačiau automatizuotam klasių modelio sudarymui ši modeliavimo aplinka nėra privaloma.

Darbo metu pasiūlytas naujas algoritmas, kuris buvo grindžiamas ISK atliktais tyrimais. Buvo analizuojamos kelios koncepcijos, kaip klasių modelis galėtų būti gaunamas pagal turimas veiklos žinias. Remiantis atlikta analize, suprojektuotas naujas algoritmas.

Svarbu paminėti, kad galutinis sudaryto metodo rezultatas – sugeneruotas klasių modelis pagal savo koncepciją yra veiklos modelio atitikmuo. Tiksliau galima šį modelį įvardyti kaip veiklos objektų modelį, išreikštą klasėmis (nuo platformos nepriklausomų modelių lygmenyje). Tačiau tai taip pat nėra pilnas modelis, nes nėra sudaromi kardinalumai ryšiams tarp klasių. Taip pat nėra sudaromi ryšiai tarp srauto tipo klasių, nors pilnam modeliui šie ryšiai yra privalomi. Plačiau tai nagrinėjama analizės dalyje.

Darbo rezultatai patikrinti eksperimento metu. Eksperimente buvo naudojamas pavyzdinis modelis. Rezultatai parodė, kad gaunamas korektiškas modelis. Nustatyta palyginus gautus rezultatus su neautomatizuotai gautais rezultatais. Taip pat eksperimento rezultatai leido identifikuoti pasiūlyto metodo trūkumus.

*Darbo struktūra:*

*Analizės skyriuje* analizuojamos esamos informacinių sistemų modeliavimo priemonės, metodikos, siekiai [2, 15, 16, 17, 18]. Analizuojami vartotojų poreikiai siekiant nustatyti darbo reikalingumą [3]. Analizuojamos įvairios technologijos, veiklos žinių ir klasių modeliai, jų saugojimo bei panaudojimo galimybės [1, 6, 7, 10, 11, 12, 14]. Šios analizės tikslas – nustatyti galimybes bei priemones algoritmo realizavimui. Tuo tikslu taip pat analizuojamas paketo „MagicDraw UML“ paketas bei jo išplėtimo galimybės [17, 18, 22]. Taip pat analizuojami įvairūs pasiūlyti algoritmai, kurie nėra realizuoti [1, 4, 5, 8, 9, 11, 19, 20, 21]. Tačiau atlikta analizė apibrėžė gaires algoritmo prototipo realizavimui.

*Projektavimo skyriuje* sudaryti visi projektiniai sprendiniai reikalingi prototipo realizavimui. Atlikti UML meta-modelio išplėtimo trūkstamais komponentais projektavimo darbai [22, 23, 24]. Darbų rezultatas – suprojektuotas „MagicDraw UML“ paketo profilis. Taip pat suprojektuotas įskiepio realizavimas. Šie darbai apima algoritmo bei įskiepio vartotojo sąsajos modelio sudarymą.

Išanalizuota pasiūlyta veiklos žinių saugykla. Analizės pagrindu sudarytas pasiūlymų, papildymų sąrašas, kuriais remiantis suprojektuota optimizuota veiklos žinių saugykla.

*Realizacijos skyriuje* aprašoma visų sprendinių realizacija. Aprašomos visų komponentų realizacijos eiga, metodai ir technologijos. Taip pat nurodomos iškilusios problemos ir pastebėti trūkumai. Bendros realizuotų komponentų charakteristikos atsikleidžiamos atlikus eksperimentą.

Prototipas testuojamas eksperimentinio pavyzdžio pagrindu. Atliktas eksperimentas parodo metodo efektyvumą, atskleidžia privalumus ir trūkumus. Ši informacija leidžia atlikti darbo apibendrinimus bei suteikia galimybę apibrėžti pasiūlymus metodo optimizavimui.

# ENTERPRISE KNOWLEDGE-BASED DEVELOPMENT OF THE INFORMATION SYSTEM CLASS MODEL.


## SUMMARY

Insufficient capabilities of information systems design tools extends the duration of the design process. Often, the design cycle consists of several stages. Usually at the beginning, then the domain is described (collected knowledge of the activities), analytics specifies user requirements. Only after this design solution (the system class models, etc.) can be developed. Enterprise Knowledge-based Development of the design solutions is not automated therefore the duration of the design process is long.

There are a lot of disadvantages of not automated process. Non-automated design not only takes longer, but also creates a greater possibility of the error occurrence. That is because the human factor can be taken into account because, that not all the activities stored in the knowledge base. Also records may not be correctly interpreted, and etc.

Thus, the expected solution is a method which will be used to generate information system design solutions (specifically - information system class model of the platform independent level) based on the knowledge (the description of the domain) stored in knowledge base. Of all design solutions, the class model is one of the major information system project solutions. Therefore this should efficiently optimize the design process of information system.

In this paper you will discover the possible solution – proposed prototype of algorithm and prototype of knowledge base. Solution is realized on “MagicDraw UML” platform therefore it can be potentially used by IS engineers. Prototype is functional and is based on research of class model generation based on Enterprise Knowledge‘s.



*Key words:* UML, Class model development, Enterprise Knowledge-based Development, IS development.

# 2. VEIKLOS ŽINIOMIS GRINDŽIAMO KLASIŲ MODELIO SUDARYMO METODŲ ANALIZĖ

## 2.1. Analizės uždaviniai

Atlikta analizė turi padėti pasirinkti tinkamą darbo vykdymo metodiką. T.y. turi padėti pasirinkti reikiamas kryptis, technologijas bei įrankius skirtus įgyvendinti užsibrėžtą darbo tikslą.

Šios analizės uždaviniai yra:

1. Išanalizuoti metodo reikalingumą, klasių modelio sudarymui pagal veiklos žinias (tyrimo objekto bei vartotojų analizė);
2. Išanalizuoti esamą situaciją, kaip veiklos modeliai yra, kaip gali būti taikomi IS kūrimo procese ir kaip, kokiose priemonėse yra realizuotas automatizuotas modelių sudarymas (CIM->PIM->PSM->Kodas perėjimas (žiūrėti.: terminų žodynas);
3. Išanalizuoti klasių modelio sudarymo pagal veiklos žinias galimybes;
4. Išanalizuoti Informacinių sistemų katedros (ISK) vykdomus tyrimus veiklos žiniomis grindžiamų IS kūrime;
5. Išanalizuoti „MagicDraw UML“ paketo funkcionalumo išplėtimo bei pakeitimo galimybes.

## 2.2. Analizės metodai

Pagrindinis analizės metodas – informacijos šaltinių (literatūra, straipsniai, publikacijos, ...), pasirinktąja darbo tema, paiešką bei analizė.

Kadangi pasirinktai darbo temai nėra realizuoto modelio, tik ISK bei kitų autorių pasiūlyti modeliai, tai pagrindiniai informacijos šaltiniai yra straipsniai. Atradus reikiamą informacijos šaltinį, aptinkamos nuorodos į kitus su šia tema susijusius šaltinius. Tada analizuojami nauji šaltiniai.

Kitas metodas – ISK pasiūlyto metodo analizė. T.y. bus analizuojamas pasiūlytas metodas (modelis) siekiant jį pagerinti.

## 2.3. Tyrimo objekto analizė

Didėjant poreikiui automatizuoti – kompiuterizuoti verslo procesus bei kurti įvairias informacines sistemas, didėja ir sistemų projektavimo poreikis. Norint, kad sistema būtu realizuojama kuo greičiau, reikia, maksimaliai, pagreitinti projektavimo ir bendrą IS kūrimo (angl. infomation system develpment - ISD) procesą. Kadangi IS kūrimas paprastai yra iteracinis procesas, labai svarbu, kad iteracijų kiekis būtų kuo mažesnis. Tai galima įgyvendinti gerai atlikus pirmas IS kūrimo proceso dalis: 1) informacijos surinkimą bei specifikacijos sudarymą ir 2) IS projektavimą. Šiuos du etapus (specifikaciją bei projektavimą) reikia ne tik pagreitinti, bet ir užtikrinti, kad jie būtų išsamūs, atsižvelgiant į visus reikalavimus.

Paprastai projektavimas atliekamas panaudojant IS projektavimo priemones [2], sudarinėjant įvairius IS modelius (klasių modelis, veiklos modelis, ryšių modelis ir t.t.). Tačiau informacijos rinkimas, specifikacijos sudarymas ir perėjimas prie projektavimo nėra automatizuotas ir atlikus pirmąjį IS kūrimo etapą, vėl nuo pradžių reikia pradėti projektavimą. Perėjimas nuo vieno sudaryto modelio prie

kito nėra automatizuotas. T.y. projektuotojai paprastai ne pilnai vadovaujasi ar išvis nesivadovauja ankstesniame projektavimo etape sudarytais modeliais, o dirba pagal surinktą specifikaciją bei pagal savo darbo metu įgytą praktiką. [3]

Vienas iš būdų pagreitinti ir pagerinti projektavimo etapą – automatinis modelių generavimas pagal sukauptą informaciją (sudarytą žinių bazę). Idealiu atveju perėjimas nuo modelio prie modelio turėtų būti visiškai automatizuotas (ypač vėlesniems projektavimo modeliams). Idealiu atvejų tūrėtų būti toks automatizuotas perėjimas: veiklos modelis (nuo realizacijos nepriklausomas modelis – CIM) -> projektiniai sprendiniai (nuo platformos nepriklausomas modelis – PIM) -> platformai pritaikytas modelis (PSM) -> programos kodas [1].

Vienas iš svarbiausių uždavinių, skirtų IS kūrimo proceso pagerinimui – perėjimo nuo modelio prie modelio automatizavimas, kur pagrindinis tikslas – automatizuoti CIM->PIM parėjimą

## 2.4. Vartotojų analizė

### 2.4.1. Vartotojų tipai ir savybės

Darbe reikia atsižvelgti į kelias vartotojų grupes (tipai). Pagrindinė vartotojų grupė – IS kūrėjai - projektuotojai, nes šio tiriamojo darbo rezultatas – priemonė, leidžianti pagreitinti ir pagerinti IS kūrimo procesą. Ši vartotojų grupė yra gerai susipažinusi su IS kūrimo terminologija, todėl jiems neatsiranda sunkumų suprasti projektavimo metu sudarytus modelius (modeliai standartizuoti). Tačiau jiems gali atsirasti sunkumų skaitant ir interpretuojant užsakovų reikalavimus (specifikaciją). Problemai išvengti reikia naudoti žinių bazę.

Kita vartotojų grupė – IS užsakovai. Paprastai projektuojant IS reikalavimai būna patikslinami bei koreguojami darbo metu. Ši vartotojų grupė taip pat įtraukiama į vartotojų aibę, atliekant šį projektą.

### 2.4.2. Vartotojų tikslai ir problemos

IS kūrėjų pagrindinis tikslas – automatizuotas klasių modelio generavimas pagal veiklos žinių bazėje sukauptą informaciją. Taip pat tikimasi, kad veiklos žininių bazėje saugoma informacija bus lengvai suprantama IS užsakovams. Esama problema – projektavimo metu gaunami modeliai yra suprantami tik projektavimo technologijas ir jų terminologiją suprantantiems specialistams.

Norint pagerinti IS kūrimo procesą, reiktų atsižvelgti į tai, kad nors užsakovams labai sudėtinga suprasti projektavimo metu sudarytus modelius, jiems turėtų būti aiški veiklos žinių bazė, kurią jie galėtų peržiūrėti bei pateikti pakeitimus (papildymus, pataisymus). Tai svarbu, nes darbo uždavinys – klasių modelis (PIM) generuojamas veiklos žinių pagrindu (CIM). Šiuo metu užsakovams pateikiami įvairūs modeliai (gauti projektavimo įrankiais), kaip IS specifikacija, tačiau tai nėra užsakovams aiški informacija.

### 2.4.3. Vartotojų analizės apibendrinimas

Vartotojų analizė padėjo identifikuoti vartotojų tipus ir pagrindinius darbo uždavinius projektuojant ir realizuojant prototipą.

Bendri vartotojų analizės rezultatai pateikiami lentelėje 1.

Lentelė 1 Vartotojų tipų palyginimas

| Vartotojų tipas | IS kūrėjai | IS užsakovai |
|---|---|---|
| **1. Kompiuterinis raštingumas** | Aukšto lygio (ekspertai) | Žemo – vidutinio lygio |
| **2. Tikslas** | Greitas IS sukūrimas | Greitas IS sukūrimas |
| **3. Projektinių modelių suprantamumas:** | Aukšto lygio | Žemo – vidutinio lygio |
| **3.1. Veiklos modeliai** | Gerai suprantami | Suprantami |
| **3.2. Klasių modelis** | Gerai suprantamas | Sunkiai suprantamas |

Lentelėje pateikiama, kad IS kūrėjai turi daug patirties dirbant su kompiuteriais. T.y. jų kompiuterinis raštingumas – aukšto lygio. Tuo tarpu IS užsakovų darbo sritis gali būti nesusijus su kompiuteriais. Dėl šios priežasties jų kompiuterinis raštingumas gali būti labai žemo lygio.

Priešingai nei IS kūrėjams, IS užsakovams gali būti sunkiai suprantami atitinkami projektiniai modeliai. Dėl šios priežasties reikia optimizuoti modelių pateikimą, informatyvumą.

Abiems vartotojų tipams yra bendras tikslas – greitas IS sukūrimas. Dėl šios priežasties yra svarbus numatyto metodo realizavimas.

## 2.5. Esamų sprendimų analizė

### 2.5.1. Modeliavimo siekiai

Tam tikri IS projektavimo siekiai numato veiklos modelius, kurie turėtų būti struktūrizuotos informacijos, apie realaus pasaulio (veiklos srities) žinias, šaltiniai. Jie tūrėtų būti naudojami IS kūrimo ciklo įvairiuose etapuose (vartotojų reikalavimų analizėje, specifikacijoje, detalių IS projektų, sprendimų sudaryme, ...).

Yra daugybė grafinių IS projektavimo bei verslo procesų modeliavimo įrankių. Taip pat yra daug veiklos modeliavimo metodikų ir siekių (CIMOSA, GERAM, IDEF suite, GRAI) [15], standartų (ISO 14258, ISO 15704, PSL, ISO TR 10314, CEN EN 12204, CEN 40003) bei veiklos modeliavimą leidžiančių priemonių [16]. Tačiau išlieka problema – kaip integruoti šiuos įrankius ar priemones siekiant automatizuoti IS projektavimo procesą.

Tendencija tapo CASE įrankių funkcijų perkrovimas tobulinant esamus IS projektavimo įrankius. Taip pat didėja galimų diagramų – modelių kiekis, tačiau nėra realizuojamas integralumas tarp jų. T.y. nėra sudaromos galimybės gauti vienus modelius iš kitų. Tai lemia tam tikrų modelių atsisakymą projektuojant IS. Tai daroma siekiant pagreitinti IS projektavimo procesą, nors taip gali nukentėti IS kokybė, nes sistema nebus pilnai išanalizuota.

Norint išvengti šių kylančių problemų, reikia padidinti integralumą tarp IS ir veiklos modeliavimo. T.y. reikia suderinti šiuos du procesus ir net automatizuoti (pvz. klasių modelio generavimas veiklos žinių bazės pagrindu). Vienas iš siekių yra unifikuotos veiklos modeliavimo kalbos (UEML) įvedimas.

### 2.5.2. UEML koncepcija

Kylant poreikiui panaudoti veiklos žinių modelį IS kūrime, bandoma rasti įvairių sprendimų, kaip susieti šį modelį su IS projektavimo priemonėm. Vienas iš būdų – normalizuoto veiklos žinių meta-

modelio apibrėžimas, kuris apibrėžtų pagrindinių (branduolio) elementų integralumo struktūrą, sukūrimą. Šie elementai galėtų būti paimti iš skirtingų veiklos modeliavimo koncepcijų bei metodologijų. Ir pagrindinis siekis būtų – apibrėžtos veiklos modeliavimo kalbos (angl. Unified Enterprise Modeling Language) sukūrimas.

Šiuo metu viena iš populiariausių IS projektavimo kalbų yra OMG pasiūlytas UML. Tačiau UML nėra pilnai tinkama bei ne pilnai atitinka reikalavimus probleminės srities (angl. Domain) žinių bazės modeliavimui projektuojant IS.

Šiuolaikinis IS projektavimas tūrėtų būti grindžiamas žiniomis (žinių baze). Projektavimo sistema tūrėtų būti susieta su bendra veiklos žinių saugykla (angl. Repository). Saugykloje turi būti apibrėžtos bendros tiek organizacijos verslo veiklai, tiek ir IS projektavimo veiklai skirtos, žinios.

Tad pagrindiniai UEML versijų komponentai:

UEML branduolys – **UEMLcore** = {Objects (Event, Time, Agent, Process, Activity, Function, Input Object, Output Object,

Environment); Relations (Raises, Triggers, Produces, Happens at, Made of, Performs, Used by)}

Tai pirmoji UEML versiją. Joje apibrėžti pagrindiniai veiklos meta-modeliui bei veiklos modeliavimui reikalingi komponentai. Pagrindiniai komponentai būtų procesas bei funkcija. Taip pat veiklos ir srautai. Tačiau vėlesnėse versijose pagrindiniai komponentai skiriasi.

UEML modelis – **UEML_model** = {

UEML_Object (UEML_model,

Object (Information Object,

Resource (HumanResource, MaterialResource)),

Port (ResourceRole, Anchor (InputPort, OutputPort, ConnectionOperator), Geometry),

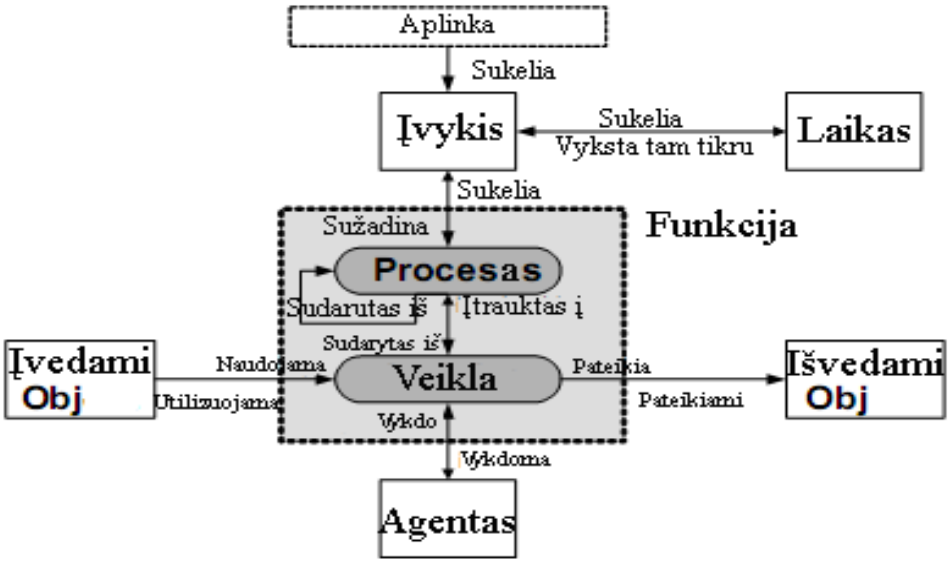
Flow (IOFlow, ResourceFlow, ControlFlow (TriggerFlow, ConstraintFlow)),

Activity (InputPort, OutputPort)),

Relations (Precedence relation, HasIP, HasOP, Contains, IOcarried, Object_carried,

Resource_carried)} [12].

Tad atitinkamai UEML pasiūlyto veiklos meta-modelio schema galima atvaizduoti grafiškai. Modelio schema pateikiama sekančiame paveikslėlyje:




Pav. 1 Veiklos meta-modelis pagal UEML [12]

Svarbu paminėti, kad skirtingose UEML versijose yra skirtingi pagrindiniai komponentai. Naujesnėje versijoje pagrindiniai komponentai lieka tik du – veikla (angl. activity) bei srautas (anlg. flow). Šiuo atveju nelieka tokių komponentų iš senesnės versijos kaip procesas ar funkcija. Šie komponentai išlieka sutraukti į veiklos (angl. activity) elemento struktūrą. Abi (tiek naujesnė, tiek senesnė) UEML versijos apibrėžia tik du pagrindinius elementų tipus. Tai objektai bei ryšiai tarp jų. Šiuo atveju nėra atsižvelgta į dinaminius veiklos modeliavimo aspektus. Taip pat nėra atsižvelgta į sąveiką (angl. sequence) tarp veiklos srities elementų [11].

### 2.5.3. Valdymas – funkcija bei procesas

Sistemų ir valdymo teorija (angl. System and control theory) teigia, kad sistemos valdymas yra efektyvus tik tada, jei yra išpildytos dvi sąlygos. Viena jų – turi būti naudojami valdymo ciklai (angl. feedback loops). Į šiuos valdymo ciklus reikia atžvelgti (juos reikia panaudoti) projektuojant veiklos žinių meta-modelį [10].

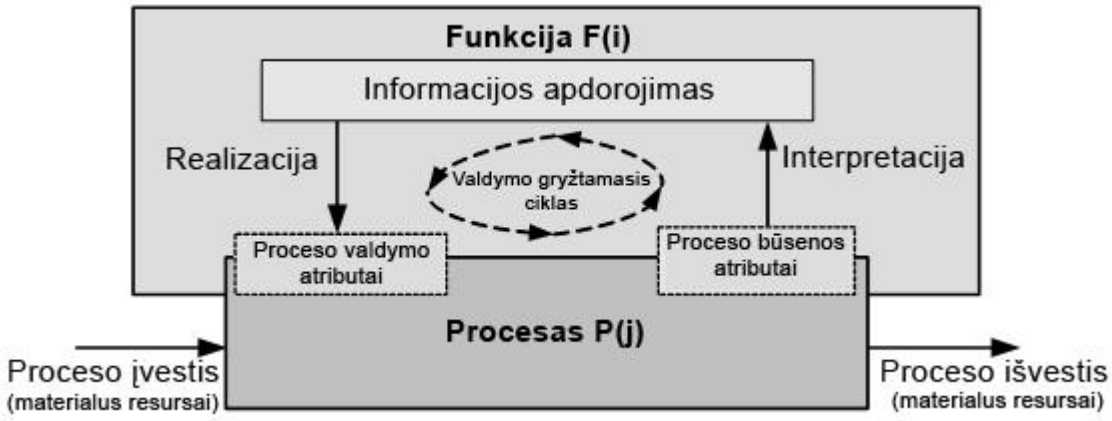
Kitas sistemų ir valdymo teorijos teiginys, apibrėžia tipišką sistemos struktūrą – realią sistemą ir jos vidinius mechanizmus. Apibrėžiamos privalomos sudėtinės dalys: realaus pasaulio procesas, valdymo sistema ir valdymo ciklai, kurie sudaro informacijos srautus (valdymo srautus) tarp procesų ir valdymo sistemos [10].

Pagal ankstesnius analizuotus apibrėžimus bei sistemų ir valdymo teorijos taisykles, galima teigti, kad sistemos sudedamosios dalys yra: procesas, funkcija, valdymo sistema ir valdymo (grįžtamieji) ciklai.

- Procesai gali būti sudaryti iš kelių procesų sekos.
- Funkcijos savo ruožtu susideda iš valdymo sistemos ir grįžtamųjų ciklų.
- Valdymo sistema susideda iš kelių privalomųjų dalių: duomenų apdorojimo bei sprendimų priėmimo. Ir šios dalys siejamos duomenų srautais.

- Valdymo (grįžtamieji) ciklai sukuria sąsają tarp procesų ir valdymo sistemos. Taip valdymo sistema pakeičia procesų būklės atributų seką į savą procesų valdymo atributų seką, tokiu būdu įtakodama pačią procesų eigą.
- Valdymo funkcijos apibrėžiamos kaip privaloma žingsnių seka. Tiksliau jos apibrėžiamos kaip sąveika tarp valdymo sistemos ir valdymo (grįžtamųjų) ciklų. Ši sąveika yra tarsi informacijos perdavimo procesas.

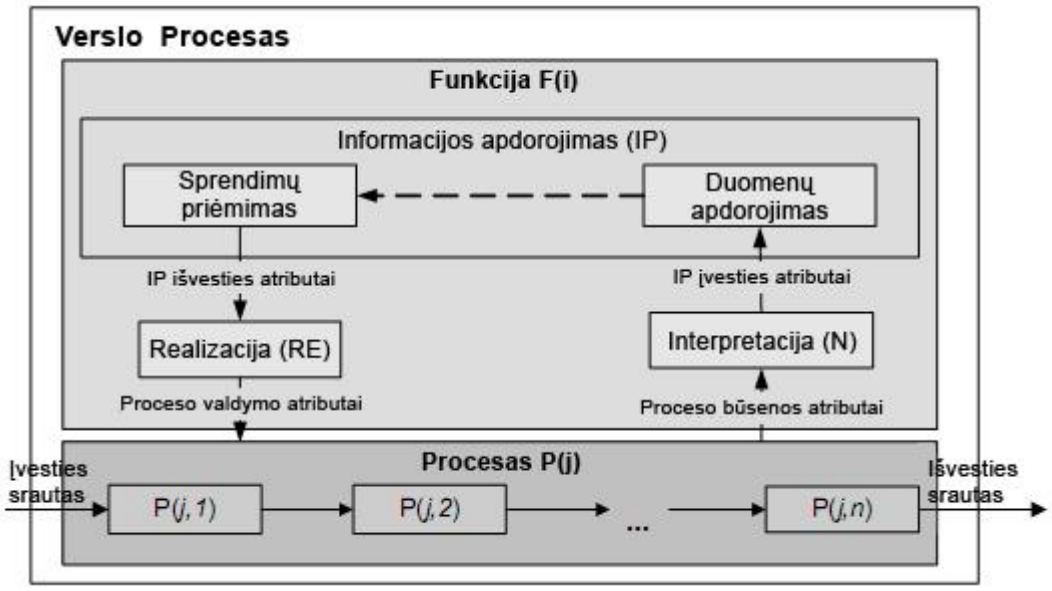
Visą šia sąveiką galima atvaizduoti grafiškai:




Pav. 2 Formalizuotas sistemos funkcijos ir proceso sąveika[11]

Schemoje pateikta, valdymo funkcijos kaip vienas sudėtinis veiklos modelio objektas, susidedantis iš valdymo sistemos ir grįžtamųjų (valdymo) ciklų bei tam tikros numatytos sąveikų tarp jų sekos.

Valdymo funkcija susideda iš kelių žingsnių (interpretavimo, informacijos apdorojimo ir realizacijos). Šie žingsniai apibrėžia valdymo ciklą. Šią schemą galima atvaizduoti detaliau. Pavyzdys pateikiamas sekančiame paveikslėlyje, kur atvaizduojama bendra verslo proceso schema.



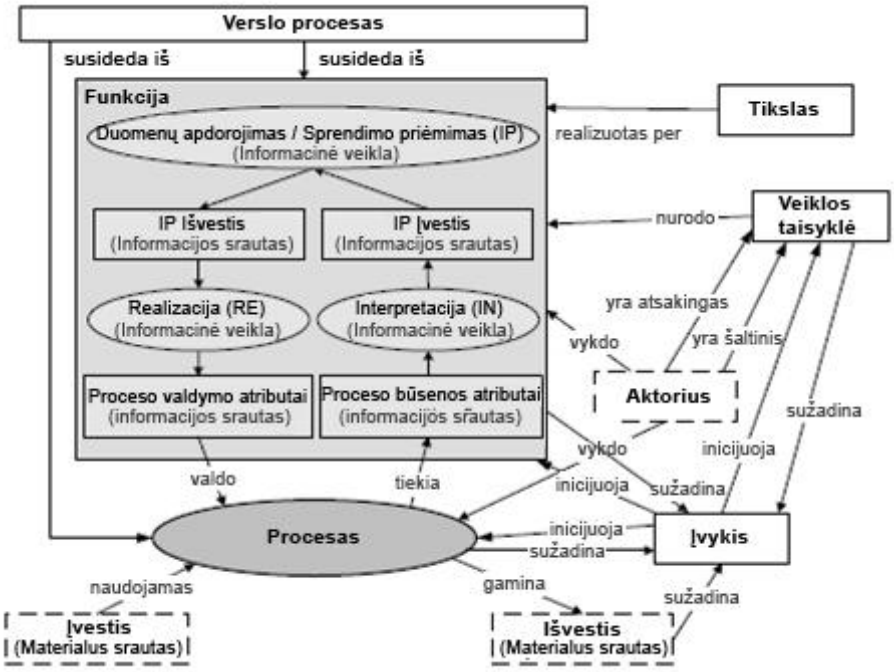

Pav. 3 Struktūrinis veiklos proceso modelis (funkcijos ir proceso sąveika)[13]

Šiame modelyje procesas pateikiamas kaip procesų seka. Tai gali būti tiek vienas procesas, tiek procesų kaskada, tačiau bendroje koncepcijoje žvelgiama, kaip į vieną procesą.

Šios koncepcinės schemos pagrindu toliau galima sudarinėti veiklos žinių meta-modelį.

### 2.5.4. Veiklos žinių bazė – Veiklos žinių meta-modelis (EMM)

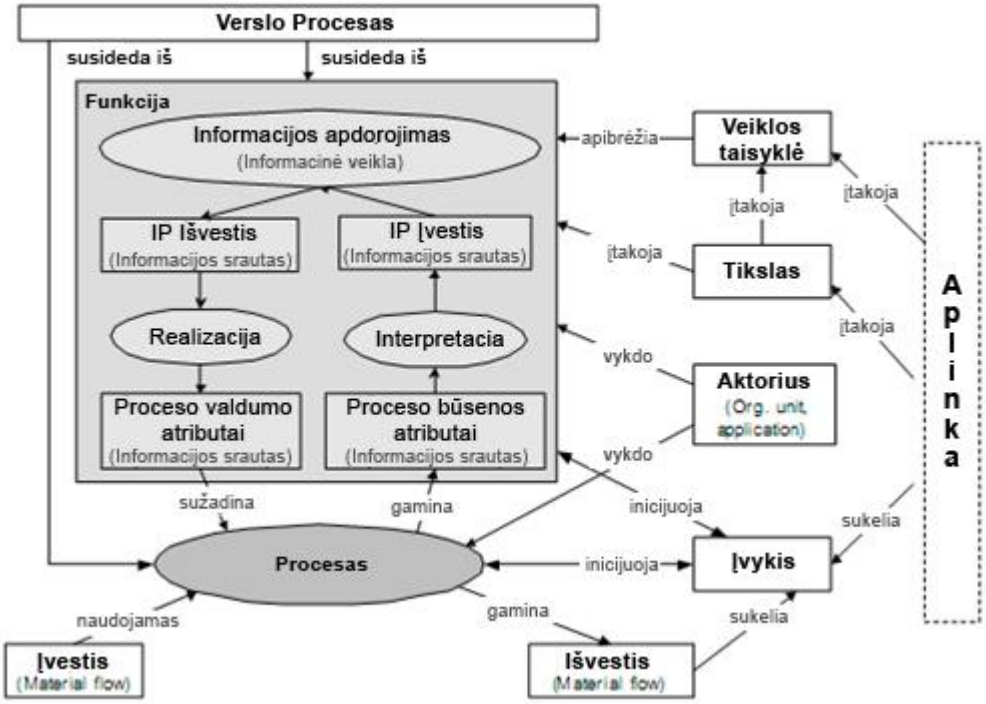
Veiklos žinių bazės sudarymas tūrėtų būti pirmas žingsnis projektuojant IS. Tuo labiau, kai UML pagrindu sudarytuose, modeliais grindžiamos architektūros (MDA), metoduose yra numatytos veiklos žinių modelio panaudojimo galimybės [1]. Tačiau pats UML nepalaiko pagrindinių reikalavimų objekto, veiklos žinių modeliavimui, projektuojant IS. Veiklos žinių meta-modelio koncepcinė schema pateikiama pav. 1.:




Pav. 4Veiklos žinių meta-modelio koncepcinė schema [1].

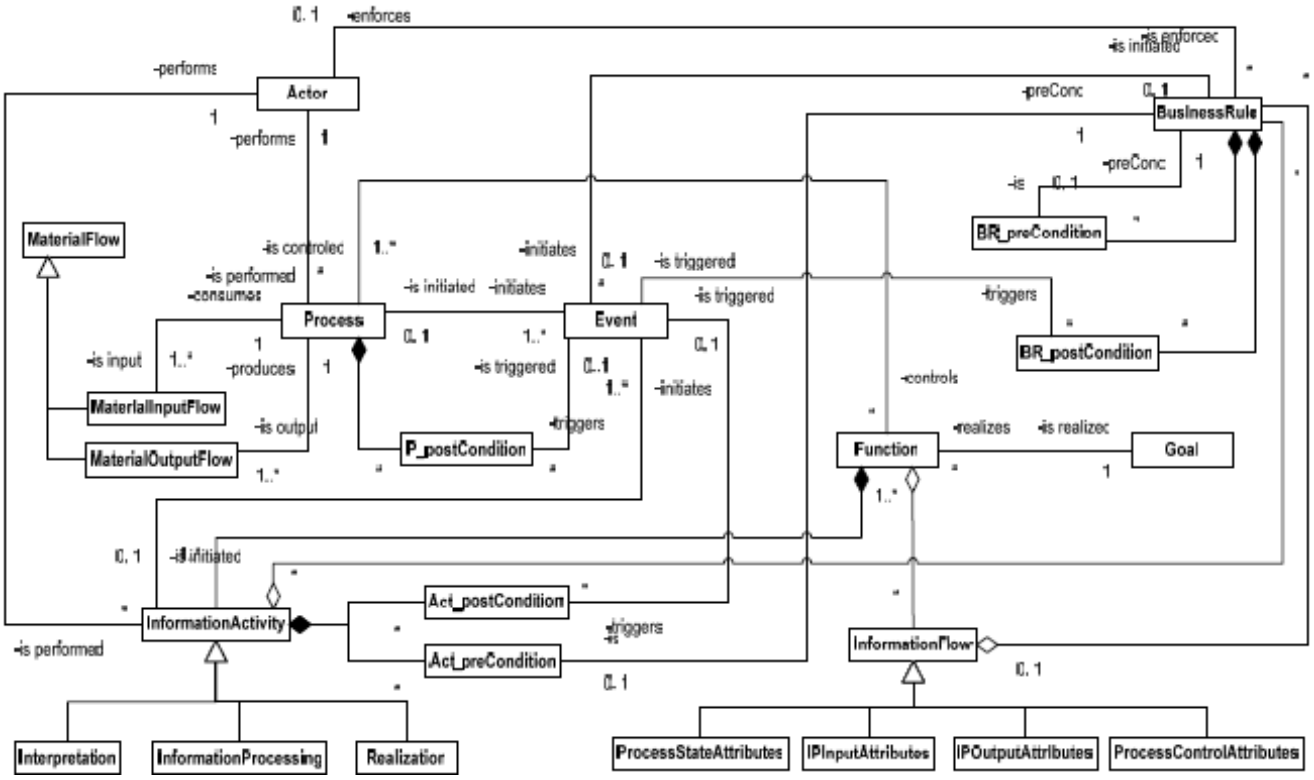
Paveikslėlyje pateikiamoje schemoje matomi veiklos žinių modelio pagrindai. T.y. čia vaizduojamas veiklos procesas, kaip žingsnių seka, kurias reikia įvykdyti, kad būtų pasiekti rezultatai (materialūs rezultatai). Taip pat yra veiklos proceso būsenų atributai ir valdymo atributai (įvesties, išvesties). Tačiau procesas bus efektyviai valdomas tik tuo atveju, jei bus panaudoti grįžtamieji ciklai [5, 6].

Norint veiklos žinių meta-modelį susieti su aplinka (parodyti kokius objektus ir kaip juos įtakoja srities aplinka) ir panaudoti IS kūrime, o taip pat kituose GC (gyvavimo ciklo) etapuose, kaip MD (angl. Model-Driven) proceso elementą, naudojama kiek kitokia schema. Šis veiklos žinių bazės meta-modelis turėtų būti naudojamas kaip „srities“ (angl. domain) žinių šaltinis visuose IS kūrimo etapuose (pav. 5).




Pav. 5Veiklos žinių meta-modelio koncepcinė schema [11].

Iš šio modelio matyti, kad aplinka įtakoja veiklos taisykles, tikslus (angl. objectives), kurie taip pat įtakoja veiklos taisykles ar vykdomas funkcijas. Atitinkamai reaguojant į aplinką gali būti sužadinami tam tikri įvykiai. Kitaip sakant aplinka sukelia tam tikrus sistemos įvykius (pav. 6).




Pav. 6 Veiklos žinių meta-modelio koncepcinė schema UML pagrindu [1].

Pagal šią schemą veiklos procesai yra sužadinami tam tikrų įvykių. Veiklos taisyklės sudarinėjamos atsižvelgiant į „aktorius“, kuriais gali būti asmenys, organizacijos ir pan. Siekiai – tikslai yra organizuojami kaip hierarchinė struktūra pagal veiklos svarbą. Veiklos taisyklės yra organizacijos sprendimų priėmimo mechanizmo dalis.

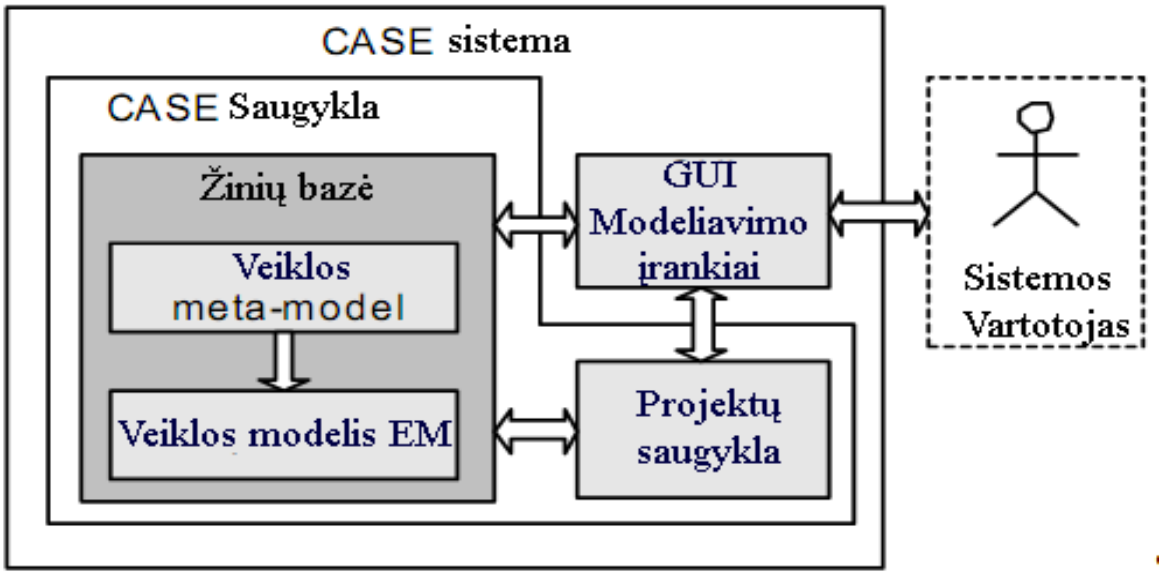
### 2.5.5. Veiklos žiniomis grindžiamos CASE sistemos architektūra

Norint pagerinti IS kūrimo bei palaikymo sąlygas, reikia veiklos žinių bazę integruoti į IS projektavimo sistemas. T.y. veiklos žinių bazė turi tapti CASE sistemų sudėtine dalimi.

Pradžioje reikia apsibrėžti, kad žinių bazė susideda iš dviejų pagrindinių sudėtinių dalių: veiklos meta-modelio (EMM) ir veiklos modelio (EM). Kitaip tariant veiklos meta-modelis yra pagrindinio (pirminio – angl. generic) lygio modelis. T.y. veiklos meta-modelis susideda iš dalinių (angl. partial) ir konkrečių (detalių – angl. particular) lygių modelių atitinkančių GERAM [14].

Manoma, kad veiklos žinių bazė turi tapti trečiuoju srities apibrėžimų, informacijos šaltinių projektuojant IS. Kai kiti - įprastiniai šaltiniai yra patys vartotojai bei analitikai. Veiklos meta-modelis (EMM) naujoje, papildytoje CASE sistemoje tūrėtų būti iš anksto apibrėžtų žinių (šablonų) šaltinis, kuris būtų naudojamas veiklos srities žinių perpratimo, išmokimo ir analizės procesų valdymui. Šis meta-modelis taip pat turi būti naudojamas veiklos modelio, tam tikrai konkrečiai veiklos sričiai, sudarymo proceso valdymui.

Tokios sistemos koncepcinė schema pateikiama sekančiame paveikslėlyje pav. 7:




Pav. 7 Veiklos žiniomis grindžiamos CASE sistemos architektūra Klasių modelis (CMM) [11]

Daroma prielaida, kad tokiose sistemose (žinių baze grindžiamose CASE sistemose) IS kūrimo gyvavimo ciklo visi etapai grindžiami veiklos žinių baze. T.y. CASE sistemos žinių bazė, IS analizės bei projektiniai modeliai (sprendiniai) suteikia galimybę patikrinti (angl. verificate) ir ratifikuoti (angl. validate) IS kūrimo gyvavimo ciklo etapus. Visa tai galima atlikti panaudojus tam tikrus algoritmus.

Sutikrinant sudarytą veiklos modelį (EM) su veiklos meta-modelių (EMM) galima patikrinti veiklos modelio teisingumą. CASE įrankio žinių bazė gali būti panaudota veiklos srities apibrėžimų (žinių) patikrai. Šie apibrėžimai buvo gauti ir apibrėžti analizės metu bei toliau panaudoti veiklos modelio sudaryme. Ši patikra suteikia analitikams galimybę aptikti ir atpažinti logines spragas atsiradusias veiklos procesų modeliuose, o taip pat formalias spragas, kurios galėjo atsirasti gautuose valdymo funkcijų modeliuose.

Loginės spragos – semantiniai netolydumai tarp probleminės srities modelių elementų (pvz. workflow modeliai). Šios spragos pasirodo, kai probleminė sritis nėra pilnai apibrėžta. Konkrečiai, kai gaunama ne pilna informacija.

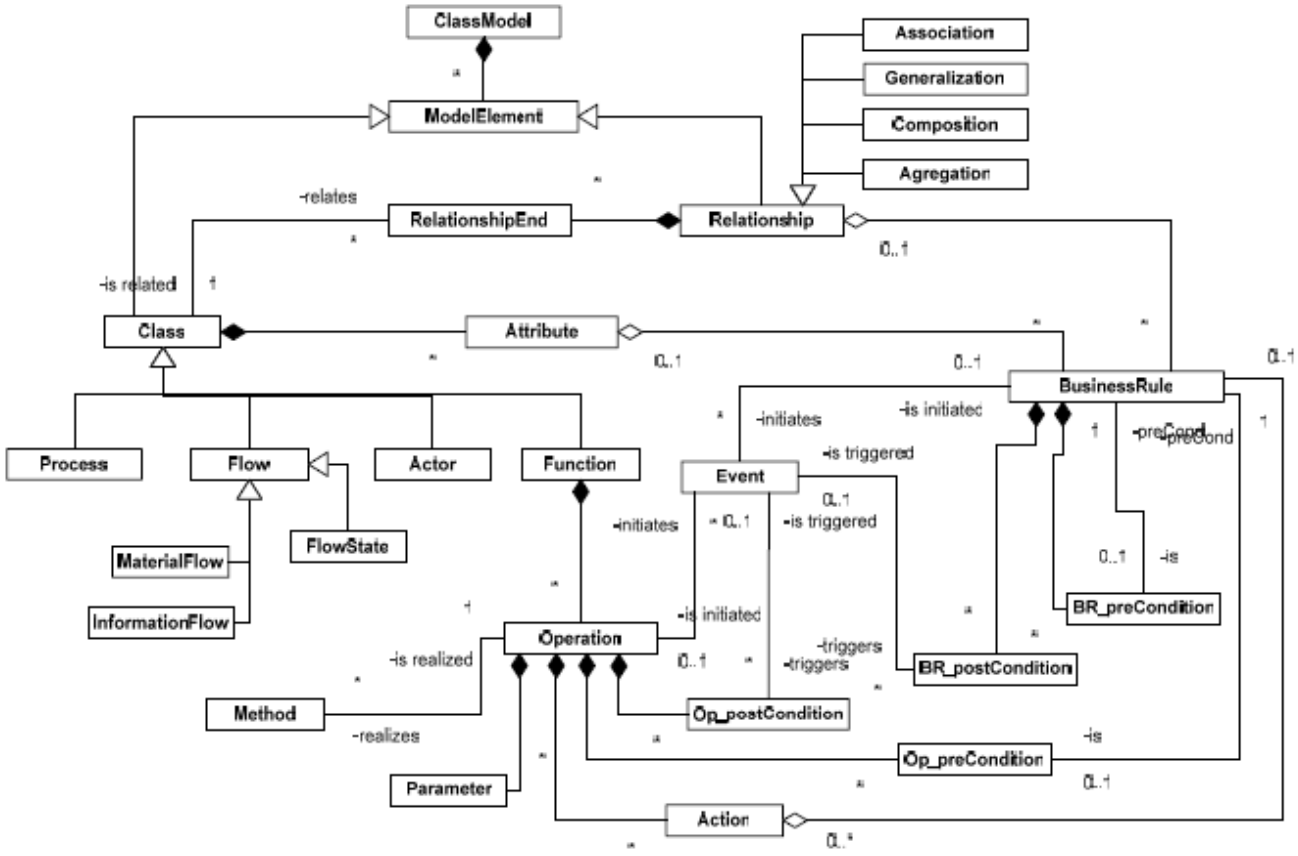
Formalios spragos – rodo tam tikrų privalomųjų apibrėžtų formalių komponentų trūkumą tam tikrame modelyje (pvz.: klasių modelyje). Šios spragos gali būti aptiktos modelių patikros (angl. verification) metu, lyginant modelio apibrėžimus (struktūrinius komponentus) su atitinkamu formaliu apibrėžimu, apibrėžtu veiklos meta-modelyje (EMM).

Veiklos meta-modelis (EMM) atitinka architektūrą, kuri pagerina veiklos modeliavimo proceso bei IS projektavimo proceso integravimą (susiejimą). Todėl CASE sistemos žinių bazė gali būti panaudota ne tik aptikti neatitikimus tarp veiklos procesų modelių ir IS projektinių modelių, bet ir automatiniam tam tikrų modelių generavimui pereinant tarp atitinkamu projektavimo lygių (PIM->CIM) (pvz. Klasių modelio automatiniam generavimui) [11].

### 2.5.6. Klasių modelis (CMM)

Atsižvelgiant į MDA – UML principus, Klasių modelis yra pagrindinis PIM modelis ir yra vienas iš svarbiausių projektuojant IS, nes vėliau gali būti transformuojamas į kitus PSM modelius. Klasių modelis taip pat tūrėtų būti gaunamas iš veiklos žinių bazės.

Šiuo metu situacija yra tokia, kad kuo labiau yra vystoma UML ir jos standartai, tuo vis labiau ji yra perkraunama nereikalingais elementais. T.y. atsiranda papildomų komponentų nereikalingų veiklos logikai [7].




Pav. 8 Klasių meta-modelis UML pagrindu. [1]

Pasiūlytame meta-modelyje pagrindiniai komponentai yra:

Klasių modelis – gali būti pačios klasės ar ryšiai tarp jų. Kiekvienas ryšys jungia mažiausiai dvi klases bei gali turėti apibrėžtas taisykles.

Klasėms yra suteikiami specialūs tipai [1]: veikla (process), srautas „*flow*“, aktorius arba funkcija. Tai skirta tam, kad susieti klasių modelį su veiklos žinių meta-modeliu (EMM).

Kintamos klasės (*Flow*) gali turėti būseną (*flow state*).

Klasių atributai, operacijos. Operacijas turi tik funkcijų tipų klasės.

Klasių operacijos – veiksmų sekos, turinčios parametrus bei metodus dėl to gali tūrėti pradžios bei pabaigos sąlygas (prie. ir post.).

Veiksmas – aprašomas viena veiklos taisykle (buisnes rule). Šios taisyklės taip pat turi pradžios bei pabaigos sąlygas (prie. ir post.).

Veiklos taisyklės, operacijos gali būti sužadintos tam tikrų įvykių, bei gali susižadinti pačios.

## 2.6. Veiklos žinių meta-modelio ir Klasių meta-modelio susiejimas

Pagal skyriuje 2.6 atliktą analizę, veiklos žinių meta-modelį galima tiesiogiai susieti su klasių meta-modeliu. Tai labai svarbu realizuojant automatinį klasių modelio generavimą. Norint juos susieti, reikia susieti atskirus meta-modelių komponentus. Tai pateikta lentelėje nr. 2.:

Lentelė 2Veiklos žinių meta-modelio susiejimas su klasių meta-modeliu. [1]

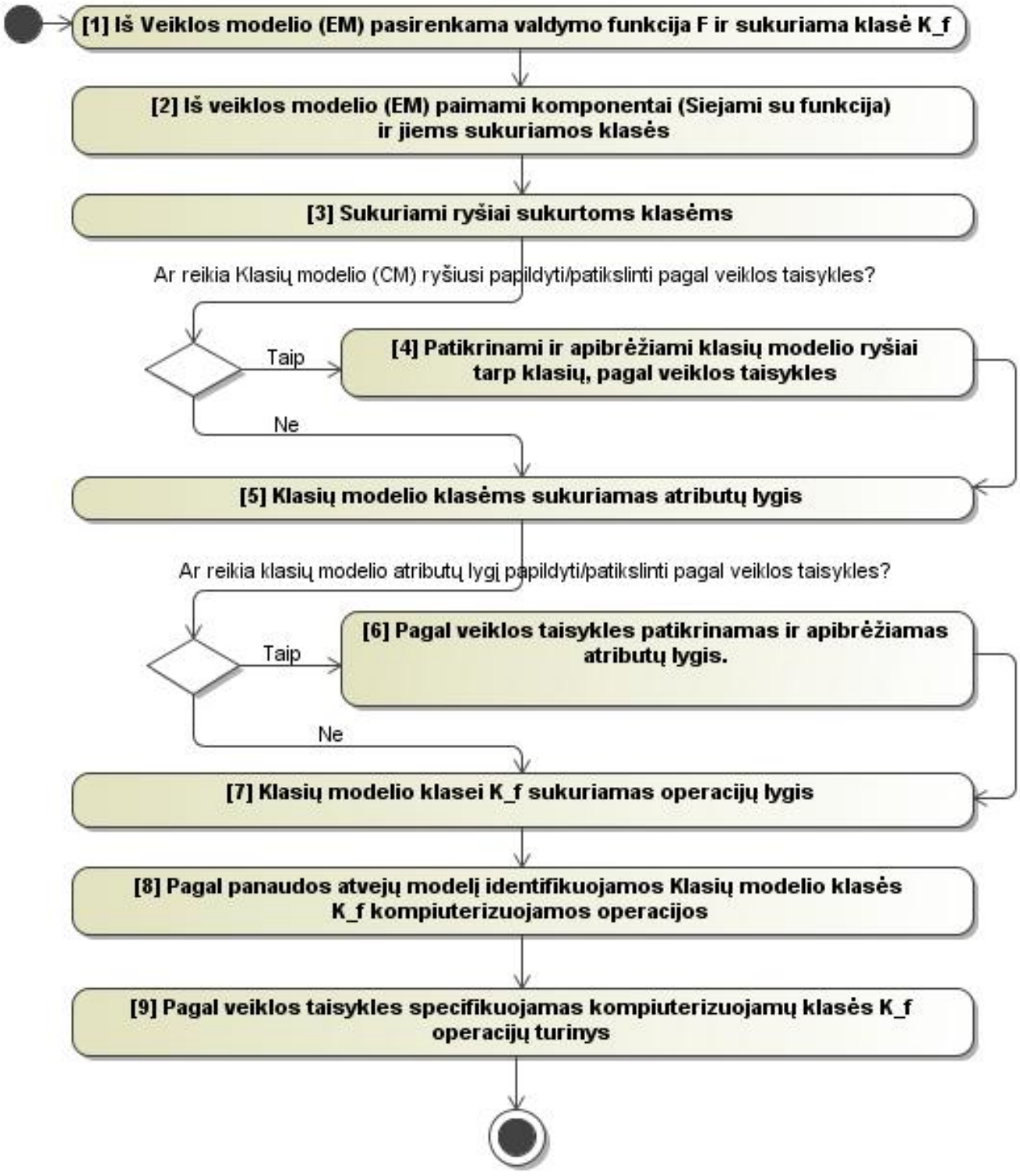
| EMM elementai | Susiejimas | Klasių meta-modelio elementai |
|---|---|---|
| <EMM.ModelElement> | φ1 | <CMM.ModelElement> |
| <EMM.Function> | φ2 | <CMM.Class>,<KMM.Function> |
| <EMM.Process> | φ3 | <CMM.Class>,<KMM.Process> |
| <EMM.MaterialFlow> | φ4 | <CMM.Class>,<KMM.Flow>,<KMM.FlowState> |
| <EMM.InformationFlow> | φ5 | <CMM.Class>,<KMM.Flow>,<KMM.FlowState> |
| <EMM.Actor> | φ6 | <CMM.Class>,<KMM.Actor> |
| <EMM.Event> | φ7 | <CMM.Event> |
| <EMM.InformationActivity> | φ8 | <CMM.Operation> |
| <EMM.Attribute> | φ9 | <CMM.Attribute> |
| Rel-ships among EMM elements | φ10 | <CMM.Relationship>,<KMM.RelationshipEnd> |
| <EMM.BusinessRule> | φ11 | <CMM.Action>, <KMM.BusinessRule> |
| <EMM.BusinessRule> | φ12 | <CMM.Relationship>,<KMM.RelationshipEnd>, <KMM.BusinessRule> |
| <EMM.BusinessRule> | φ13 | <CMM.Attribute>, <KMM.BusinessRule> |

Šioje lentelėje pateikta, su kokiais veiklos žinių meta-modelio elementais siejami klasių meta-modelio elementai. Vienas veiklos žinių meta-modelio elementas gali būti susiejamas su keliais klasių modelio lementais. Pvz.: susiejimas φ3 reiškia, kad EMM elementas „Process“ yra susietas su klasių meta-modelio elementais „Klass“ bei „Process“ (klasė yra pagrindinis elementas o veikla - tipas). Svarbu paminėti, kad ši susiejimų aibė yra pakankama sudarinėjant klasių modelį PIM lygyje.

## 2.7. Klasių modelio generavimo algoritmas

### 2.7.1. Klasių modelio generavimo algoritmo analizė

Pagal apsibrėžtą meta-modelių susiejimų aibę sudaromas algoritmas skirtas klasių modelio, grindžiamo veiklos žiniomis, saugomomis CASE sistemos žinių bazėje (Veiklos žinių modelyje) automatiniam generavimui [1].




Pav. 9 Klasių modelio generavimo algoritmas (PIM lygyje) [1].

Pagrindiniai algoritmo elementai yra EMM bei CMM siejančiųjų elementų aibė $\{\varphi 1..\varphi 13\}$ bei veiklos modelio saugomų žinių analizės taisyklių rinkiniai. Taisyklių panaudojimo sekos sudarymas yra atliekamas pačio algoritmo.

Šis algoritmas gali būti aprašytas keturiais pagrindiniais etapais (lentelė nr. 3): (1) klasių modelio (toliau KM) generavimas, (2) ryšių tarp klasių generavimas, (3) klasių atributų (atributų lygis) generavimas, (4) klasių operacijų lygio generavimas. Šie etapai dalinami į žingsnius ir žingsniai atliekami vienas po kito „krioklio“ principu (griežtai vienas paskui kitą), tačiau gali būti pertraukti (sistemos ar operatoriaus) ir tuo atveju procesas tampa iteraciniu.

Lentelė 3 Klasių modelio generavimo etapai ir žingsniai. [1]

| Etapai | Žingsniai | Sąsajos |
|---|---|---|
| Etapas 1. Probleminės srities veiklos objektų identifikavimas ir klasių modelio klasių generavimas. | Žingsnis [1] Iš Veiklos modelio (EM) pasirenkama valdymo funkcija F ir sukuriama klasė K_f. | $\varphi 1, \varphi 2$ |
| | Žingsnis [2] Iš veiklos modelio (EM) paimami komponentai (Siejami su funkcija) ir jiems sukuriamos klasės. | $\varphi 1, \varphi 3 - \varphi 6$ |
| Etapas 2. Ryšių tarp klasių identifikavimas ir generavimas. | Žingsnis [3] Sukuriami ryšiai sukurtoms klasėms. | $\varphi 10$ |
| | Žingsnis [4] Patikrinami ir apibrėžiami klasių modelio ryšiai tarp klasių, pagal veiklos taisykles. | $\varphi 12$ |
| Etapas 3. Klasių atributų lygio generavimas. | Žingsnis [5] Klasių modelio klasėms sukuriamas atributų lygis. | $\varphi 9$ |
| | Žingsnis [6] Pagal veiklos taisykles patikrinamas ir apibrėžiamas atributų lygis. | $\varphi 13$ |
| Etapas 4. Klasių operacijų lygio generavimas. | Žingsnis [7] Klasių modelio klasei K_f sukuriamas operacijų lygis. | $\varphi 7, \varphi 8$ |
| | Žingsnis [8] Pagal panaudos atvejų modelį identifikuojamos Klasių modelio klasės K_f kompiuterizuojamos operacijos | – |
| | Žingsnis [9] Pagal veiklos taisykles specifikuojamas kompiuterizuojamų klasės K_f operacijų turinys | $\varphi 7, \varphi 11$ |

Iš lentelės matyti, kad algoritmas sudarytas iš devynių žingsnių. Žingsniai 4, 6, 9 gali būti atliekami be eilės ir nepriklausomai nuo kitų algoritmo žingsnių. Ši galimybė demonstruojama žingsniuose (8, 9). Šios galimybės privalumas yra tame, kad vėliau, bet kuriuo metu bus galima patikrinti (validate) sudarytą klasių modelį.

Siekiamas algoritmas yra sudarytas iš 1, 2, 3, 5, 7, 8 žingsnių. Žingsniai 4, 6 ir 9 nebus vykdomi projektuojamo algoritmo. Tai yra dėl to, kad algoritmas turi tik sudaryti patį klasių modelį. Tokios realizacijos dėka pats algoritmas yra supaprastinamas, nes nereikės projektuoti modelio analizavimo priemonių (patikros ir papildymo pagal veiklos taisykles). Taip realizuojamas tik pats klasių modelio sudarymas pagal veiklos žinių bazę.

Klasių modelio patikros ir vėliau papildymo ar apribojimo funkcionalumas gali (ir turėtų) būti realizuotas atskiru algoritmu. Toks realizavimas neprieštarauja bendrai logikai, nes patikrą galima atlikti

bet kuriuo metu. Siekiant užtikrinti modelio ir viso IS projekto teisingumą, modelį (jau anksčiau sudarytą) reikia pastoviai tikrinti įvairiais IS projektavimo etapais (pvz. atlikti pakeitimai).

### 2.7.2. PIM lygio klasių modelio realizacija

Siekiamas darbo rezultatas – algoritmas, pagal veiklos žinių bazėje saugomą informaciją generuojantis klasių modelį PIM lygyje. Tačiau analizuojamas algoritmas nerealizuoja ryšių tarp srautų (informacinių, materialių srautų).

Srautai atitinka realius duomenis. Tad realizuojamas srautų (klasių objektų) atvaizdavimas atitinka duomenų modelį. Tokiame modelyje atitinkami duomenys (srautai) gali būti susiję. Tai galima išreikšti matematiškai. Pavyzdžiui: egzistuoja srautų aibė **Φ:**

$$\exists\Phi = \{\varphi \mid \varphi - srautai\} \quad (1)$$

Taip pat yra srautų atributų aibės:

$$\exists\Omega = \{\sigma \mid \sigma - srauto\_atributai\} \quad (2)$$

Bei srauto būsenų aibės:

$$\exists Z = \{\tau \mid \tau - srauto\_busenos\} \quad (3)$$

Srautų būsenų bei atributų aibės gali būti tuščios:

$$\exists\Omega = \text{ø} \quad (4)$$
$$\exists Z = \text{ø} \quad (5)$$
$$\forall\varphi \in \Phi\exists\Omega \quad (6)$$
$$\forall\varphi \in \Phi\exists Z \quad (7)$$

Galima daryti prielaidą, kad egzistuoja srautas $\varphi_1$ iš aibės **Φ** turintis ne tuščią atributų aibę $\Omega_1$ ir srautas $\varphi_2$ iš aibės **Φ** turintis ne tuščią būsenų aibę $\Omega_2$.

$$\forall\varphi_1 \in \Phi\,\exists\Omega_1 \quad (8)$$

$$\forall\varphi_2 \in \Phi\,\exists\Omega_2 \quad (9)$$

$$\Omega_1 \neq \text{ø} \text{ ir } \Omega_2 \neq \text{ø} \quad (10)$$

Tada jei egzistuoja bent vienas elementas iš aibės $\Omega_1$ esantis ir aibėje $\Omega_2$, tai tos aibės siejasi, o kartu siejasi ir elementai $\varphi_1$ ir $\varphi_2$.

$$\text{Jei } \Omega_1 \cap \Omega_2 \neq \text{ø} \quad (11)$$

$$\text{Arba } \Omega_1 \cup \Omega_2 = C \quad (12)$$

$$\text{T.y. } \exists\sigma \quad (13)$$

$$\sigma \in \Omega_1 ir\, \sigma \in \Omega_2 \quad (14)$$

$$\text{Tada } \Omega_1 \cap \Omega_2 \neq \text{ø} \quad (15)$$

$$\text{Bei } \Omega_1 \supset \Omega_2 \quad (16)$$

Iš (4) seka, kad srauto atributų aibė gali būti tuščia. Tačiau tokiu atveju, jei bent vieno srauto atributų aibė yra tuščia, aibių sankirta yra tuščia aibė ir srautai nebus susiję. Dėl to abiejų srautų atributų aibės turi būti ne tuščios aibės (10).

Net jei abi aibės nėra tuščios, to dar nepakanka. Reikia patikrinti, ar aibių sankirta nėra tuščia aibė (11). T.y. tikrinama ar yra bent vienas bendras elementas (14). Radus tokį elementą, reiškia, kad atributų aibės kertasi. Iš to seka, kad **Φ** aibės elementai $\varphi_1$ ir $\varphi_2$ irgi siejasi.

Tad atitinkami duomenų objektai gali sietis ir tai reikia atvaizduoti sugeneruotame klasių modelyje. Tik tokiu atveju bus gaunamas pilnas PIM lygio modelis. Tačiau planuojamoje realizacijoje šis funkcionalumas nebus realizuojamas. Todėl planuojamas sugeneruotas klasių modelis nebus pilnas PIM lygio modelis. Vartotojas, žinodamas atitinkamus susijusius elementus, galės neautomatizuotai juos susieti, taip modelį papildydamas iki pilno PIM lygio modelio.

## 2.8. Siekiamas sprendimas

Siekiama sudaryti modelį – priemonę, suderinamą su populiariom IS projektavimo sistemomis (pvz. MagicDraw UML), leidžiančią, sudarytos veiklos žinių bazės pagrindu, generuoti tam tikrus IS modelius (Klasių modelį - CM). Tai galima būtų atlikti, suprojektavus ir sudarius MagicDraw UML paketo papildymą – išorinę įterpiamą biblioteką (plug-ins) – įskiepį.

Kitas galimas sprendimas – pateikti pasiūlymus, kuriais galima pagerinti IS kūrimo procesą į jį integruojant veiklos modeliavimą veiklos žinių bazės pagalba. Tai gali būti atlikta pateikiant pasiūlymus ar papildymus ISK pasiūlytam modeliui, ar sudarant naują modelį. Darbas vykdomas orientuojantis ISK atliktais tyrimais ir modeliavimais.

Siekiant nustatyti sprendimo realizavimo galimybes atliekama tolimesnė (detalesnė) esamų priemonių ir technologijų analizė.

## 2.9. Esamos CASE IS projektavimo sistemos

Kaip jau buvo minėta, yra daugybė įvairių grafinių IS projektavimo bei verslo procesų modeliavimo įrankių. Taip pat yra daug veiklos modeliavimo metodikų bei siekių (CIMOSA, GERAM, IDEF suite, GRAI) [15] bei standartų (ISO 14258, ISO 15704, PSL, ISO TR 10314, CEN EN 12204, CEN 40003) ir veiklos modeliavimą leidžiančių priemonių [16]. Tačiau išlieka problema, kaip integruoti šiuos įrankius ar priemones siekiant automatizuoti IS projektavimo procesą.

Didelis įrankių ar priemonių pasirinkimas nedaro IS projektavimo optimalaus. Nes dėl gausaus metodikų ar priemonių kiekio projektavimo metu gaunamas didelis modelių, aprašančių probleminę sritį (IS projektas) rinkinys. Tačiau pats modelių gavimas nėra palengvinamas. T.y. procesas nėra automatizuotas arba tik dalinai automatizuotas. Net jei perėjimas nuo modelio prie modelio ir yra automatizuotas, jis yra tarp analogiškų modelių (iš darbų sekų modelio gaunamas procesų modelis). Taip nėra optimizuojamas IS projektavimo procesas.

Praktiškai visi IS projektavimo įrankiai yra komerciniai todėl daugumos IS projektavimo įrankių funkcionalumas nėra laisvai prieinamas. Taip pat yra mažai įrankių, kurių funkcionalumą galima laisvai keisti ar papildyti. Jei net suteikiama galimybė keisti funkcionalumą, nėra pateikiama atitinkamos dokumentacijos. Paprastai pateikiama informacija tik kaip reikia naudotis atitinkama IS projektavimo sistema.

„MagicDraw UML“ yra IS komercinė IS projektavimo sistema, kuri turi papildymo galimybę. Taip pat paketo autoriai pateikia išsamią dokumentaciją apie paketą, naudojimąsi juo bei kaip galima pakeisti ar papildyti jo funkcionalumą. Šis paketas taip pat pastoviai yra atnaujinamas ir kuriamos naujos jo versijos. Žinoma tai vienas iš populiariausių IS projektavimo paketų [18, 17].

Dėl šių priežasčių „MagicDraw UML“ geriausiai tinka kaip platforma, kurioje būtų galima realizuoti algoritmą. Pakete kuriamų modelių informacija (veiklos žinių bazė) gali būti naudojama kitų modelių (klasių modelio) automatiniam generavimui.

## 2.10. „MagicDraw UML“ išplėtimo galimybės

„MagicDraw UML“ yra vienas iš labiausiai pasaulyje vertinamų IS projektavimo įrankių. Įrankio populiarumą įtakoja ir tai, kad jis sukurtas ant „Java“ platformos. T.y. šis produktas yra nuo platformos nepriklausomas (Windows, MacOS, Linux, ...). Produkto populiarumui įtaką turi ir tai, kad įrankį galima derintini pagal savo poreikius (papildyti ar apibrėžti įrankio funkcionalumą). Dėl šios priežasties šis paketas geriausiai tinka tikslo realizavimui. Planuojama „MagicDraw UML“ paketui sudaryti įskiepį, kuris išplės paketo funkcionalumą.

„MagicDraw UML“ kūrėjų teigimu, vienintelis būdas pakeisti produkto funkcionalumą yra įskiepio (angl. plug-in) naudojimas. Pagrindinė įskiepių panaudojimo architektūros paskirtis yra papildyti produkto funkcionalumą. Taip yra galimybė panaikinti ar apriboti paketo esamą funkcionalumą. Tačiau ši galimybė yra labai ribota [17].

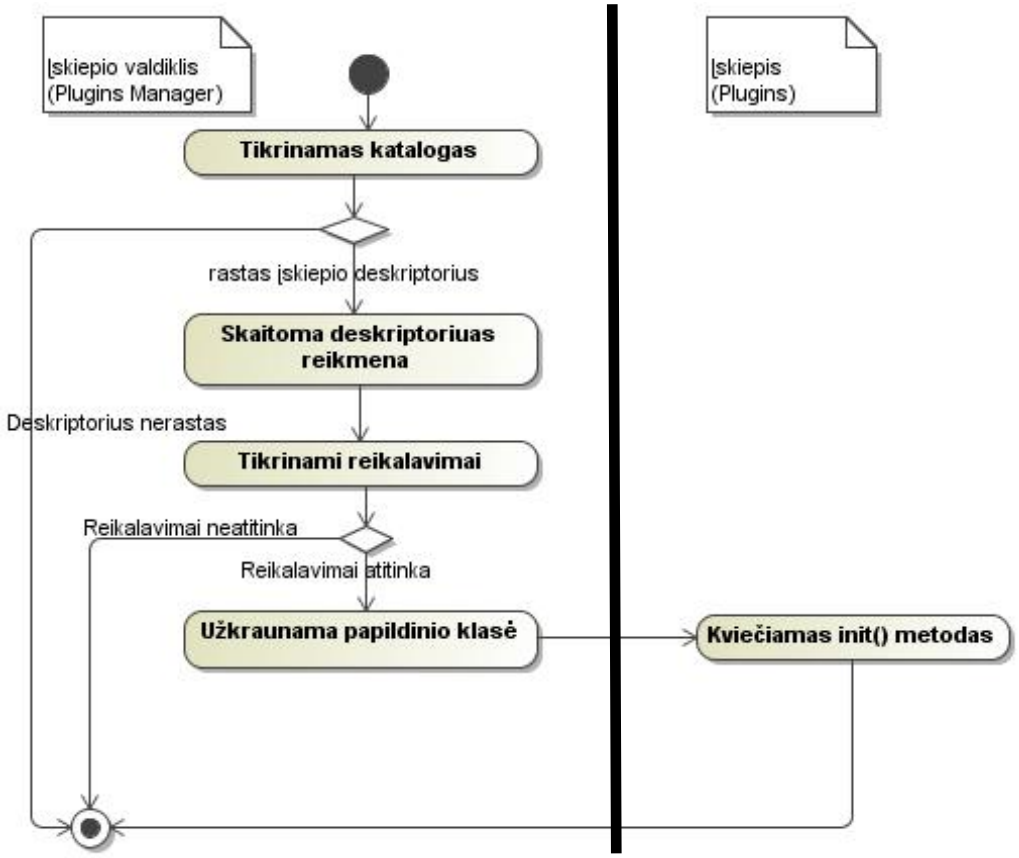
„MagicDraw UML“ paketo įskiepis sudarytas iš kelių privalomų išteklių (resursų) [17]:

- Katalogas (direktorija);
- Sukompiliuotos „Java“ bylos laikomos, sudarytose paketuose kaip „jar“ rinkmenos;
- Įskiepio aprašo rinkmeną (toliau įskiepio deskriptorius);
- Kitos papildomos įskiepio rinkmenos (pagal poreikį);

Paprastai, siekiant palengvinti vartotojų darbą, įskiepiai sukuria tam tikrus grafinės sąsajos elementus, kurie suteikia galimybę panaudoti įskiepio funkcionalumą. T.y. [17] autorių teigimu, įskiepiuose galima aprašyti (susikurti) tam tikrus grafinius komponentus, kurie palengvina įskiepio naudojimą. Apskritai tai nėra būtina todėl, kad įskiepis pats gali stebėti tam tikrus pakeitimus projekto atžvilgiu ir pats save aktyvuoti ir prisiderinti prie reikiamų, pageidaujamų pakeitimų [17].

## 2.11. „MagicDraw UML“ įskiepio įterpimas ir veikimas

Paketo įskiepiai gali būti naudojami, kaip standartinės darbinės aplinkos. T.y. kiekvieną kartą paleidžiant programą, iškart paleidžiamas įskiepio funkcionalumas, kuris suteikiamas „MagicDraw UML“ paketui. Tai yra realizuojama paketo pasileidimo metu įskiepių kataloge ieškant pakatalogių (Pav. 10).




Pav. 10 "MagicDraw" įskiepių užkrovimas paketo pasileidimo metu [17.: psl. 8].

Iš pateiktos schemos matyti, kad kiekvieno pasileidimo metu tikrinamas įskiepių katalogas ir ieškoma vidinių katalogų (sub-katalogų) ir juose esančių įskiepių deskriptorių rinkmenų. Jei kataloge yra deskriptoriaus rinkmena, ji yra nuskaitoma.

Tikrinama informacija apie įskiepio reikalavimus. Jei paketas atitinka deskriptoriaus rinkmenoje apibrėžtus reikalavimus, valdiklis (plugins manager) užkrauna reikiamas aprašytas klases. Yra tam tikros (būtinos) klasės, kurios privalomai turi būti užkrautos iš *com.nomagic.magicdraw.plugins.Plugin* klasės [17.: 8-9 psl.].

Užkraunant įskiepio klases yra dar gali būti vykdomi tam tikri metodai. Pvz. *init(),* kuris yra visada vykdomas klasės užkrovimo metu. Šis metodas gali atlikti įvairias operacijas įskaitant ir paruošiamąsias. Pvz. panaudojant veiksmų (operacijų) architektūrą, metodas gali sukurti ir pridėti tam tikrus vartotojo grafinės sąsajos (angl. GUI) komponentus. Taip pat metodas gali atlikti kitus veiksmus ir grąžinti tam tikrus rezultatus. *Init()* metodas vykdomas tik tuo atveju, jei nustatymuose nustatyta *isSupported()* reikšmė (*isSupported()* yra vykdomas prieš *init()* ir metodui grąžinama reikšmė *true,* jei įskiepis palaikomas arba *false,* jei įskiepis nepalaikomas).

Toliau „MagicDraw UML“ pakete įskiepiai naudojami pagal paskirtį. T.y. pakeičiamas ar papildomas paketo funkcionalumas. Svarbu paminėti, kad vienu metu gali būti naudojami keli įskiepiai [17]. Juos galima įtraukti ir atjungti bet kuriuo metu. Tai leidžia keisti ir tvarkyti įskiepių rinkinį.

Paketo išjungimo metu tvarkingai atjungiami visi vykdomi įskiepiai. Atjungimo metu vykdomas *close()* metodas, kuriame nurodytas funkcionalumas išjungimo metu.

## 2.12. „MagicDraw UML“ įskiepio panaudojimas veiklos žinių analizei ir modelių sudarymui

### 2.12.1. Reikalavimai algoritmo realizavimui

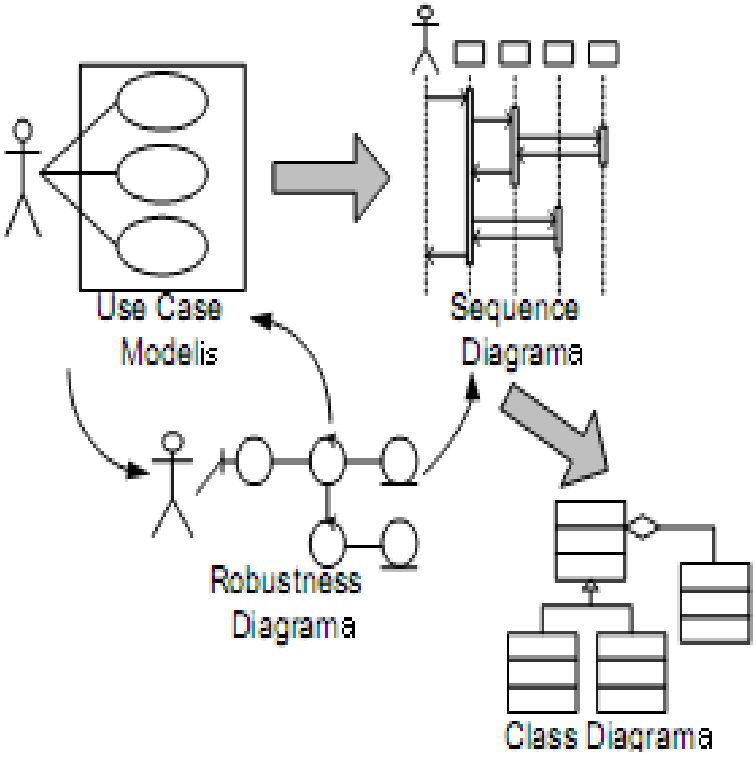
Algoritmas realizuoja automatinį klasės modelio generavimą pagal veiklos žinių bazę. Norint gauti išsamius modelius, reikia turėti išsamią veiklos žinių bazę. Pagal [19, 20, 21] klasių modelius (angl. Class model) galima generuoti turint tik panaudos atvejų modelį (Use case model). Tačiau norint gauti korektišką modelį su visais privalomais komponentais (atributais bei metodais), reikalingi ir kiti modeliai. Šiuo atveju reikalingas sekų modelis (angl. Sequence diagram) [19, 21].

Svarbu tai, kad paketas „MagicDraw UML“ (esamos versijos – 15.0) atitinka UML2 standartus ir palaiko reikiamus modelius ir diagramas [18].

Taip pat svarbu paminėti, kad sudarius panaudos atvejų modelį, pagal jį reikia atlikti realizaciją. Tam naudojama bendradarbiavimo (angl. „Collaboration“) diagrama, kuri parodo kaip panaudos atvejai realizuojami bei siejami su kitais objektais. Siekiant padidinti informatyvumą gali būti susiejamos dvi diagramos: bendradarbiavimo bei sekų diagrama. Tačiau neturint objektų aibės nėra įmanoma, tiksliau, yra labai sudėtinga pereiti nuo panaudos atvejų prie bendradarbiavimo diagramos. Iš to seka, kad objektus reikia aprašyti prieš sudarant bendradarbiavimo ir kitas diagramas. Tačiau šios problemos galima išvengti įtraukiant tarpinį žingsnį – sudarius „robustness“ diagramą [19, 21].

Panaudojant „robustness“ diagramą, objektų elgsena atvaizduojama trimačiam stačiakampyje (trijų matavimų erdvėje, kurią sudaro ašys: objektų sąsajos (angl. interface), esybių objektai bei kontrolės objektai). T.y. „robustness“ diagramą sudaro trys stereotipų objektai ir sąveika tarp jų.

Sąryšis tarp modelių, diagramų ir perėjimo seka matoma paveikslėlyje pav. 11.




Pav. 11 Perėjimas nuo Panaudos atvejų modelio prie Klasių diagramos [19]

Modelyje panaudojant „robustness“ diagramą, jos pagalba aptinkami ir identifikuojami objektų stereotipai bei objektų atributai. Taip pat identifikuojama ir sąveika tarp objektų. Taip pat gaunama aprašyta elgsena tarp skirtingų objektų [19, 21].

Visas šias diagramas palaiko „MagicDraw UML“ paketas. Taip pat yra galimybė panaudoti diagramų bei modelių informaciją kuriant naujus modelius. Iš to seka, kad galima realizuoti automatinį klasės modelio generavimą.

Išanalizavus ISK pasiūlytą sprendimą, galima generuoti klasių modelį visai nenaudojant anksčiau minėtų modelių ir diagramų.

Pasiūlytas algoritmas (pav. 9) automatizuotam modelio generavimui nenaudoja jokių „MagicDraw UML“ pakete esamų modelių ar diagramų informacijos. Visa reikiama informacija gaunama iš veiklos žinių bazės (priedas nr. 1), kuri yra realizuota duomenų bazėje. Tokių būdu užtikrinamas automatinis klasių modelio generavimas net neturint jokių veiklos modelių „MagicDraw UML“ pakete.

Taip pat pagerinamas modelių taisyklingumas. T.y. daroma prielaida, kad veiklos žinių bazėje yra saugomi korektiški modeliai. Tuo tarpu naudojant vidinius paketo modelius ar diagramas, kurios gali būti ne pilnos ar sudarytos nekorektiškai, atsiranda klaidos pernešimo galimybė. Taip pat tam tikri objektai gali būti nesusieti, tokiu būdu bus prarandama informacija.

Tokiam algoritmo realizavimui vienas iš svarbiausių reikalavimų yra „MagicDraw UML“ paketo susiejimas su išorinėmis duomenų bazėmis.

Norint nustatyti algoritmo realizavimo galimybes, buvo atlikta analizė, kurios tikslas nustatyti įvairių duomenų bazių panaudojimo galimybes. Tai svarbu, nes paketo įskiepis bus kuriamas „Java“ programavimo kalba. Nesant galimybei įskiepio susieti su pasirinkta duomenų baze, algoritmas neveiks, nes neturės duomenų klasių modelio generavimui.

Šaltinių [17, 18, 22] analizė parodė, kad įskiepis skirtas „MagicDraw UML“ paketui ir parašytas „Java“ programavimo kalba gali būti susietas su daug įvairių skirtingų duomenų bazių („MySQL“, „Microsoft SQL server“, ir daug kitų). Norint naudoti skirtingas duomenų bazes, reikia įdiegti atitinkamas duomenų bazių tvarkykles (atitinkamas bibliotekas). Šios bibliotekos gali būti iš kart įtrauktos į realizuojamą įskiepį.

Informacija leidžia realizuoti algoritmo funkcionalumą. Didelis duomenų bazių tipų pasirinkimas leidžia algoritmą naudoti skirtingose sistemose. T.y. algoritmo kodas nesikeičia („Java“ kalba parašytos programos veikia ant skirtingų platformų), tik keičiama ar parenkama kita duomenų bazės palaikymo biblioteka.

Veiklos žinių saugojimas išorinėse duomenų bazėse, o ne „MagicDraw UML“ paketo projektuose suteikia galimybę paprasčiau šias žinias panaudoti kituose projektuose. Taip sistemos tampa lygtinėmis. Pakartotinis žinių panaudojimas pagerina kitų projektų vykdymo procesą.

### 2.12.2. „MagicDraw UML“ funkcionalumo panaudojimas

Pagal pateiktą paketo dokumentaciją, paketo funkcionalumą galima papildyti įterpiant savo sudarytus ar kitus įskiepius (plug-ins). Sudarant įskiepius galima ne tik sukurti naujas funkcijas ar objektus (sudaryti naujus modelius, papildyti meta-modelius ir pan.), bet ir panaudoti turimus objektus (meta-modelių informaciją, projektų informaciją ir pan.)

Pakete pagrindinė informacijos saugykla yra „projektas“. Jame saugoma visa pagrindinė informacija: pagrindiniai paketai (t.y. modeliai) bei modelių diagramos. Visa informacija yra išreiškiama elementais. Elementai talpinami (išdėstomi) pagal medžio struktūrą (yra šakninis elementas, aukštesnio lygio elementas – „tėvas“, žemesnio lygio elementas – „vaikas“). Visas modelis yra apibrėžtas šakniniu elementu. Per elementų medį galima keliauti (analizuoti) įvairiomis kryptimis (tiek gilyn, tiek aukštyn). Kiekvienas elementas turi (apibrėžia) tam tikrą informaciją. T.y. elementas yra kaip konteineris [17].

Medžio tipo duomenų struktūra, kurioje laikomi visi elementai (visa projekto informacija) labai palengvina analizę. Galima realizuoti įvairias operacijas su elementais, kurios realizuojamos adresais. T.y.: operacija: adresas elemento, elemento „tėvo“ ar elemento „vaiko“.

Pakete yra galimybė atlikti operacijas su modeliais ir elementais (lentelė 4.):

Lentelė 4 „MagicDraw UML“ galimos operacijos su modeliais, elementais, diagramomis [17.: psl. 39-53]

| **Nr.** | **Operacija** | **Aprašas** |
|---|---|---|
| 1 | Sukurti modelio elementą | Sukuriamas modelio elementas – modelio esybė, kurioje laikoma atitinkama informacija. |
| 2 | Keisti modelio elementą | Keičiama modelio elemento informacija |
| 3 | Sukurti naują elementą ar priskirti jį kitam elementui | Sukuriamas naujas elementas nurodytam „tėvo“ elementui. Taip pat galima esamam elementui pakeisti „tėvo“ elementą. T.y. priskirti jį kitam elementui. |
| 4 | Pašalinti modelio elementą | Panaikinamas (ištrinamas) nurodytas elementas. |
| 5 | Sukurti diagramą | Sukuriama diagrama, kurioje aprašomi objektai (elementai). |
| 6 | Sukurti ryšio objektą | Ryšiai tarp objektų realizuojami objektais. Ryšio objektas aprašo kaip ir kokius objektus sieja. |
| 7 | Objektų informacijos peržiūra | Paketas suteikia galimybę peržiūrėti elementų, objektų informaciją. |

Lentelėje pateikiamos pagrindinės operacijos. Tačiau yra ir daugybė kitų operacijų, kurias galima realizuoti sudarant paketo įskiepį.

Kadangi projektą sudaro modeliai ir diagramos, o jų informacija saugoma elementų medyje, tai realizuojant algoritmą, reikia atlikti atitinkamuose elementuose saugomos informacijos analizę ir pagal atitinkamus analizės rezultatus sudarinėti naujus elementus. Šios operacijos gali būti įvykdomos panaudojus esamą paketo funkcionalumą (operacijas).

## 2.13. Analizės išvados

1. Tyrimo objekto bei vartotojų analizė parodė, kad siekiant optimizuoti IS projektavimo procesą, reikia automatizuoti perėjimus tarp CIM->PIM->PSM->Kodas lygmenų. Vienas iš žingsnių – automatizuotas klasių modelio gavimas pagal veiklos žinių bazėje saugomus įrašus apie veiklos sritį. Taip pat atskleidžiamas problemos aktualumas.

2. Esamos situacijos analizė parodė, kad šiuo metu yra daugybė priemonių, įrankių, leidžiančių projektuoti, aprašinėti veiklos sritį, sudarinėti veiklos modelius. Taip pat yra priemonės IS projektavimui. Tačiau šios priemonės nėra integruojamos tarpusavyje, tad tam tikros veiklos tiesiog atsisakoma, taip taupant IS kūrimui skirtą laiką. Tai yra dėl to, kad kūrimo procesas nėra pilnai automatizuotas ir siekiama išvengti perteklinio darbo. Tad dažniausiai atsisakoma detalios veiklos žinių bazės sudarymo. Klasių modelio automatizuotas sudarymas veiklos žinių pagrindu ar pilnas CIM->PIM->PSM->Kodas perėjimas nėra realizuoti.

3. Klasių modelio sudarymo pagal veiklos žinias galimybių atlikta analizė parodė, kad yra pasiūlytos kelios galimybės, kaip pagal veiklos žinių bazėje (EM) saugomą informaciją galima automatiškai generuoti IS klasių modelį (CM), naudojant papildytą klasių meta-modelį (CMM).

4. Atlikta ISK vykdomų tyrimų analizė parodė, kad pasiūlytas algoritmas yra pagrįstas ir gali būti naudojamas, kaip pagrindas. Tad klasių modelio generavimui skirtas metodas bus projektuojamas ir realizuojamas ISK pasiūlyto metodo pagrindu.

5. Atlikus „MagicDraw UML“ paketo funkcionalumo išplėtimo galimybių analizę, nustatyta, kad paketas suteikia galimybę papildyti jo funkcionalumą, naudojant įskiepius (angl. plug-ins) bei profiliavimo mechanizmą.

6. Analizės metu nustatyta, kad sudarytas klasių modelis bus veiklos modelio (veiklos objektų modelio) atitikmuo PIM lygmenyje. Tai taip pat nebus pilnas PIM lygio modelis, nes nebus sudaromi ryšiai tarp srautų (srauto tipo klasių – „Flow“).

# 3. KLASIŲ MODELIO SUDARYMO METODO PROJEKTAVIMAS

## 3.1.Projektavimo uždaviniai

Remiantis atliktos analizės apibendrinimais užsibrėžiami pagrindiniai projektavimo uždaviniai:

1. „MagicDraw UML“ paketo išplėsto funkcionalumo projektavimas;
2. Veiklos žinių saugyklos panaudojimo bei realizavimo projektavimas;
3. Klasių meta-modelio išplėtimo projektavimas;
4. „MagicDraw UML“ profilio panaudojimo projektavimas;
5. Algoritmo praktinio panaudojimo tyrimas (analizė ir projektavimas);
6. Vartotojo sąsajos projektavimas.

Pagal užsibrėžtus uždavinius atliekami tolimesni projektavimo darbai.

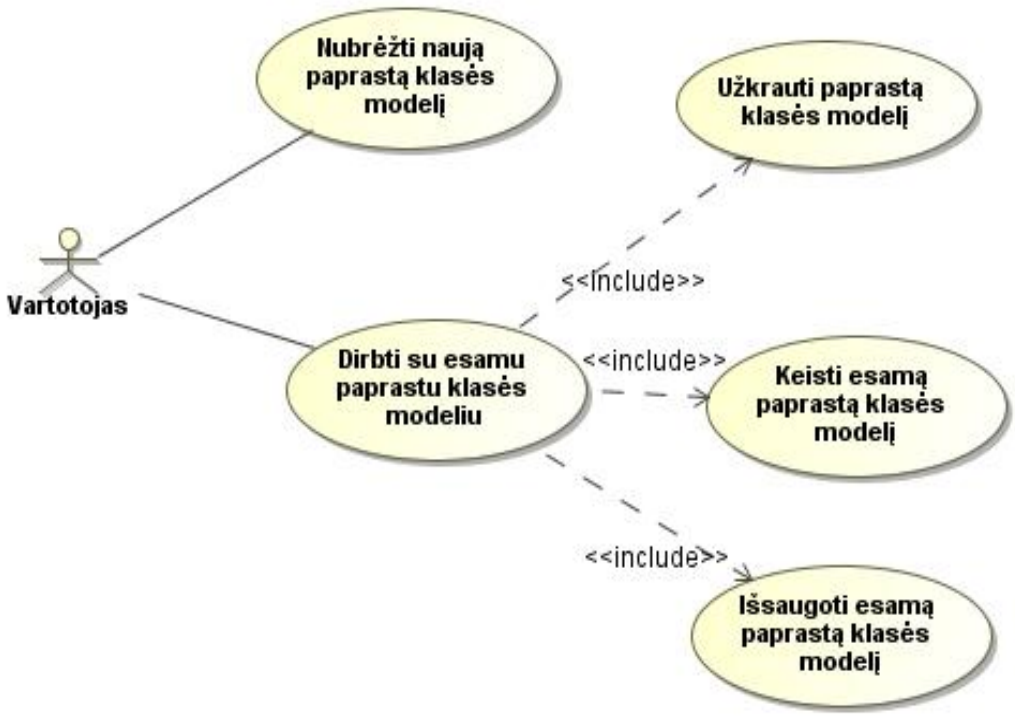
## 3.2.„MagicDraw UML“ paketo funkcionalumo projektavimas

Paketo funkcionalumo projektavimas susideda iš dviejų etapų. Pirmasis – funkcinių reikalavimų apibrėžimas. Šį etapą galima įvardyti, kaip funkcionalumo projektavimą.

Antrasis etapas yra funkcionalumo detalizavimas – paketo komponentų projektavimas. Šiame etape aprašomi pagrindiniai „MagicDraw UML“ naudojami komponentai ir projektuojami funkcionalumą papildantys komponentai.

### 3.2.1. Funkcionalumo projektavimas

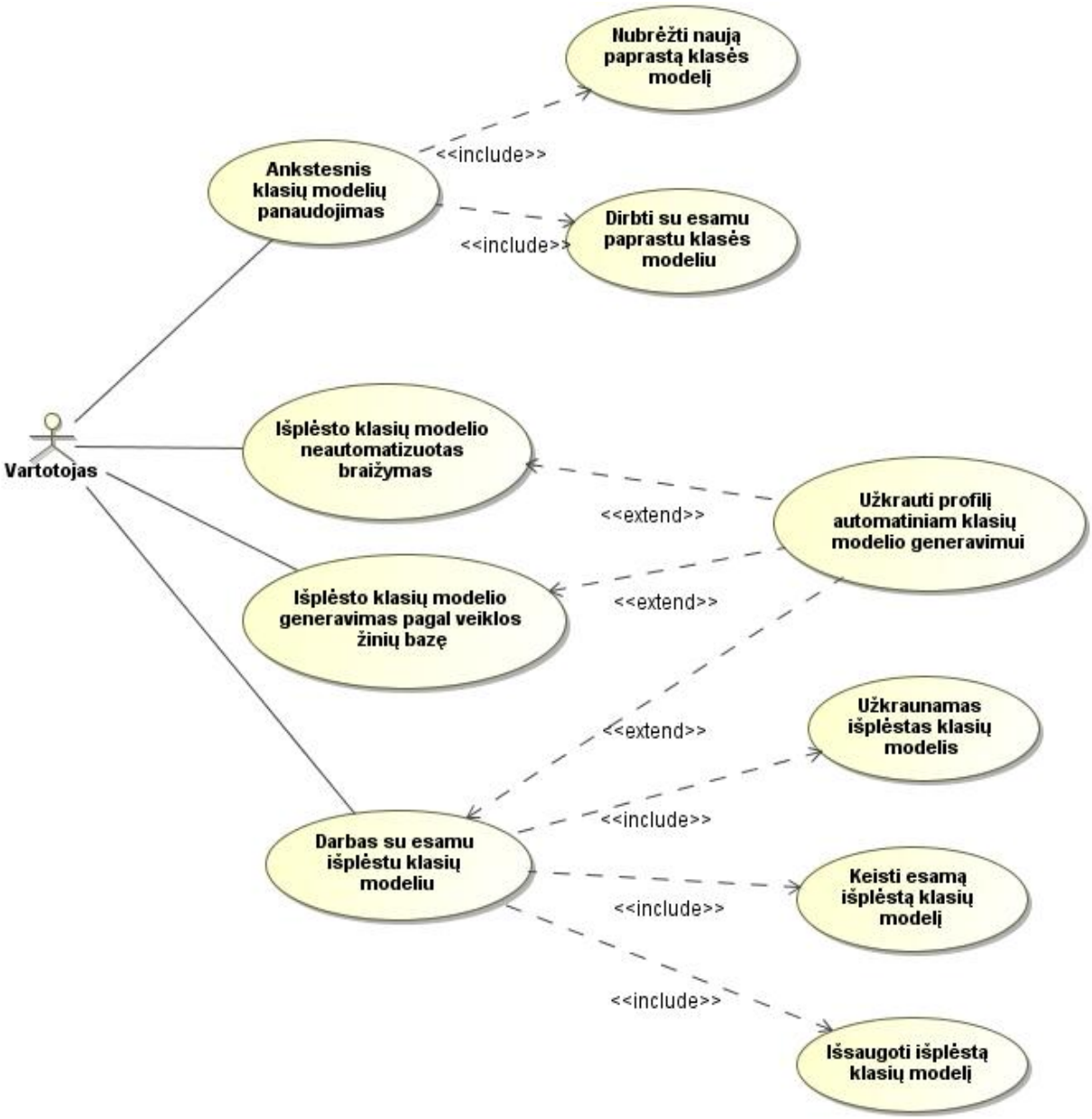
Darbo metu bus dirbama su „MagicDraw UML“ 15 versija. Kaip ir ankstesnėse versijos, yra standartinės operacijos vartotojams, skirtos klasių modelio projektavimui bei peržiūrai. Jas galima atvaizduoti sekančiu panaudos atvejų modeliu pagalba (pav. 12). Šie modeliai atitinka funkcinius sistemos reikalavimus.




Pav. 12 Starnadartiniai "MagicDraw UML" panaudos atvejai dirbant su klasių modeliu

Iš šio modelio gerai matyti, kad vartotojui suteikiama galimybė sudarinėti naujus standartinius klasių modelius bei peržiūrėti ar keisti esamus modelius. Tai atliekama užkraunant atitinkamus projektus, juos keičiant ar išsaugant iš naujo.

Tačiau šis funkcionalumas neturi jokios automatizacijos, kuri galėtų pagerinti darbą su paketu. Tuo tikslu projektuojamas funkcionalumas (pateiktas pav. 13) išplečiantis esamą paketo funkcionalumą. T.y. suteikiamos naujos galimybės neprarandat turėtų operacijų.



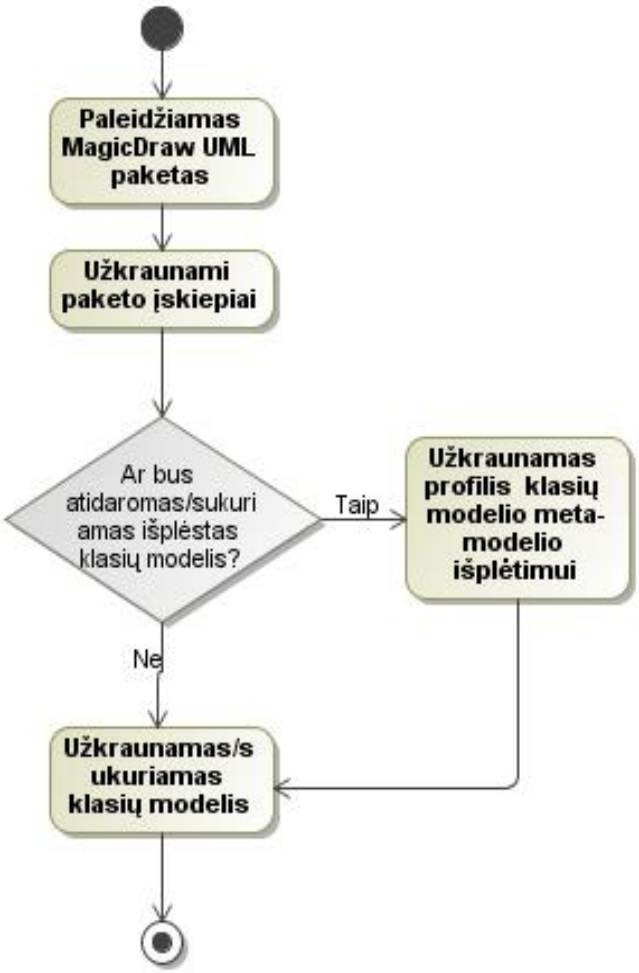

Pav. 13 Išplėsto "MagicDraw UML" paketo panaudos atvejų modelis

Nors išplėstas ir paprastas klasių modeliai skiriasi tarpusavyje komponentais, paketas vis vien turės galimybę dirbti su abiejų tipų klasių modeliais. Funkcionalumas nėra prarandamas, tik išplečiamas naujomis galimybėmis.

Paketo išplėtimas leis generuoti klasių modelį pagal veiklos žinių bazę. Tai turi ženkliai optimizuoti visą projektavimo procesą. Taip pat išlieka galimybė neautomatizuotu būdu sudarinėti išplėstą kla-

sių modelį. Tokiu atveju algoritmą optimizuotų kitas algoritmas – skirtas sudaryto išplėsto klasių modelio patikslinimui ir apribojimui. Šiame darbe jis nėra projektuojamas ar realizuojamas.

Darbas su esamu išplėstu klasių modeliu (sudarytu automatizuotu ar neautomatizuotu būdu) yra analogiškas darbui su paprastu klasių modeliu. T.y. vartotojas taip pat gali užsikrauti esamą projektą, jį modifikuoti bei išsaugoti pakeitimus. Tai pateikta sekančiame modelyje (pav. 14)

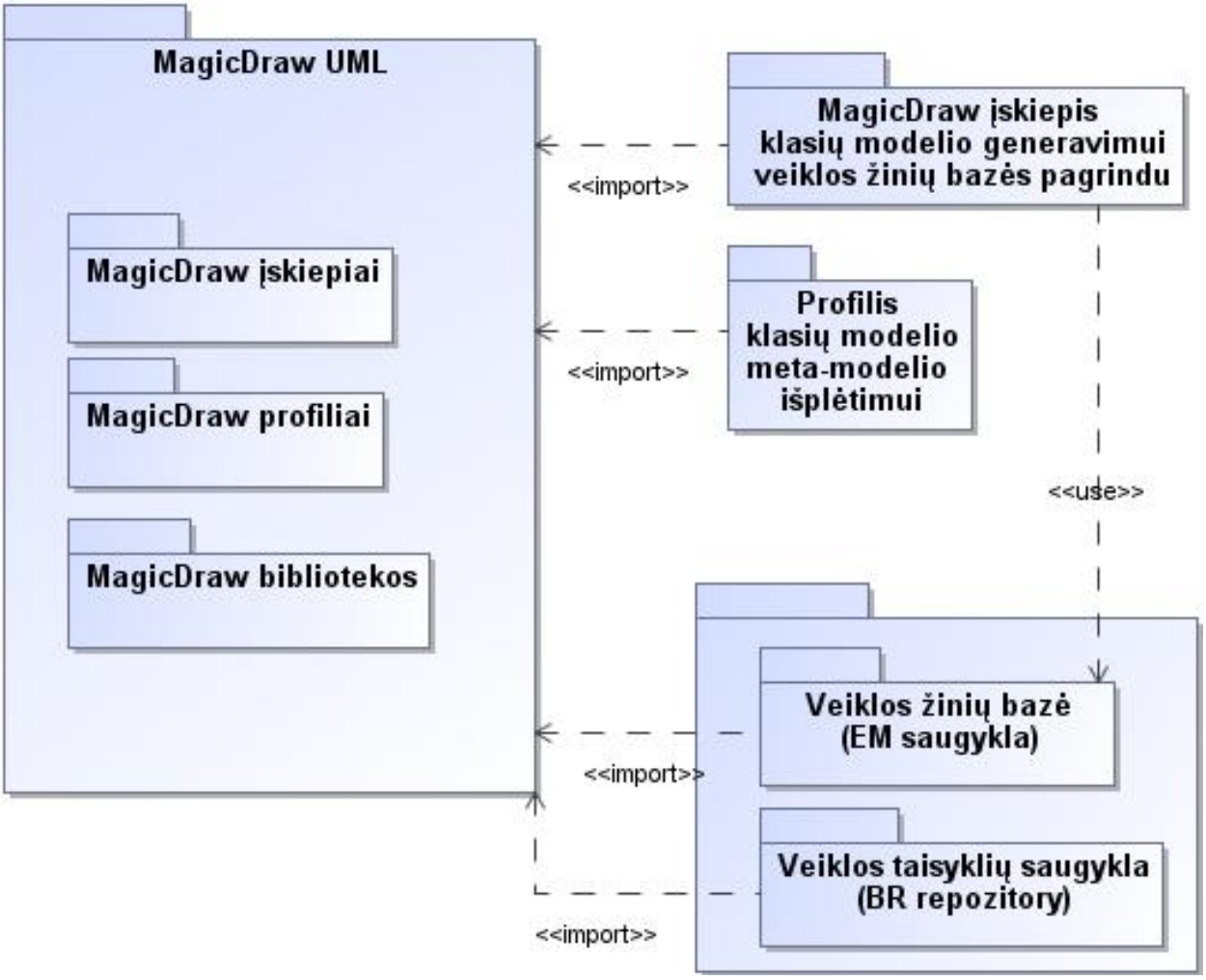



Pav. 14 Modelių atidarymo/sukūrimo veiklos diagrama.

Iš pateiktos veiklos diagramos gerai matyti skirtumas tarp funkcionalumu dirbant su išplėstu ir paprastu klasių modeliais. Dirbant su išplėstų klasių modelių, prieš užkraunant darbinę aplinką (klasių modelio modeliavimo aplinką – modelį), reikia užkrauti darbui su išplėstu klasių modeliu „MagicDraw UML“ paketui sudarytą profilį (anlg. profile). Šiuo profiliu yra išplečiamas esamas klasių meta-modelis. Profilis suteikia naują modeliavimo aplinką (naują modelį) – išplėstą klasių modelį, kuriame klasės yra stereotipizuojamos atitinkamų tipų klasėmis.

### 3.2.2. „MagicDraw UML“ komponentų projektavimas

Planuojama, kad realizuojamas algoritmas neapribos esamo „MagicDraw UML“ paketo funkcionalumo. T.y. išliks standartinis paketo teikiamas funkcionalumas, kuris, reikalui esant, bus išplečiamas įskiepio (realizuojamas klasių modelio generavimo algoritmas) bei profilio (išplečiamas esamas klasių modelio meta-modelis) pagalba. Komponentų panaudojimas pateikiamas sekančioje diagramoje pav. 15.




Pav. 15 "MagicDraw UML" paketui projektuojami komponentai

Iš pateiktos komponentų diagramos gerai matyti, kad visi papildomi komponentai yra įterpiami į „MagicDraw UML“. Papildomi komponentai gali būti įterpiami bei naudojami pasirinktinai. Jei norima tik peržiūrėti ar neautomatizuotu būdu sudarinėti išplėstą klasių modeli, užtenka tik profilio. Jei planuojama automatizuotu būdu (generuoti) sudarinėti klasių modelį, be profilio reikalingas įskiepis ir veiklos žinių bazė.

Įskiepio detalesnis aprašymas pateikiamas lentelė 5.

Lentelė 5 Komponento - „MagicDraw UML įskiepis“ aprašymas

| Komponetas | „MagicDraw UML“ įskiepis |
|---|---|
| Jį naudoja/iškviečia | Vartotojas (pasirinktinai) iš „MagicDraw UML“ paketo |
| Komponentui perduodamas srautas | • Generavimo ir susijungimo su DB nustatymai;<br>• Modelio (projekto) informacija; |
| Vidiniai srautai | Komunikavimas su veiklos žinių baze; |
| Grąžinamas srautas (Iš komponento) | • Sugeneruotas klasių modelis (rezultatai);<br>• Pranešimai; |
| Komponento operacijos | • Vartotojo sąsajos (nustatymams pateikimas);<br>• DB operacijos (susijungimas, žinių išgavimas);<br>• Komunikavimas su „MagicDraw UML“;<br>• Klasių modelio sudarymas; |
| Komponento ribojimai | • Privalo būti sudarytas projektas, modelis prieš įtraukiant profilį;<br>• Privalo būti veiklos žinių bazė; |

Lentelėje pateikiamos pagrindinės komponento charakteristikos. Pateikiama informacija, kas naudoja elementą. Taip pat nurodomi srautai (perduodami komponentui, grįžtamieji). Vidiniai srautai, parodo vidines operacijas. Tai svarbu, nes įskiepis gali bendrauti su veiklos žinių baze.

Svarbi informacija – įskiepio ribojimai. Čia pateikiama, privalomos sąlygos, reikalingos norint panaudoti įskiepį. Analogiškai aprašomas ir kitas komponentas – veiklos žinių bazė (lentelė 6).

Lentelė 6 Komponento - „Veiklos žinių bazė“ aprašymas

| Komponetas | Veiklos žinių bazė |
|---|---|
| Jį naudoja/iškviečia | „MagicDraw UML“ paketo įskiepis |
| Komponentui perduodamas srautas | • Užklausos (SQL);<br>• Komandos (SQL); |
| Vidiniai srautai | - |
| Grąžinamas srautas | • Atsakymas |
| Komponento operacijos | • Užklausų/komandų vykdymas;<br>• Rezultatų pateikimas; |
| Komponento ribojimai | - |

Priešingai nei įskiepis, veiklos žinių bazė neturi nurodytų ribojimų. Šiuos ribojimus lemia pats DB serveris ir jo galimybės. Taip pat serveris įtakoja vidinius srautus. T.y. šios charakteristikos nėra aktualios ir nėra nagrinėjamos projektuojant. Yra žinoma, kad esamo funkcionalumo pakanka prototipo realizacijai.

Atitinkamai lentelėje 7. aprašomas ir paketo profilis.

Lentelė 7 Komponento - „MagicDraw UML profilis, meta-modelio išplėtimui“ aprašymas

| Komponetas | MagicDraw UML profilis, meta-modelio išplėtimui |
|---|---|
| Jį naudoja/iškviečia | „MagicDraw UML“ paketas |
| Komponentui perduodamas srautas | - |
| Vidiniai srautai | - |
| Grąžinamas srautas | • Elementų išplėtimo informacija;<br>• Stereotipų aibė ir panaudojimo informacija |
| Komponento operacijos | - |
| Komponento ribojimai | Privalo būti aprašyti visi veiklos modelio elementai |

Profilį naudoja „MagicDraw UML“ paketas. Komponentui nėra perduodama jokia specifinė informacija. Profilis taip pat nevykdo jokių vidinių operacijų, dėl to nėra jokių vidinių srautų. Svarbios paketo charakteristikos yra grąžinamas srautas (nurodoma, kas ir kaip išplečiama) bei ribojimai. Grąžinami srautai atskleidžia profilio panaudojimo paskirtį.

Svarbu paminėti, kad veiklos žinių bazėje saugomos veiklos žinios. Taip pat pakete gali būti naudojama ir veiklos taisyklių saugykla. Ji gali būti naudojama sudarytiems išplėstiniams klasių modeliams patikslinti ar apriboti. Tačiau šiame darbe ši galimybė nėra realizuojama.

Saugyklos projektuojamos kaip duomenų bazės. Jos gali būti realizuotos tiek vietiniame kompiuteryje, tiek ir nutolusiame serveryje. Todėl saugyklos diagramoje išskirtos į atskirą paketą.

### 3.3. EM (veiklos žinių bazės) saugyklos modelis

Šiame projektavimo etape yra analizuojama esama saugykla (esamas saugyklos modelis). Analizės tikslas – identifikuoti saugyklos trūkumus.

Atliktos analizės pagrindu yra projektuojami saugyklos pakeitimai ir papildymai, siekiant optimizuoti VŽ saugyklą.

#### 3.3.1. Esamas EM (veiklos žinių bazės) saugyklos modelis

Srities veikla gali būti apibrėžta veiklos modeliais. Atitinkamai modeliai gali būti saugomi veiklos žinių bazėje (saugykloje), veiklos žinias išreiškiant atitinkamomis formalizuotomis išraiškomis.

Šiame darbe apsibrėžiama, kad srities veiklai aprašyti vienintelis veiklos žinių šaltinis bus veiklos žinių bazė (saugykloje saugomos žinios kaip DB įrašai). Taip uždavinys tampa konkretesnis ir problema tampa aiškesnė. T.y. uždavinį galima suformuluoti taip: realizuojamas algoritmas generuos išplėstą klasių modelį pagal veiklos žinių bazėje (saugykloje) saugomas veiklos žinias (formalizuotas išraiškas).

ISK pasiūlyta veiklos žinių bazė realizuojama pasirenkamoje duomenų bazėje. T.y. šiuo metu yra pasiūlyta veiklos modelio saugyklos loginė duomenų bazės schema (priedas nr. 1). Saugyklos schema yra pagrįsta pagal OMG pateiktas dokumentacijas. Pateiktoji saugykla gali būti realizuota įvairiose duomenų bazėse. Bazė gali būti tiek esamame, tiek nutolusiame kompiuteryje.

Norint susieti veiklos žinių saugyklą (duomenų bazę) su „MagicDraw UML“ paketu, pagrindinis uždavinys yra susieti paketo klasių modelio ir klasių meta-modelio komponentus su atitinkamais veiklos žinių bazės komponentais. Turi būti galimybė korektiškai naudoti komponentus klasių modelio komponentų (objektų) generavimui. Ši problema sprendžiama objektų kūrimui ir atvaizdavimui naudojant stereotipizuotas klases. Tačiau norint naudoti ISK pasiūlytą saugyklą, reikia ją pakeisti ir papildyti taip, kad ji būtu pritaikoma skirtingoms situacijoms. Tai reikia atlikti, nes dabartinė saugykla turi du didelius trūkumus. Tai:

- Perteklinis elementų kiekis;
- Saugomas tik vienas modelis.

#### 3.3.2. Pakeitimai esamam EM (veiklos žinių bazei) saugyklos modeliui

ISK pasiūlyta saugykla (priedas nr. 1) vienu metu gali talpinti tik vieną modelį. T.y. dėl to, kad nėra identifikuojama, kokiam modeliui ar projektui priklauso atitinkamas objektas. Tad, jei į realizuotą, pagal esamą modelį, saugyklą surašyti kelių modelių informaciją, išgavus šią informaciją realiai galima būtų atskirti modelius, tačiau pats procesas būtų labai apsunkintas.

Tai padaryti galima, nes ryšiais sietųsi tik atitinkami elementai. Tačiau automatizuotas atskyrimas būtų daug sudėtingesnis nei neautomatizuotas.

Ši problema išsprendžiama įvedus naują parametrą (modelio identifikatorius). Esamoje loginėje schemoje yra numatyta vieta saugoti informaciją apie veiklos modelį lentelėje „EnterpriseModel“ (priedas nr. 1). Ši lentelė nesiejama su kitais elementais. Norint išspręsti šią problemą, pagrindiniams ele-

mentams: „Procesas“, „Aktorius“, „Įvykis“, „Funkcija“, „Informacinė veikla“, „Veiklos taisyklė“, „Materialus srautas“ bei „Informacinis srautas“ suteikiamas papildomas atributas „Model_id“. Pagal šį atributą galima paprastai identifikuoti vieno modelio elementus. Kitiems elementams nereikalingas šis papildomas atributas, nes jie susieti su pagrindiniais elementais ir yra ne sunkiai atsekami per juos.

Nors šis problemos sprendimas leis saugoti daug veiklos modelių, bet kartu ir įtakos poreikį keisti esamą klasių modelio generavimo algoritmą. T.y. reikės atsižvelgti į tai, kad saugykloje yra informacijos nesusijusios su pasirinkta veiklos sritimi.

Kita problema – informacijos dubliavimas. Ši problema nėra kritinė. Ir jos sprendimas nėra būtinas. Tačiau dabartinės loginės duomenų bazės schemos analizė parodė, kad atitinkama informacija, pagal esamą loginę duomenų bazės schemą gali būti dubliuojama keletą kartų. Pavyzdžiui esant kelioms srauto būsenoms, į duomenų bazę bus surašoma tiek informacinių to paties srauto eilučių, kiek yra srauto būsenų.

Ši problema gali išspręsta pertvarkant atitinkamas duomenų bazės lenteles. Pavyzdžiui srautų būsenas galima aprašinėti atskirose lentelėse. O srautus sieti su srauto būsenomis per tarpines lenteles (per tas, kurios srautus sieja su procesais ar informacinėmis veiklomis).

Šios problemos sprendimas nėra privalomas, tačiau toks sprendimas ne tik sumažins duomenų bazės įrašų kiekį, bet ir gali ženkliai pagreitinti informacijos talpinimo bei išgavimo trukmę, dėl to, kad informacijos, kurią reikia apdoroti, kiekis bus mažesnis.

Pakeistas koncepcinis loginis veiklos žinių bazės saugyklos modelis pateiktas priede nr. 2. Šiame modelyje palikti visi buvę elementai. Išspręsta atskirų veiklos modelių identifikavimo problema. Tad naujoje saugykloje bus galima talpinti didelį kiekį skirtingų veiklos modelių (kiek fiziškai leidžia duomenų bazė).

Antroji problema nėra pilnai sprendžiama. T.y. sukuriamos papildomos lentelės pasikartojantiems duomenims iškelti bei sukuriami papildomi identifikavimo atributai. Tačiau paliekami ir buvę atributai. Todėl informacija iš dalies lieka perteklinė. Ateityje ši problema gali būti sprendžiama. Tada sprendimas bus optimalus.

### 3.4.Esamo EM (veiklos žinių bazės) saugyklos modelio pritaikymas prie „MagicDraw“

Atlikta esamos veiklos žinių bazės analizė parodo, kad norint naudoti bazėje saugomą informaciją apie srities veiklą, siekiant „MagicDraw UML“ pakete generuoti klasės modelį, reikia išplėsti esamą paketo klasės meta-modelį. Plečiant klasių meta-modelį automatiškai yra išplečiamas ir pats klasių modelis. Tad šis toliau vadinamas išplėstu klasių modeliu (ECM).

Kadangi ISK pasiūlytas algoritmas turi papildoma funkcionalumą, kuris nėra realizuojamas šiame darbe (klasių modelio patikra, kuri apima klasių modelio papildymą reikiamais komponentais ir apribojimą esamų, pagal veiklos taisykles), paliekama galimybė pritaikyti šį funkcionalumą. T.y. „MagicDraw UML“ paketas (klasių modelio meta-modelis) bus išplėstas taip, kad turėtų visus reikiamus komponentus šio funkcionalumo realizavimui.

Klasių modelio meta-modelio išplėtimas reikalingas tam, kad nebūtų informacijos praradimo. Generuojant klasių modelį pagal, veiklos žinių bazę, bus kuriamos ne paprastos, o stereotipizuotos klasės su savais atitinkamais parametrais. Norint tai atlikti, reikalingas meta-modelio išplėtimas.

Veiklos žinių panaudojimas klasių modelio generavimui realizuojamas per paketo įskiepį. Taip paketo įskiepis turės susirišimą su veiklos žinių baze (duomenų baze) ir generavimo metu bus naudojami atitinkami saugomi įrašai.

## 3.5.Klasių meta-modelio išplėtimas „MagicDraw“ pakete

### 3.5.1. Klasių meta-modelio išplėtimas naudojant „MagicDraw“ profilį (angl. profile)

Sugeneruotas klasių modelis pagal veiklos žinių bazę, turės papildomus (išplėstus) objektus. Tad jų realizavimui reikalingas esamo paketo klasių modelio išplėtimas.

Pakete naudojama OMG pasiūlyta metodologija. T.y. naudojamas standartinis UML2 profilis. Profilis ir jo meta-modeliai saugomi „MagicDraw UML“ branduolyje (angl. kernel). Paketo autorių pateiktoje dokumentacijoje nurodyta, kad nėra techninių galimybių pakeisti „MagicDraw UML“ branduolį.[17]

Dokumentacijoje nurodoma, kad paketo funkcionalumą galima papildyti ar apriboti (dalinai) naudojant „įskiepius“. Tačiau modelių išplėtimai ir apribojimai galimi išplečiant ar ribojant meta-modelius ir gali būti atliekami trim būdais:

- Naudojant paketo profilius (angl. profiles),
- Naudojant paketo įskiepius,
- Naudojant ribojimus „constraints“

Nurodoma, kad tik šiais būdais galimas paketo papildymas ir funkcionalumo apribojimas. Planuojama naudoti du būdus: įskiepis bus naudojamas algoritmo funkcionalumo realizavimui, o profilis bus naudojamas klasių modelio meta-modelio išplėtimui bei stereotipizuoto, išplėsto klasių modelio braižymo aplinkos sudarymui. Pastarasis būdas (meta-modelio išplėtimas panaudojant profilį) galimas tik 15-toje ar naujesnėje paketo versijoje. Aišku, kad senesnių versijų vartotojai negalės naudotis pasiūlytu algoritmu [17].

Naudojant „MagicDraw UML“ profiliavimo mechanizmą, kuriamas naujas paketo profilis naudojant DSL (angl. Domain Specific Language) – srities specifinę kalbą, kurios dėka aprašomi nauji – išplėsti klasių modelio elementai. T.y. bus aprašomi nauji klasių stereotipai (stereotipizuotos klasės) bei kiti reikiami komponentai.

### 3.5.2. „MagicDraw UML“ profilio sudarymas ir panaudojimas

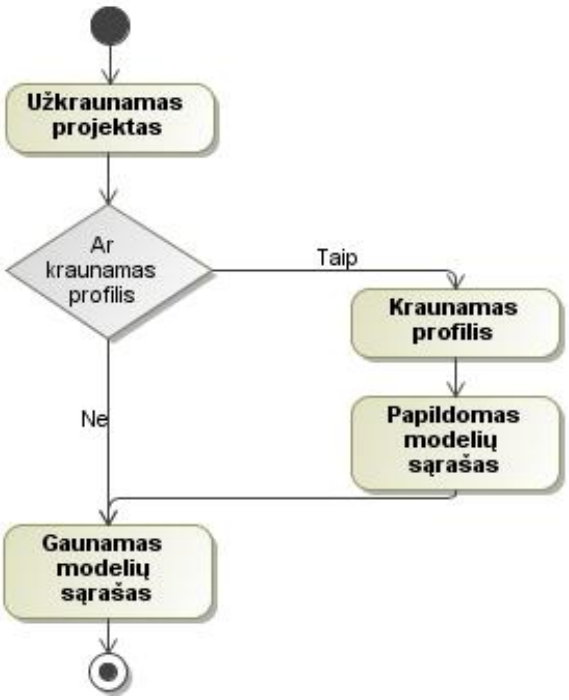
„MagicDraw UML“ pakete naudojami profiliai gali būti apibrėžti kaip tam tikros rūšies paketai, kurie naudojami bibliotekų, srities specifinių duomenų bei stereotipų rinkiniams saugoti.

Iš pateiktos dokumentacijos seka, kad „MagicDraw UML“ paketas turi kelias UML profilio išplėtimo galimybes (priemones) [22]:

- Specifinių diagramų sudarymas – suteikiama galimybė sudaryti savas diagramas atitinkamiems profiliams. Šiose diagramose gali būti naudojami atitinkami parinkti objektai, stereotipizuoti elementai, simboliai bei manipuliacijos. Taip pat galima sudaryti reikiamus operacijų meniu.
- DSL panaudojimas – suteikia galimybę derinti, keisti vartotojo sąsajas bei suteikia galimybe sudaryti specifinius dialogus.
- Profilių panaudojimas (derinimas) – suteikia galimybę paslėpti nereikalingus elementus, sudaryti naujus bei suteikia darbo su atributais (angl. tag) galimybę.

Šios priemonės suteikia galimybę panaudoti paketo profiliavimo mechanizmą išplėsto klasių modelio darbiniai aplinkai sudaryti.

Svarbi profilio savybė – profilis gali būti sudarytas bet kokiame projekte. Dažniausiai profiliai kuriami daugkartiniam panaudojimui (daugybėje projektų). Norint pakartotinai naudoti sudarytus profilius, jie turėtų būti sudaryti ir saugomi atskiruose, nepriklausomose bylose. Šios bylos turi būti paskelbtos globaliomis (shared) ir iš jų turi būti sudaryti moduliai. Moduliai yra tam tikrų modelių fragmentai, laikomi išorinėje byloje. Jie gali būti susiejami su kitais modeliais bei gali būti juose panaudojami.[22] (žiūrėti pav. 16)

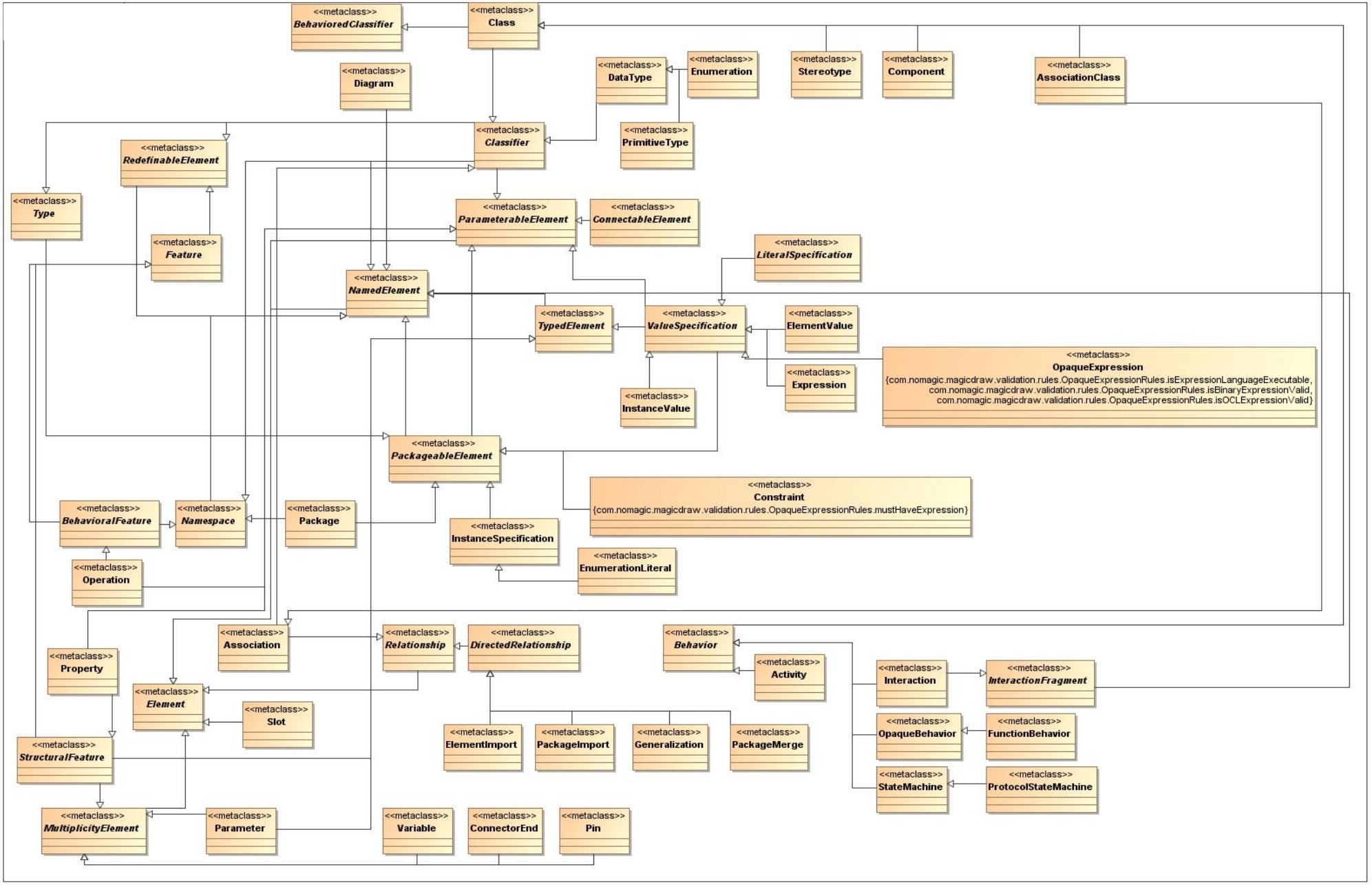



Pav. 16 Profilio teikiamas funkcionalumas.

Pakete užkrovus atitinkamą profilį, vartotojui suteikiami ne tik nauji elementai darbui, bet ir pateikiama aplinka iš kart dirbti su specifinėm diagramom. Užsikrovus pateiktą ir sudarytą profilį, vartotojams bus pateikiama reikiama darbinė aplinka, skirta dirbti su išplėstu klasių modeliu.

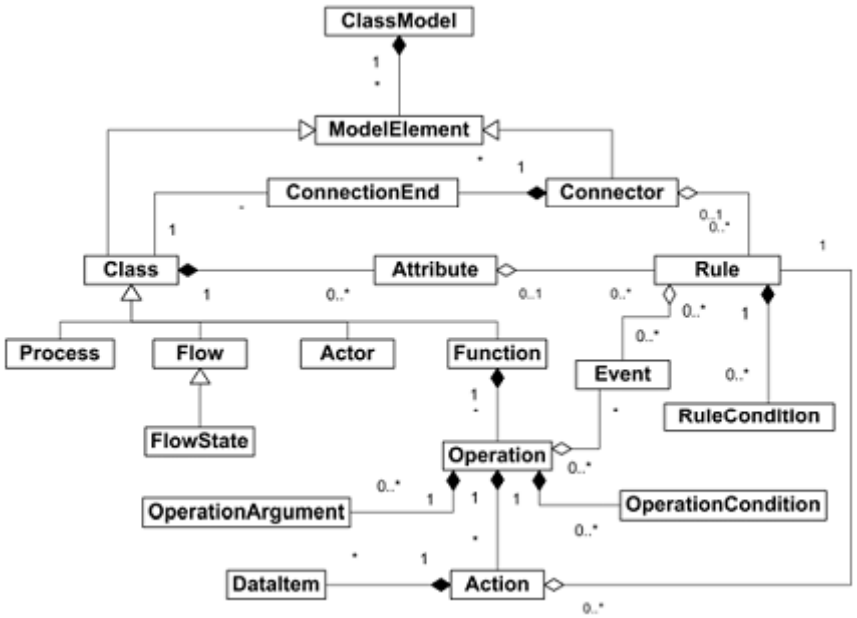
### 3.5.3. Esamas klasių meta-modelis

Darbe atlikta analizė parodė, kad paketas „MagicDraw UML“ naudoja pilną OMG pasiūlytą UML2 meta-modelį.



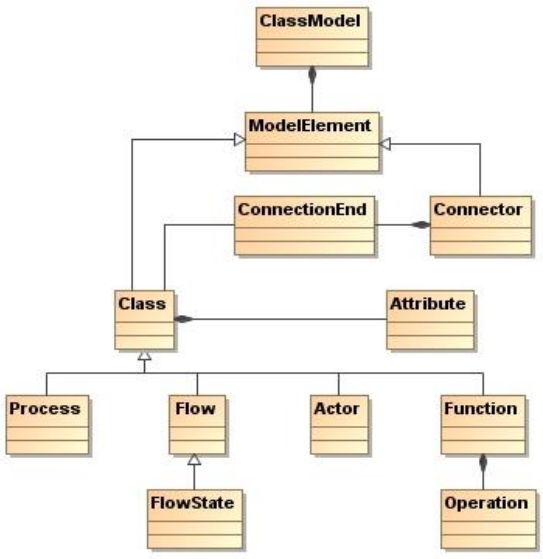

Pav. 17 Esamo "MagicDraw UML" klasių meta-modelio fragmentas

Šis modelis (modelio fragmentas) (pav. 17) gaunamas per „MagicDraw UML“ paketą. T.y. iš paketo klasių meta-modelio (iš branduolio) paimama „klasės“ meta-klasė ir paketo dėka gaunami susiję elementai. Tai – tik modelio fragmentas. Modelyje pateikiami ne visi meta-modelio elementai. Tačiau esamų visų elementų nepakanka algoritmo realizacijai. Tad pagal atliktą analizę atliekamas projektavimas reikiamiems elementams identifikuoti, kad jais būtų išplėstas meta-modelis. Tam tikslui naudojamas teorinis išplėstas klasių meta-modelis [8] (pav. 18). Taip pat naudojamas veiklos meta-modelis (pav. skyrius 2.5). Jame pateiktas detalesnis vaizdas. Nors reikiamų elementų identifikavimui pakanka esamo teorinio modelio.




Pav. 18 Išplėstas teorinis klasių meta-modelis [8]

Pateiktame teoriniame išplėstame klasių modelio meta-modelyje (pav. 18) yra algoritmui (klasių modelio generavimui) nereikalingų – perteklinių elementų. T.y. algoritmas nenumato tokių elementų sudarymo ar pildymo. Tad iš šio modelio pertekliniai elementai yra pašalinami (jie nebus naudojami) ir gaunamas patikslintas teorinis išplėstas klasių meta-modelis (pav. 19).



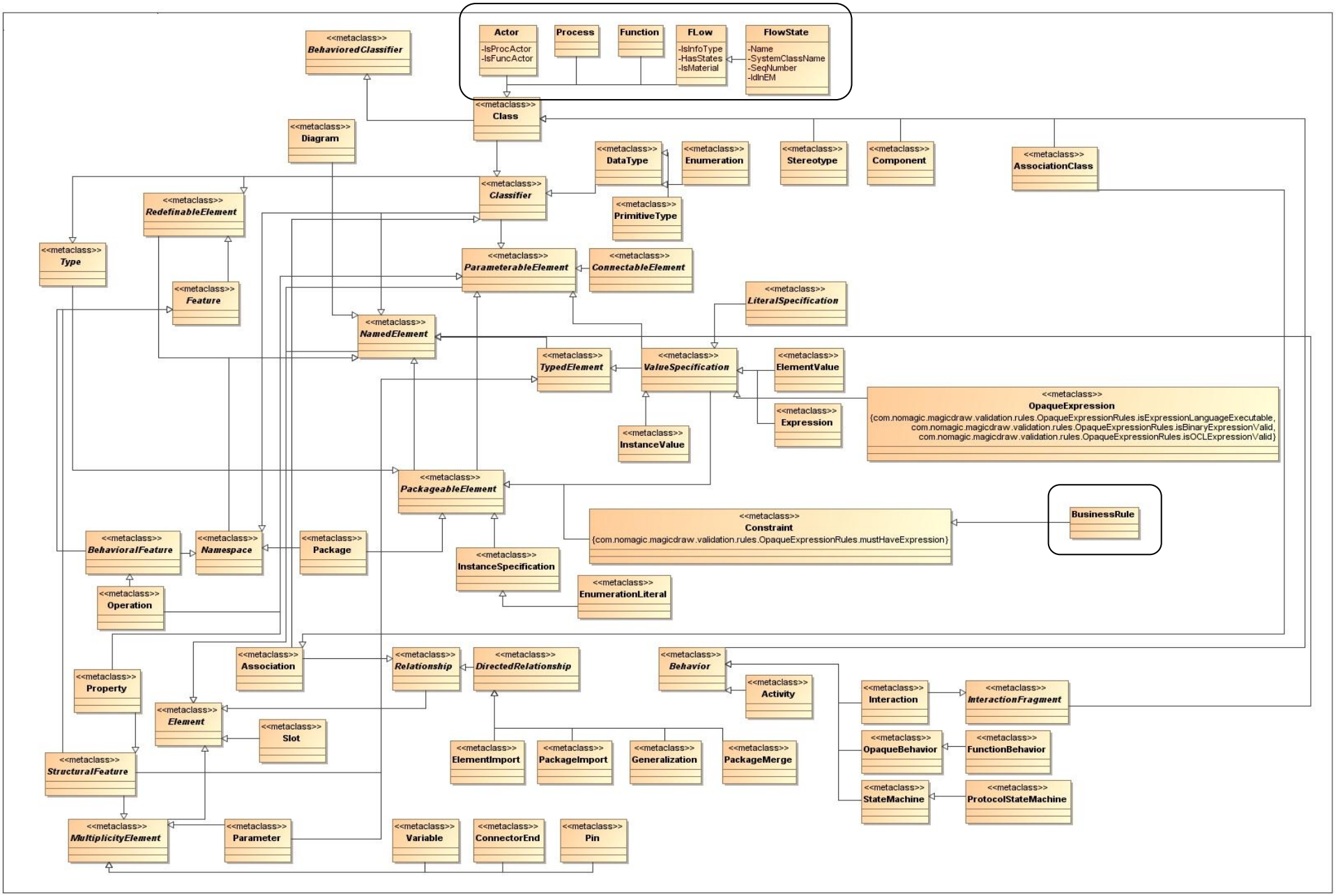

Pav. 19 Patikslintas pagal algoritmą teorinis išplėstas klasių meta-modelis

Šis modelis (pav. 19) identifikuoja visus algoritmo generuojamus elementus. Iš jo aiškiai matomi visi klasių modelio elementai – stereotipizuojamos klasės. Šis modelis toliau bus naudojamas kaip pagrindas esamo OMG pasiūlyto UML2 meta-modelio, naudojamo „MagicDraw UML“ pakete, išplėtimui.

### 3.5.4. Išplėstas klasių meta-modelis

Palyginus modelyje (pav. 19) esamus reikiamus komponentus su dabartiniame paketo klasių meta-modelyje (pav. 17) naudojamais komponentais, identifikuojami trūkstami objektai. Trūkstamais objektais papildomas esamas meta-modelis ir taip gaunamas reikiamais komponentais išplėstas klasių meta-modelis, kuris bus naudojamas algoritmo realizacijai.

Naujas – papildytas meta-modelis pateiktas sekančiame paveikslėlyje (pav. 20).




Pav. 20 Išplėsto "MagicDraw UML" klasių meta-modelio fragmentas

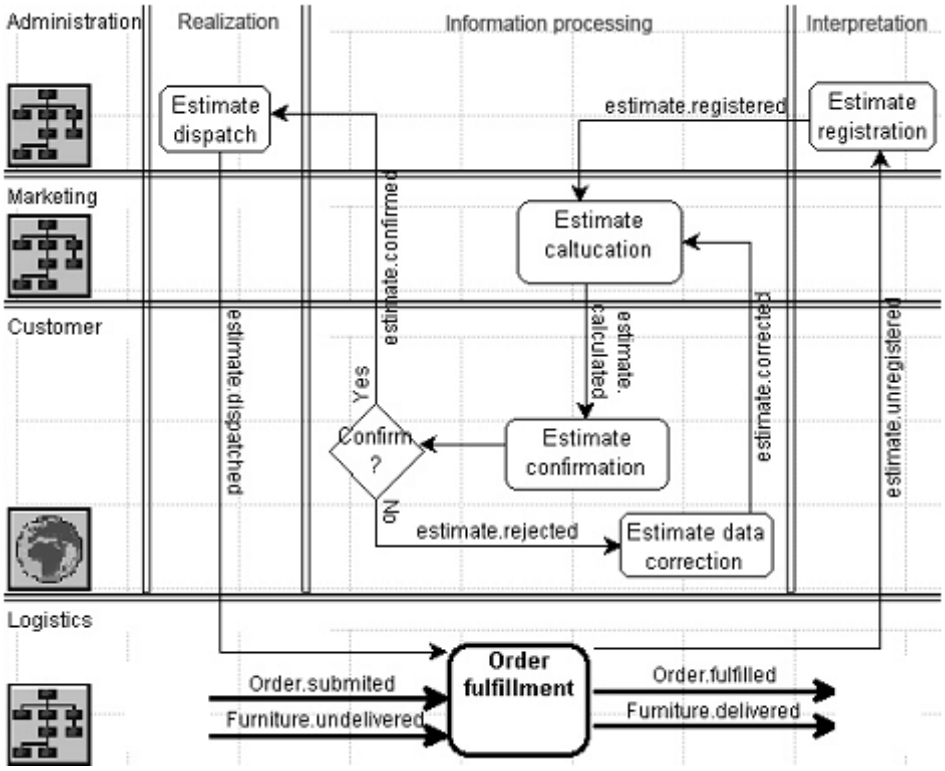
Pateiktas modelis (pav. 20) – meta-modelis yra papildytas trūkstamais elementais. Papildyti elementai yra išskirti modelyje (apibrėžti).

Algoritmo realizavimui reikalingi tik pačios klasės išplėtimai (pateikta viršuje). T.y. šie elementai atitinka teoriniame meta-modelyje (pav. 19) pateiktus elementus. Tačiau meta-modelis praplečiamas ir veiklos taisyklėmis. Tai daroma tuo tikslu, kad būtų galimybė pritaikyti kitus algoritmus sudaryto klasių modelio patikslinimui pagal VT (tai nėra realizuojama šiame darbe).

Šis meta-modelio variantas gali būti papildomas kitais elementais. Tačiau šiuo metu yra daroma prielaida, kad meta-modelis yra pakankamas algoritmo realizacijai. Kitų papildymų ar apribojimų poreikis gali atsirasti realizacijos metu.

### 3.6.Praktinis algoritmo taikymas. Išplėsto klasių meta-modelio panaudojimas generuojant klasių modelį

Nors ir yra apibrėžta, kad klasių modelis bus generuojamas tik griežtai pagal veiklos žinių bazėje saugomus įrašus, galima pateikti pavyzdį, kaip gali atrodyti sugeneruotas klasių modelis pagal pateiktą EM. Šis pateiktas pavyzdys vaizduoja, kokie turėtų būti gaunami rezultatai, kaip pagrindą naudojant pateiktą veiklos procesų modelį (pav.: 21).




Pav. 21 Veiklos procesų sekų modelis (užsakymo įvertinimo valdymas) [1]

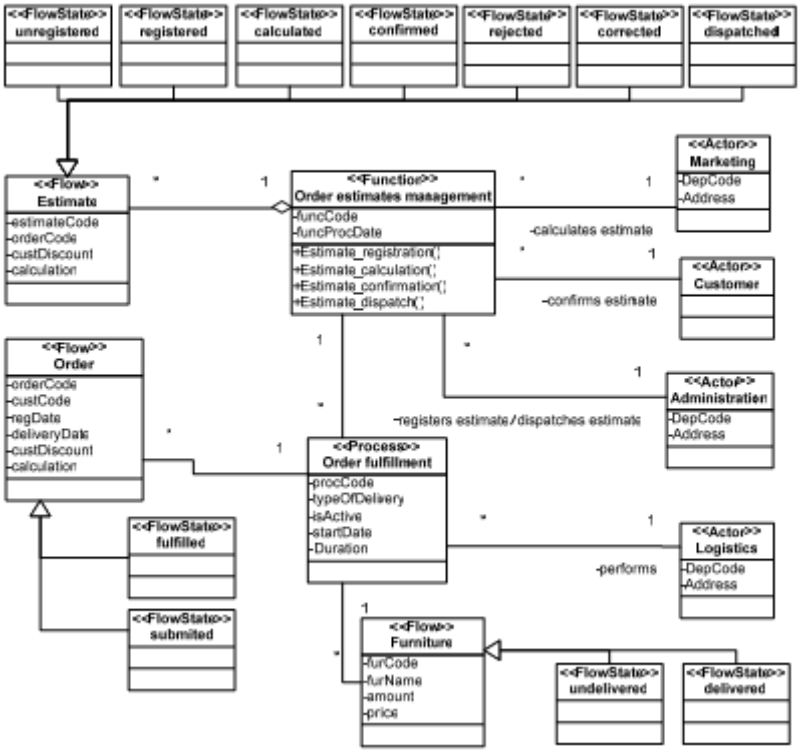
Modelyje (pav. 21) galima išskirti sąveiką tarp „užsakymo įvertinimo valdymo“ funkcijos bei „užsakymo patvirtinimo“ proceso. Modelyje taip pat matomi išskirti aktoriai (kairėje pusėje), jų vykdomi procesai ir sekos, kuriomis jie atliekami.

Tad taikant pasirinktą klasių modelio algoritmą atliekami atitinkami žingsniai:

1. Parenkama atitinkama valdymo funkcija „Užsakymo įvertinimo valdymas“ ir sukuriama jai stereotipizuota klasė (<<**Function**>> „Order estimates managment“).

2. Analizuojamas veiklos modelis (EM) ir renkama bei apdorojama visa informacija (aktoriai, procesai bei informaciniai srautai), susijusi su pirmame algoritmo žingsnyje apibrėžta klase.
   Procesams sukuriami stereotipizuoti objektai <<**Process**>>: „Order fulfillment“;
   Aktoriams sukuriami stereotipizuoti objektai <<**Actor**>>: „Marketing“, „Customer“, „Administrattion“, „Logistics“;
   Taip ieškomi ir apibrėžiami susiję materialūs srautai. Materialiems srautams sukuriami stereotipizuoti objektai <<**Flow**>>: „Estimate“, „Order“, „Furniture“;
   Srautams apibrėžiamos būsenos ir sukuriami stereotipizuoti objektai <<**FlowState**>>. Jie atvaizduojami sekančiame modelyje.
3. Pagal veiklos žinių bazę (šiuo atveju veiklos modelį) sudaromi ryšiai tarp objektų: asociacija tarp <<Process>> ir <<Function>>...
4. (Sudarytų ryšių patikra – nėra realizuojama)
5. Pagal veiklos žinių bazę generuojami klasių atributai (šiuo atveju tik sisteminiai).
6. (Sudarytų atributų patikra – nėra realizuojama)
7. Tik <<**Function**>> tipo stereotipizuotoms klasėms sudaroma operacijų aibė. Operacijų lygis generuojamas pagal informacinę veiklą, kuri yra valdymo funkcijos sudėtinė dalis:
   Estimate_registration();
   Estimate_calculation();
   Estimate_confirmation();
   Estimate_dispatch();
8. Atmetamos (pažymimos kaip nematomos) nereikalingos operacijos ir funkcijos. Tai atliekama, nes ne visos funkcijos yra kompiuterizuojamos ir kai kurios yra perteklinės. Jų identifikavimą palengvina panaudos atvejų modelis.
9. Veiklos modelio saugykloje saugomos operacijų taisyklės naudojamos formaliai apibrėžti kompiuterizuojamas operacijas.

Atlikus visus reikiamus algoritmo žingsnius, galima pabrėžti atitinkamus faktus: procesui priklauso ir materialūs srautai (įvesties bei išvesties atributai). Modelyje srautai taip pat turi atitinkamas būsenas. Tam tikros veiklos modelyje nėra kompiuterizuojamos ir jos nėra vaizduojamos generuojamame klasių modelyje. Atsižvelgiant į šiuos apibrėžimus, galimas sugeneruotas klasių modelis pagal šį veikos modelį atrodytų atitinkamai, kaip pateikta paveikslėlyje (pav. 22).




Pav. 22 Sugeneruotas klasių modelis pagal EM modelį "Užsakymo įvertinimo valdymas" [1]

Šis modelis „MagicDraw UML“ pakete bus atvaizduojamas stereotipizuotomis (išlėstomis) klasėmis. T.y. nors pateiktame klasių modelyje atvaizduojami tik klasės bei ryšio tipo objektai, šiuo atveju klasės jau yra įvairių tipų (funkcija, procesas, aktorius, srautas bei srauto būsena).

Grafiniame modelyje nėra atvaizduoti visi veiklos žinių bazės elementai. Pavyzdžiui modelyje nėra matomos veiklos taisyklės. Jos saugojamos atitinkamomis formalizuotomis tekstinėmis išraiškomis atskirai nuo procesų srautų modelio. Veiklos objektų atributai šiuo atveju yra gaunami iš veiklos modelio bazės.

Šio klasių modelio apribojimai yra apibrėžti pasirinktos valdymo funkcijos. Svarbu pabrėžti, kad sudarinėjant modelį grafinėje aplinkoje labai svarbu yra apriboti tam tikrų objektų sąveiką (<<Actor>>, <<Flow>> bei <<Function>>).

### 3.7.Algoritmo pagrindimas realizavimui pagal turimą EM saugyklą ir klasių meta-modelį

Algoritmo veiksmingumas ir pritaikomumas pagrindžiamas keliais aspektais. Pirmasis – algoritmas realiomis sąlygomis pritaikomas klasių modelio generavimui. Antrasis aspektas – yra algoritmo funkcionalumui tinkama aplinka.

Pirmasis aspektas buvo pagrįstas neautomatizuotai, taikant algoritmą klasių modelio generavimui (skyrius 3.6). Nors, šiuo atveju, buvo naudojama ne veiklos žinių saugykla, o veiklos procesų sekų modelis, tačiau saugykloje saugomi įrašai atitiks (ir net galės būti detalesni) žinias modelyje. Klasių modelis buvo sudaromas eilės tvarka vykdant algoritmo žingsnius. Ir pagal visus apibrėžimus yra gaunamas teisingas išplėstas klasių modelis.

Antrasis aspektas pagrindžiamas projektuojant esamo „MagicDraw UML“ paketo išplėtimą (išplėtimas bei apribojimas). T.y. apibrėžiamos išplėtimo priemonės (įskiepis bei profilis) bei sritys, kurias reikia išplėsti.

Algoritmo realizavimui tinkama aplinka yra gaunama panaudojant profilį. Profilis yra naudojamas esamo UML2 meta-modelio išplėtimui reikiamais komponentais, kurie nėra apibrėžti esamame modelyje. T.y. sukuriami klasių stereotipai su savais papildomais atributais bei ribojimais, siekiant panaudoti veiklos žinių bazėje saugomas žinias.

## 3.8.Algoritmo pakeitimo pagal realias sąlygas projektavimas

### 3.8.1. Sąlygų pasikeitimas

Atlikus eksperimentą su ISK pasiūlytu teoriniu algoritmu remiantis pasirinktais pradiniais duomenimis (veiklos modeliu) matyti, kad algoritmas veikia. Tačiau realios sąlygos skiriasi nuo eksperimento sąlygų. Siekiant efektyviai išplėsti ar pakeisti algoritmą, reikia identifikuoti sąlygų pasikeitimus:

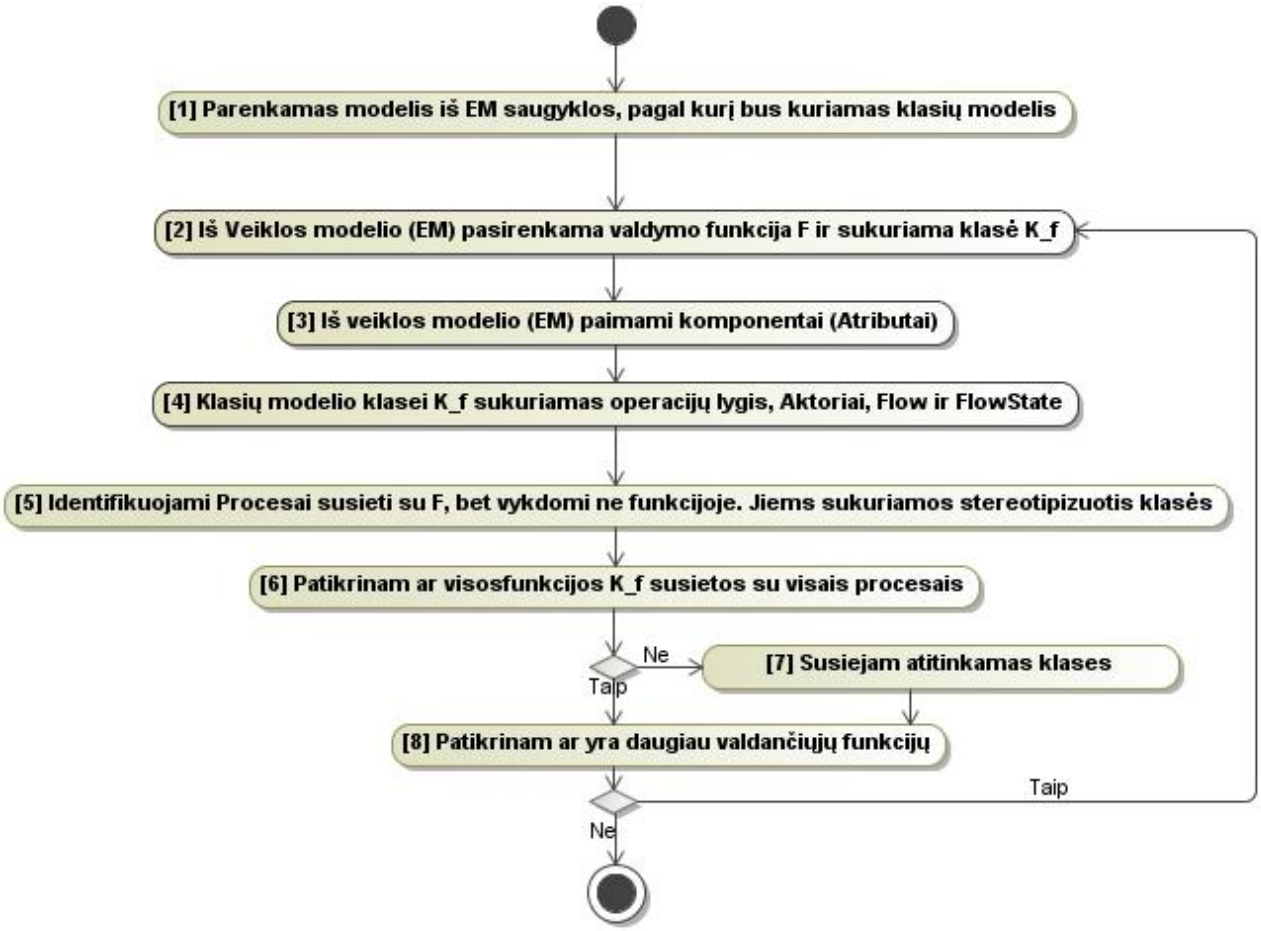
- Saugykloje saugomas daugiau nei vienas modelis – t.y. pakeitus loginę duomenų bazės schemą (skyrius 3.3), pasikeičia ir algoritmo vykdymas. T.y. reikia identifikuoti, ar naudojamas to modelio elementas.
- Papildomų valdančiųjų funkcijų įtraukimas – sugeneruotame klasių modelyje gali būti ne viena, o kelios valdančiosios funkcijos. T.y. gali būti kelios valdančiosios funkcijos, valdančios tuos pačius procesus ir net turinčios bendrus srautus.
- Informacinės veiklos atskyrimas nuo materialios – vartotojui gali būti nesvarbi materiali veikla (procesai, materialūs srautai, srautų būsenos). Todėl turi būti galimybė atvaizduoti tik informacinę veiklą (funkcijos, informaciniai srautai, srautų būsenos).
- Algoritmo žingsnių eiliškumas pagal DB informacijos išgavimą – tam tikrus algoritmo žingsnius sukeitus vietomis realizacija tampa paprastesnė. T.y. todėl, kad informacija iš veiklos žinių bazės turi būti išgaunama atitinkama tvarka (pagal tai, kaip siejamos lentelės).

### 3.8.2. Algoritmo išplėtimas

Išanalizavus sąlygų pasikeitimus atliekamas algoritmo (pav. 9) projektavimas esamu teoriniu algoritmo pagrindu (ISK algoritmas).

Pradžioje buvo išmestas nerealizuojamas algoritmo funkcionalumas. T.y. (žingsnis 4) bei (žingsnis 6) algoritmo žingsniai. Šiais žingsniai atliekama sudaryto klasių modelio (klasių atributų, klasių ryšių) patikra ir pataisymas/papildymas pagal veiklos taisykles. Norint realizuoti šias operacijas, reikia tūrėti veiklos taisyklių saugyklą. Šis funkcionalumas gali būti realizuojamas atskirai, kitu įskiepiu. Be to šiems žingsniams nėra privalomas eiliškumas nurodytas algoritme. Tad jie gali būti realizuoti atskiru algoritmu ir taikomi jau sugeneruotiems modeliams. Šios operacijos nepateikiamos naujame algoritme (pav. 23).

Pirmasis žingsnis pagal ISK pasiūlytą algoritmą tampa antruoju pakeisto algoritmo žingsniu. Algoritmas papildomas naujais žingsniais. Pradžioje atliekamas modelio pasirinkimas (žingsnis 1). T.y. atliekama todėl, kad pagal naujas sąlygas veiklos žinių saugykloje gali būti saugoma daugiau nei vienas veiklos modelis.



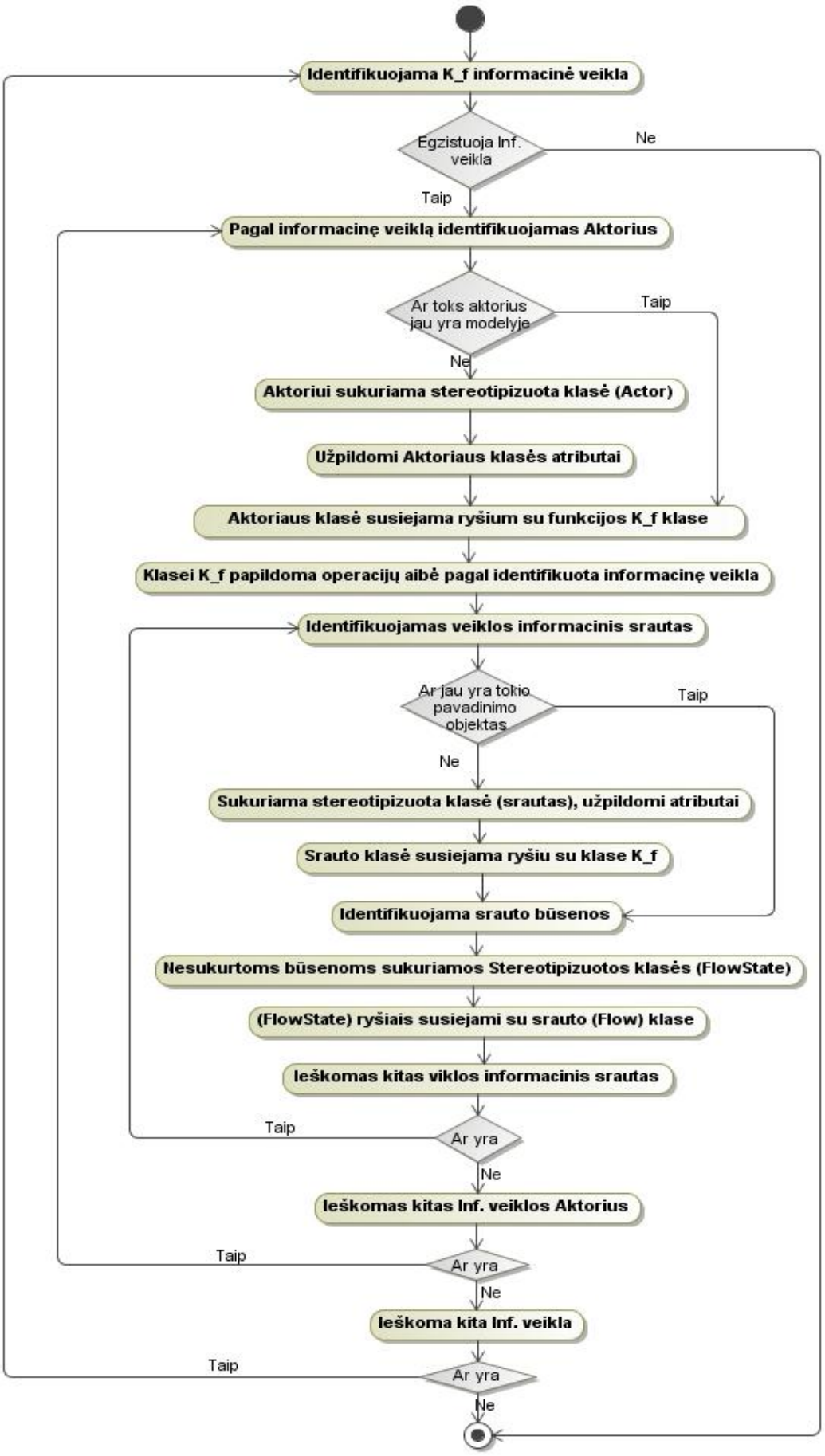

Pav. 23 Pakeistas algoritmas

Algoritmas papildomas naujais žingsniais ((žingsnis 6) bei (žingsnis 7)). Žingsnyje (žingsnis 6) patikrinama ar nėra nesusietų funkcijos tipo klasių su proceso tipo klasėmis. Radus nesusietų klasių vykdomas (žingsnis 7) algoritmo žingsnis, kuriame susiejamos atitinkamos klasės. Neradus nesusietų elementų vykdomas paskutinis algoritmo žingsnis.

Kitas numatytas papildymas yra galimybė vaizduoti kelias valdančiąsias funkcijas. Tuo tikslu algoritmas padaromas iteraciniu. T.y. įtraukiamas žingsnis (žingsnis 8), kuriame patikrinama ar nėra daugiau valdančiųjų funkcijų pasirinktam modeliui. Jei randama neatvaizduotų funkcijų, kartojama žingsnių seka (žingsnis 2) – (žingsnis 8).

Pagal pakeistą algoritmą žingsnyje (žingsnis 3) surenkami ir užpildomi funkcijos klasės K_f atributai. Pagal ISK pasiūlytą algoritmą tai buvo penktasis žingsnis ir jis, pagal savo koncepciją, kiek skiriasi nuo žingsnio naujajame algoritme. T.y. ISK algoritme žingsnis 5 užpildomi visų klasių atributų lygis. Tuo tarpu šiame algoritme, dėl to, kad procesas tapo iteracinis, atributų lygis užpildomas kiekvienai klasei jos sukūrimo metu.

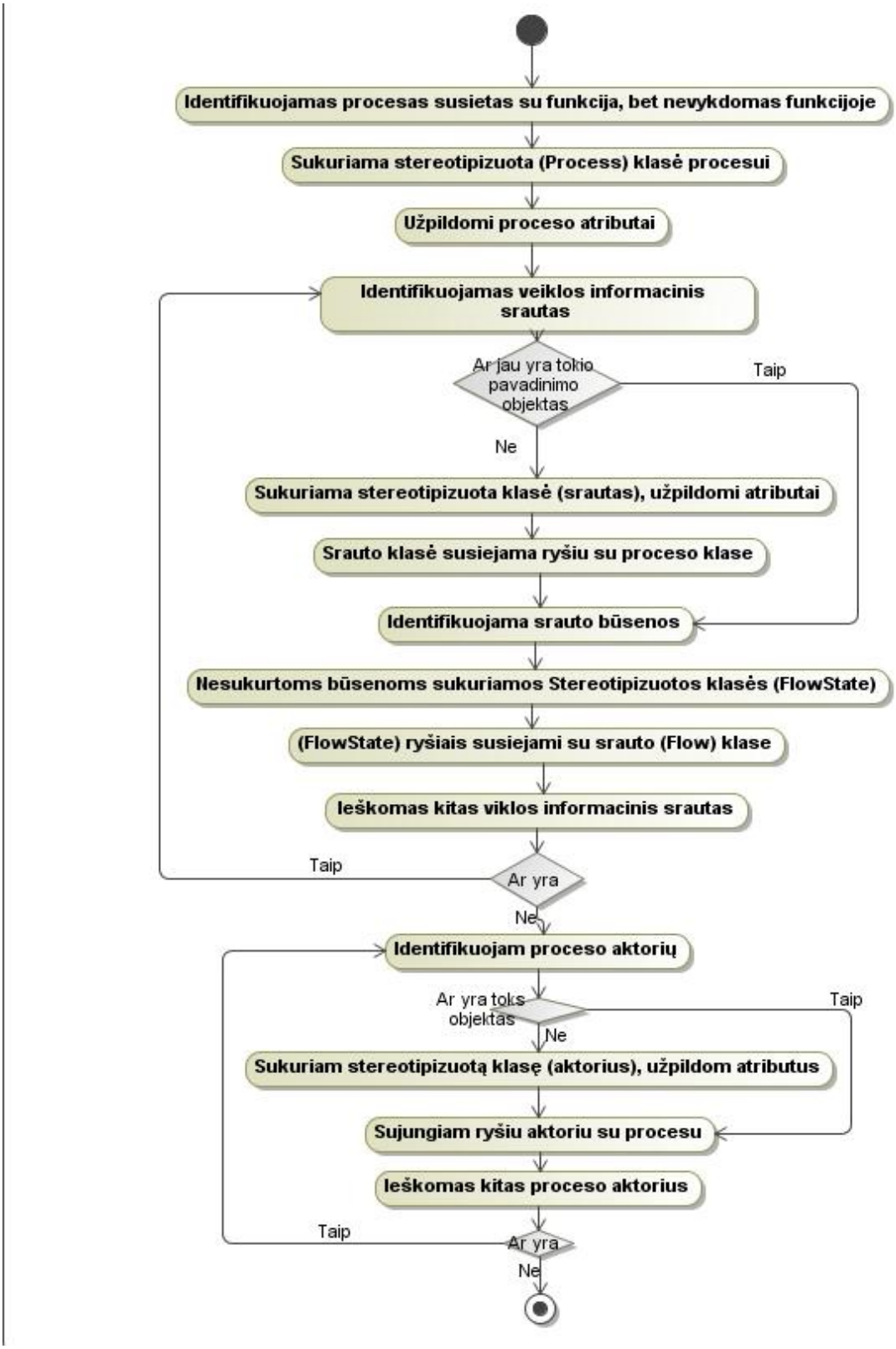
Algoritmas papildomas žingsniu (žingsnis 4), kuriame užpildomas funkcijos operacijų lygis bei funkcija susiejama su informaciniais srautais. Detalesnis vaizdas pateikiamas (pav. 24).




Pav. 24 Pakeisto algoritmo [4] žingsnio išplėtimas

Žingsnyje (žingsnis 4) generuojami visi funkcijos informacinės veiklos objektai. T.y. pagal informacines veiklas užpildomas funkcijos operacijų lygis (ISK algoritme žingsnis (žingsnis 7)). Taip pat sudaromi srauto tipo bei srauto būsenos tipo klasių objektai. Srautų bei jų būsenų išrinkimas ir sudarymas – iteraciniai procesai, kurių metu identifikuojami ir sudaromi tik trūkstami klasių egzemplioriai. Analogiškai identifikuojami ir veiklų aktoriai. Visi sudaryti klasių egzemplioriai siejami ryšiais su valdančiąja funkcija, tik srautų būsenos siejamos su srautais.

Žingsnyje (žingsnis 5) (pav. 25) sudaromi materialios veiklos objektų egzemplioriai.




Pav. 25 Pakeisto algoritmo [5] žingsnio išplėtimas

Procesas pagal savo struktūra yra labai panašus į informacinę veiklą. Tik šiuo atveju procesus galima įvardyti kaip materialią veiklą. Tad algoritmas pagal vykdomus žingsnius labai panašus į prieš tai aprašytą pagrindinio algoritmo žingsnį (žingsnis 4).

Šiame žingsnyje (žingsnis 5) identifikuojamas procesas susijęs su funkcija K_f ir užpildoma visa proceso informacija. Kiti algoritmo žingsniai yra iteraciniai. T.y. iteracijų kiekis priklausys nuo proceso srautų bei aktorių kiekių. Nes sekantys žingsniai yra skirti „Flow“ bei „FlowState“ klasių, susijusių su procesu, identifikavimui ir sukūrimui.

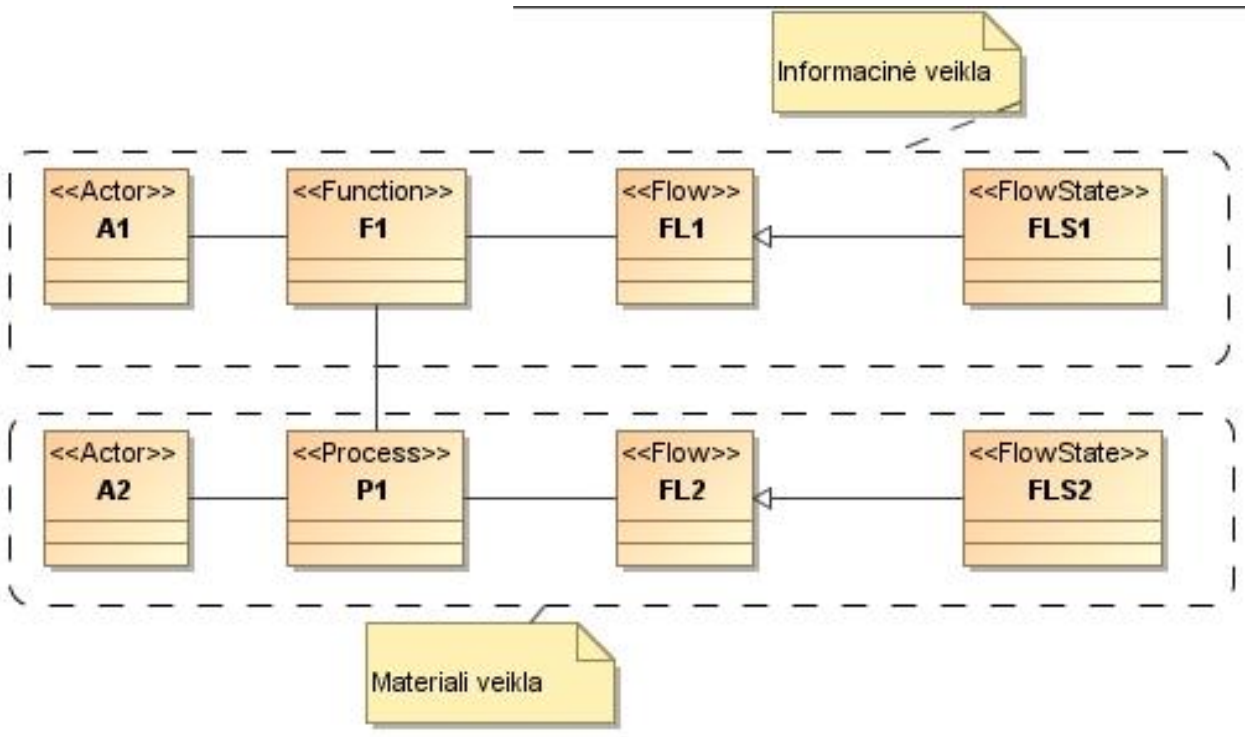
Numatytas būtent toks algoritmo žingsnių eiliškumas. T.y. tokio algoritmo realizacija yra paprasčiausia pagal turimas pradines sąlygas (turimą veiklos žinių bazės modelį).

### 3.9.Informacinės veiklos atskyrimas nuo materialios veiklos

Algoritmo vykdymo metu sudaromus objektus galima skirstyti į dvi grupes:

- Informacinės veiklos objektai;
- Materialios veiklos objektai.

Informacinės veiklos objektai yra valdančiosios funkcijos ir su ją susiję srautai (srautai ir srautų būsenos) bei aktoriai. Tuo tarpu visi objektai susiję su procesais (valdomais valdančiųjų funkcijų) yra materialios veiklos objektai. Tai taip pat gali būti aktoriai, srautai bei srautų būsenos. Vizualiai informacinės ir materialios veiklos objektai pateikti sekančiame paveikslėlyje (pav. 26).




Pav. 26 Informacinės bei materialios veiklos objektai

Kompiuterizuojant probleminę sritį svarbiausia projektuotojams yra informacinė veikla. T.y. galima sakyti, kad informacinė veikla yra kompiuterizuojama veikla. Tuo tarpu materiali veikla ir jos objektai nėra kompiuterizuojami. Dėl šios priežasties nėra tikslinga visais atvejais generuoti materialios

veiklos objektus. Tam tikrose situacijose jie tik apkraus modelius nereikalingais, pertekliniais elementais. Tokiu būdu bus apsunkinamas IS modeliavimo procesas.

Siekiant optimizuoti IS modeliavimo procesą, priimtas sprendimas – negeneruoti materialios veiklos objektų. T.y. standartiškai bus generuojami tik informacinės veiklos objektai (valdančiosios funkcijos ir su jomis susiję objektai).

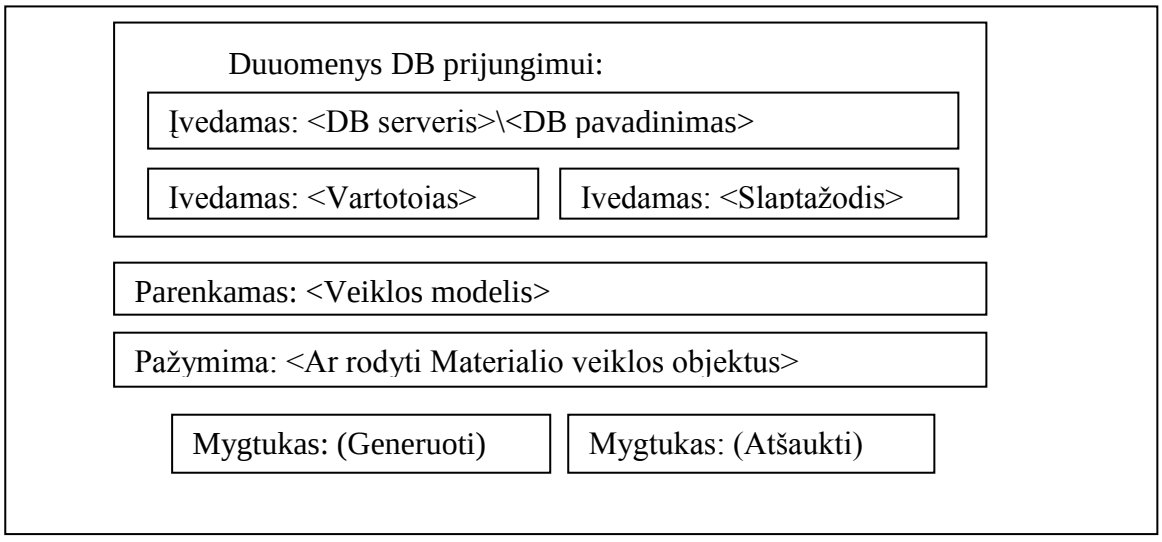
Tačiau tam tikrais atvejais IS projektuotojams sugeneruotame klasių modelyje gali būti reikalingi ir materialios veiklos objektai. Tad optimalus sprendimas yra pasirinktinis šių objektų generavimas. T.y. suteikiama galimybė vartotojams pasirinkti generuoti ar negeneruoti materialios veiklos objektus.

Remiantis anksčiau atliktais projektavimais, galima teigti, kad šio funkcionalumo realizavimui nereikia keisti suprojektuoto algoritmo, skirto klasių modelio generavimui pagal veiklos žinių bazę (skyrius 3.9). T.y. algoritme numatyta, kad materialios veiklos objektai yra sudaromi atskirais algoritmo žingsniais ([žingsnis 5] – procesų ir su jais susijusių objektų sudarymas, [žingsnis 6] – procesų susiejimo ryšiais su funkcijomis korektiškumo patikrinimas, [žingsnis 7] – nesusietų procesų susiejimas su funkcijomis). Tai leidžia nevykdyti šių algoritmo žingsnių (juos peršokti), kai nereikia generuoti materialios veiklos objektų.

Toks funkcionalumas (operacijų apribojimas) turi būti nurodomas vartotojo sąsajoje. T.y. paleidžiant klasių modelio generavimo algoritmą, vartotojui turi būti pateikta anketa, kurioje jis galėtų nurodyti, kokį klasių modelį jis norėtų matyti.

### 3.10. Sąsajos projektavimas

Atlikus analizę, buvo planuota, kad klasių modelio generavimas veiklos žinių bazės pagrindu bus paleidžiamas vienu mygtuko paspaudimu. T.y. tik paleidžiant „MagicDraw UML“ paketo įskiepio funkcionalumą. Projektuojant algoritmą, buvo pakeistos pradinės sąlygos ir išplėstas algoritmo funkcionalumas. Dėl šių pakeitimų atsirado papildomos „MagicDraw UML“ paketo įskiepio vartotojo sąsajos poreikis. Buvo suprojektuota sekanti vartotojo sąsaja, kurios maketas pateiktas paveikslėlyje (pav. 27).




Pav. 27 "MagicDraw" įskiepio vartotojo sąsajos maketas

Iš įskiepio vartotojo sąsajos maketo matyti, kad sąsajos langas skirstomas į dvi dalis: duomenų bazės susijungimo nustatymus (veiklos žinių bazės nurodymui) bei klasių modelio generavimo nustatymus.

Pirmoji sąsajos dalis (DB susijungimo informacijos nurodymas) reikalinga, nes analizės ir projektavimo metu nustatyta, kad galima naudoti skirtingas veiklos žinių bazes. Galima jungtis prie duomenų bazių esančių nutolusiame kompiuteryje. Tam tereikia tinkamai nurodyti susijungimo informaciją. T.y. reikia įvesti <DB serveris> lauką, kur nurodomas serveris prie kurio jungiamasi bei <DB pavadinimas> lauką, kur nerodomas veiklos žinių bazės pavadinimas nurodytame serveryje. Be šių laukų reikia įvesti vartotojo, kuriam leidžiama prisijungti prie veiklos žinių bazės informaciją (vartotojo vardas – laukas <Vartotojas>, vartotojo slaptažodis – laukas <Slaptažodis>). Šie laukai būtinai turi būti užpildyti (užpildyti korektiškai), priešingu atveju nebus susijungta su veiklos žinių baze ir algoritmas nebus vykdomas.

Klasių modelio generavimo nustatymai nėra būtini. Planuojama, kad nurodžius veiklos žinių bazę, laukui <Veiklos modelis> automatiškai parenkama pirmoji reikšmė iš žinių bazės (tik sėkmingo susijungimo atveju). Tačiau vartotojas gali iš iškrentančio meniu (laukas <Veiklos modelis>) parinkti kitą veiklos modelį.

Sekantis nustatymas yra skirtas nurodyti ar reikia generuoti materialios veiklos objektus (procesus ir su jais susijusius aktorius, srautus bei srautų būsenas). Tai laukas, kuriame uždėjus varnelę bus nurodoma, ar generuoti pilną klasių modelį (informacinės veiklos bei materialios veiklos objektus). Standartiškai varnelė yra uždėta tad rezultatas – sugeneruotas pilnas klasių modelis. Nuėmus varnelę generuojamas klasių modelis, kuriame bus sugeneruoti tik informacinės veiklos objektai (funkcija, jos aktoriai, srautai bei srautų būsenos).

Likę sąsajos elementai yra mygtukai. Mygtukas (<Generuoti>) skirtas patvirtinti nustatymus ir paleisti generavimo algoritmą. Mygtukas (<Atšaukti>) išjungia atidarytą „MagicDraw UML“ paketo įskiepio sąsajos langą.

### 3.11. Projektavimo apibendrinimas

Atliktos analizės metu nustatyta, kad paketas „MagicDraw UML“ tinka, kaip pagrindas, algoritmo realizacijai. T.y. jo turimi įskiepių panaudojimo bei profiliavimo mechanizmai leidžia išplėsti esamą funkcionalumą. Į tai atsižvelgiant, šis paketas pasirenkamas, kaip bazė, algoritmo funkcionalumo realizacijai. Ir tolimesnis projektavimas buvo grindžiamas „MagicDraw UML“ paketu.

Darbe aprašytas planuojamas būsimas sistemos (paketo) išplėstas funkcionalumas (atitinka funkcinius reikalavimus)(plačiau skyrius 3.2). T.y. projektuojant, numatyta, kad algoritmo realizacijos metu bus naudojamas įskiepis bei paketo profilis. Šie komponentai bus naudojami pasirinktinai ir tik išplės dabartinį paketo funkcionalumą. Išliks ir senas (dabartinis) funkcionalumas, tačiau priklausomai nuo to, su kokiais projektais bus dirbama (dabartiniais ar išplėstais), bus užkraunami arba ne atitinkami papildomi komponentai (profilis bei įskiepis).

Paketo profilis bus naudojamas esamo meta-modelio išplėtimui trūkstamais komponentais (plačiau skyrius 3.5). Taip pat paketo profilis leis sukurti naują modeliavimo aplinką. T.y. esama modeliavimo aplinka bus papildyta nauju modeliu (išplėstu klasių modeliu). Ši modeliavimo aplinka nėra būtina algoritmo realizacijai. Nors joje bus naudojamos stereotipizuotos klasės bei kiti objektai, kurie reikalingi norint klasių modelį sudarinėti neautomatizuotai.

Projektuojant numatyta, kad paketo įskiepis bus naudojamas algoritmo realizavimui bei susirišimui su veiklos žinių baze. T.y. apsibrėžta, kad veiklos žinių bazė bus realizuota duomenų bazėje. Duomenų bazė bus realizuota pagal ISK pasiūlytos bazės schemą (ISK pasiūlyta bazės koncepcinė schema – priedas Nr.1), ją pakeitus ir papildžius pagal atliktus projektavimus (nauja bazės schema – priedas Nr.2). Projektavimo metu numatyta, kad klasių modelio generavimui bus galima naudoti įvairias veiklos žinių saugyklas, atitinkančias duomenų bazės schemą. Jos net galės būti nutolusiame kompiuteryje.

Tam, kad algoritmas galėtų panaudoti veiklos žinių bazę, buvo planuojamas esamo meta-modelio išplėtimas.

Esamo meta-modelio išplėtimui buvo atlikta detalesnė modelio analizė. Pradžioje buvo detaliai analizuojamas OMG pateiktas UML2 meta-modelis. Ši dokumentacija laisvai prieinama. Atlikus šią analizę, OMG meta-modelis buvo palygintas su „MagicDraw UML“ pakete naudojamu meta-modeliu. Nustatyta, kad meta-modeliai atitinka.

Projektuojant meta-modelio išplėtimą, buvo lyginamas esamas pakete naudojamas meta-modelis su teoriniu veiklos meta-modeliu (veiklos žinių VMM bei teoriniu išplėstu meta-modeliu – detalesni aprašymai analizėje). Lyginant modelius buvo nustatomi trūkstami elementai. Pagal šiuos elementus sudarytas projektas meta-modelio papildymui (papildytas klasių meta-modelis) (plačiau skyrius 3.5).

Projektuojant buvo patikrintas praktinis algoritmo panaudojimas - eksperimentas (plačiau skyrius 3.6). T.y. veiklos modeliui buvo pritaikytas algoritmas. Algoritmas buvo vykdomas neautomatizuotai. Tačiau eilės tvarka buvo vykdomi algoritmo žingsniai ir buvo identifikuoti visi reikiami objektai. Pagal gautus objektus buvo sudaryti atitinkami klasių stereotipai ir sudaryta klasių diagrama.

Praktinis algoritmo pritaikymas bei veiklos žinių bazės schemos pakeitimo projektavimas leido identifikuoti privalomus algoritmo pakeitimus. T.y. projektavimo metu buvo identifikuotas ir suprojektuotas naujas algoritmo funkcionalumas (skyrius 3.8). Taip pat pakeistas tam tikrų algoritmo žingsnių eiliškumas. Visi pakeitimai leido optimizuoti algoritmą pagal realias pasikeitusias pradines sąlygas.

Pasikeitus „MagicDraw UML“ paketo įskiepio funkcionalumui, suprojektuota grafinė sąsaja, skirta pradiniams duomenims nurodyti (skyrius 3.10). Pasikeitus pradinėms sąlygoms ši sąsaja pasidarė būtina.

Atlikta analizė ir visi projektavimo darbai, kuriais remiantis galima realizuoti „MagicDraw UML“ paketo įskiepį bei modeliavimo aplinką, kuri reikalinga algoritmo funkcionalumui užtikrinti.

### 3.12. Projektavimo išvados

1. Suprojektuotas „MagicDraw UML“ paketo funkcionalumą išplečiantis įskiepis. Kuris pagal projektus bus realizuojamas „Java“ programavimo kalba. Pagal projektus įskiepis bus naudojamas kaip papildomas paketo funkcionalumas

2. Suprojektuota veiklos žinių saugyklos realizacija atsižvelgiant į projektavimo metu sudarytus ISK pasiūlytos veiklos žinių saugyklos pakeitimus, papildymus (optimizuota veiklos žinių saugykla). Suprojektuota saugykla gali būti realizuota įvairiose DB platformose. Projekte numatyta, kad saugykla bus realizuojama „Microsoft SQL Server 2005“ DB serveryje.

3. Suprojektuotas mechanizmas klasių meta-modelio išplėtimui trūkstamais komponentais. Šiuo tikslu naudojamas paketo profiliavimo mechanizmas, kuris leidžia funkcionalumą išplėsti bei apriboti profilio pagalba.

4. Suprojektuotas „MagicDraw UML“ paketo profilis klasių meta-modelio išplėtimui (klasių modelis išplėstas stereotipizuotomis klasėmis) bei atlikti projektavimai modeliavimo aplinkos sudarymui. Sudaryta aplinka gali būti naudojama stacionariai toje vietoje, kur buvo sudaryta ar perkeliama į kitą kompiuterį, kartu su profiliu. T.y. kaip ir profilis, modeliavimo aplinka yra laisvai perkeliama.

5. Remiantis naujai užsibrėžtu funkcionalumu ir pasikeitusiu veiklos žinių saugyklos modeliu suprojektuotas optimizuotas klasių modelio sudarymo pagal veiklos žinias algoritmas.

6. Suprojektuota įskiepio vartotojo sąsaja, leidžianti pasirinkti norimą veiklos žinių saugyklą bei suteikianti galimybę įvesti klasių modelio sudarymo nustatymus.

# 4. METODO REALIZACIJA BEI EKSPERIMENTAS

## 4.1.Realizacijos uždaviniai

Atlikus analizę ir projektavimo darbus užsibrėžti uždaviniai realizacijai yra:

1. Klasių modelio generavimo/sudarymo aplinkos (profilio bei modelio) realizacija;
2. Veiklos žinių bazės realizacija;
3. Algoritmo – „MagicDraw UML“ paketo įskiepio prototipo (vartotojo sąsajos bei funkcionalumo) realizacija;
4. Realizuoto algoritmo prototipo bandymai – eksperimentas (eksperimentinio pavyzdžio pagrindu);

Užsibrėžti uždaviniai realizuojami pagal atliktus projektavimus. Sekančiuose skyriuose detaliai aprašyta realizacijos eiga, gauti rezultatai ir apibendrinimai.

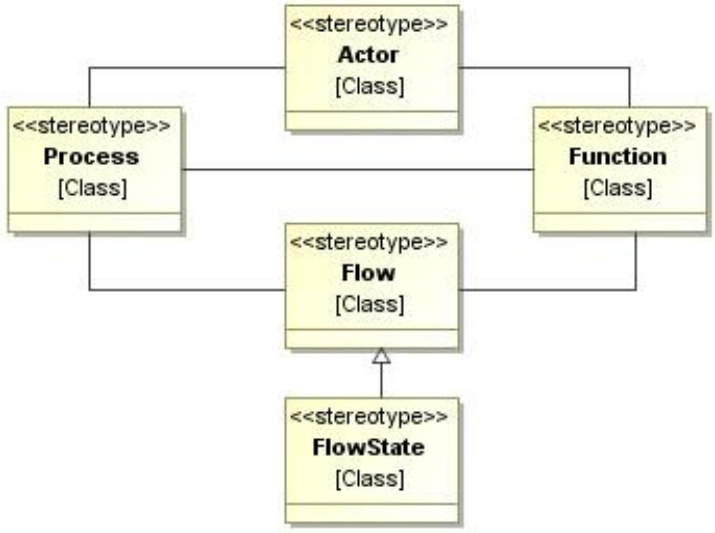
## 4.2.Profilio realizacija

Profilio pagalba išplečiamas paketo „MagicDraw UML“ funkcionalumas. T.y. sukuriami klasės stereotipai (stereotipizuotos klasės), kurie bus naudojami naujajame klasių modelyje. Šių klasių stereotipų egzemplioriai bus generuojami algoritmo pagal veiklos žinių bazę.

Profilio realizaciją galima išskaidyti į dvi dalis:

- Stereotipų aibės sudarymas;
- Sudarytų stereotipų parametrų ir apribojimų nustatymas.

Stereotipų aibės sudarymas tai visų galimų stereotipų (stereotipizuotų objektų) sudarymas. Konkrečiai nurodomi visi galimi išplėsto klasių modelio elementai (pav. 28).

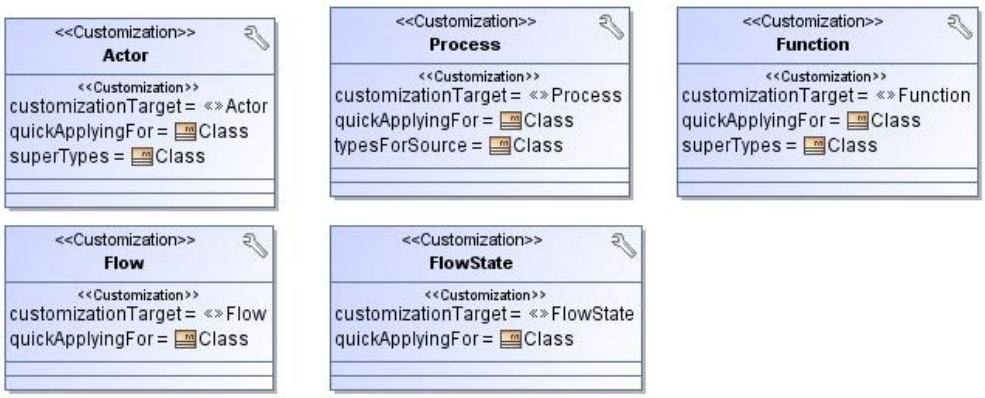



Pav. 28 Stereotipų aibė

Iš paveikslėlio (pav. 28) matoma, kad naudojami visi planuoti klasės stereotipai (aktoriaus tipo, funkcijos tipo, proceso tipo, srauto tipo bei srauto būsenos tipo). Visi stereotipai taikomi klasės tipo meta-klasės elementui. Taip rezultatas – visi gaunami elementai yra stereotipizuotos klasės.

Tarp stereotipų nurodomi ryšiai. Taip nurodoma kokie objektai galės būti susieti. T.y. apibrėžiami ryšiai tarp stereotipizuota klasių. Tai svarbu, nes toks stereotipas, kaip srauto būsena, gali būti jungiamas tik su srauto stereotipu ir jis gali būti tik srauto dalimi. Tokiu būdu nurodomi tam tikri apribojimai.

Kiti apribojimai nurodomi antrojoje profilio realizavimo dalyje (pav. 29).




Pav. 29 Stereotipų parametrai ir apribojimai

Šie nustatymai (nustatymų modelis „customization“) skirti nurodyti detalę stereotipų elgseną. Nurodomi pagrindiniai pradiniai parametrai. Visiems stereotipams bendras nustatymas – greitojo priskyrimo nustatymas. Tai reiškia, kad nurodomi nustatymai stereotipų susiejimui su nestereotipizuotomis klasėmis, kai yra naudojamas šis profilis (reikšmė – „quickApplyingFor“).

Kiti nustatymai nėra keičiami, kas reiškia, kad stereotipizuotos klasės savo parametrais bus analogiškos nestereotipizuotoms klasėms. T.y. jų atributų aibės bei galimų nustatymų aibės yra tokios pat. Tačiau profilį galima bet kada keisti. Iš to seka, kad, esant reikalui, galima nurodyti reikiamus apribojimus ar įtraukti naujus elementus.

### 4.3.Išplėsto klasių modelio realizacija (modeliavimo aplinka)

Realizuoto profilio patogesniam panaudojimui reikalinga modeliavimo aplinka. Reikia sudaryti naują modelį. Tai leis iš pasirinkimo meniu pasirinkti naują modelį, kuris naudos profilio elementus. Taip vartotojui bus sukurti nauji grafinės sąsajos elementai, tarp kurių bus matomi profilio stereotipizuotos klasės ir galimi ryšiai.

Analizės metu nustatyta, kad algoritmo realizacijai šis modelis reikalingas, nes vartotojai galės rankiniu būdu keisti sugeneruotą klasių diagramą. Taip pat bus galimybė sudarinėti klasių diagramą neautomatizuotu būdu.

Tačiau ši modeliavimo aplinka nėra būtina. T.y. yra galimybė turėti tokį pat funkcionalumą standartinėje klasių diagramos sudarymo aplinkoje. Tuo atveju reikia kurti klasės egzempliorius ir pritaikyti jiems stereotipus. Tada, neautomatizuotai keičiant ar sudarant klasių diagramą, darbas tampa labai neefektyvus (darbas užtrunka ilgiau).

Aplinkos realizacija atliekama pagal „No Magic“ pateiktą metodologiją. Sudarymo procesas yra paprastas ir lengvai suprantamas vartotojui. Taip pat, kaip sudarytas profilis, modeliavimo aplinka gali

būti naudojama tiek toje vietoje, kur buvo sudaryta, tiek perkelta į kitą kompiuterį . T.y. realizuota modeliavimo aplinka gali būti perkeliama kartu su profiliu. Tokia realizacija turi privalumų, nes kiekvienam vartotojui nereikės individualiai, pagal pateiktą profilį, kurti savo modeliavimo aplinkos. Taip palengvinamas darbas mažiau patyrusiems vartotojams.

Algoritmas suteikia galimybė automatizuotai gauti klasių diagramą. Tačiau norint ją papildyti ar pakeisti tenka atlikti tam tikras operacijas, nesusijusias su modeliavimo procesu (aplinkos ir profilio įtraukimas).

## 4.4.Duomenų bazės realizacija

### 4.4.1. Fizinės duomenų bazės sudarymas

Projektavimo metu buvo sudaryta pakeista veiklos modelio saugyklos loginė duomenų bazės schema (priedas Nr. 2). Naudojantis šia schema atliekamas fizinės duomenų bazės sudarymas.

Duomenų bazės realizacija atliekama automatizuotai. Panaudojant automatizuotas priemones, sudaryta loginė duomenų bazės schema naudojama fizinės duomenų bazės sudarymui. Tačiau sudarymas nėra visiškai tiesioginis. Šiuo atveju pagal loginę schemą yra sudaromas kodas, kurį įvykdžius duomenų bazės serveryje (sutrasavus kodą), sukuriama veiklos žinių bazės (duomenų bazė), su visom lentelėm ir visais ryšiais, egzempliorius.


Pav. 30 Fizinė DB realizacijos metodas

Šis sugeneruotas kodas suteikia galimybę veiklos žinių bazę sukurti ant skirtingų platformų duomenų basių serverių. Taip realizuojamas planuotas universalumas, nes veiklos žinių bazė gali būti realizuota bet kokiam nutolusiame serveryje.

Galima atlikti fizinės bazės realizaciją ir ne automatizuotu būdu. Tačiau tokiu atveju realizacijos procesas nebus optimalus. Taip realizacijos laikas bus didesnis bei atsiranda didesnė klaidos įvedimo tikimybė. Atliekant realizacija automatizuotai, generuojamas kodas (skriptas) visiškai atitinka loginę duomenų bazės schemą. Atliekant neautomatizuotą realizaciją, vartotojas pats sudarinėja duomenų bazę pagal turimą loginę schemą. Norint išvengti klaidų, reikalinga detali patikra, kurios metu bus nustatyta ar nėra netikslumų ir ar visi ryšiai bei duomenų bazės lentelės yra korektiškai sudarytos.

Fizinė duomenų bazė realizuojama „Microsoft SQL server 2005” duomenų bazės serveryje. T.y. „Microsoft Windows“ platformoje. Prie šios duomenų bazės galima prisijungti iš įvairių platformų ar sistemų. Susijungimui tereikia nurodyti tinkamą susijungimo informaciją.

### 4.4.2. Fizinės duomenų bazės pildymas

Sudaryta fizinė veiklos žinių bazė yra tuščia. Tačiau sudaryto algoritmo testavimui reikalingi pradiniai duomenys. Tuo tikslu reikia, kad veiklos žinių bazėje būtų bent vienas modelis. Modelis turi būti surašytas korektiškai.

Optimalus variantas yra pildyti veiklos žinių bazę automatizuotai. Šiuo metu nėra priemonių, kuriomis galima būtu užpildyti realizuotą saugyklą. Nei atliekant analizę, nei projektuojant nebuvo numatytas sprendimas veiklos žinių bazės automatizuotam užpildymui. Tačiau tolimesniam darbui privaloma, kad bazėje būtu bent vienas modelis.

Sprendimas – atlikti neautomatizuotą lentelių užpildymą. T.y. nėra apsibrėžtų reikalavimų duomenis pildyti automatizuotai. Tačiau toks pildymo procesas nėra optimalus dėl didelio pildymo laiko bei klaidos įvedimo tikimybės padidėjimo.

Nepriklausomai nuo to ar veiklos žinių bazė yra pildoma automatizuotai, ar neautomatizuotai, pildymo būdas yra tas pats. Kadangi veiklos žinių bazė yra standartinė duomenų bazė, joje atliekamos operacijos SQL komandų dėka. T.y. informacijos užpildymas atliekamas standartinėmis informacijos įterpimo komandomis pvz.:

```
INSERT INTO <lentelės pavadinimas>
       (<laukas1>, ..., <laukasn>)
VALUES (<reikšmė1>, ..., <reikšmėn>)
```

Svarbu paminėti, kad reikšmių kiekis privalo atitikti laukų kiekį. Tačiau nėra privaloma išvardyti visus lentelės laukus (tik privalomus). Nenurodyti laukai nebus pildomi.

Taip pat svarbu lentelių pildymo eiliškumas. Jei lentelės siejasi, privaloma pildyti lenteles atitinkama tvarka:

1. Pirma pildoma lentelė į kurią yra nuoroda iš kitos lentelės;
2. Pildoma lentelė, kurioje nurodoma nuoroda į pirmame žingsnyje sukurtą įrašą.

Remiantis šiomis taisyklėmis pildoma veiklos žinių bazė. Pasirinktas modelis yra skyriuje 3.6 aprašytas veiklos modelis. Tad nurodyto modelio surašymui į veiklos žinių bazę sudarytas SQL pildymo komandų rinkinys (priedas Nr. 4).

Įrašytas veiklos modelis nėra pilnas. T.y. nėra užpildytas loginis tikrinimas (loginis sprendimas). Ši informacija aprašoma veiklos taisyklėmis. Tačiau klasių modelio generavimo algoritmui ši informacija nėra reikalinga. Todėl nėra privaloma šią informaciją surašinėti į duomenų bazę.

### 4.5.Algoritmo realizacija

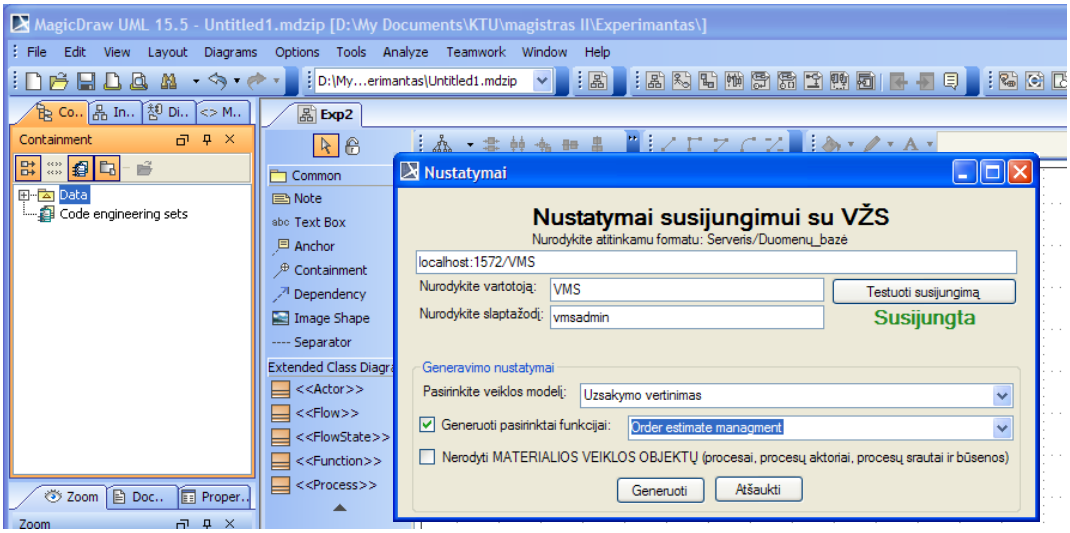
Realizuotas klasių modelio generavimo pagal veiklos žinių bazę algoritmas yra „MagicDraw UML“ paketo įskiepis. Sudarytas įskiepis susideda iš vartotojui matomos ir nematomos realizacijos dalies. Matomą realizacijos dalį sudaro:

- Vartotojo sąsaja pradiniu parametrų įvedimui;
- Rezultatai – sugeneruotas klasių modelis (ar diagrama).

Tuo tarpu nematoma vartotojui realizacijos dalis yra įskiepio funkcionalumas. T.y. klasių modelio generavimo algoritmas bei susijungimo su duomenų baze funkcionalumas. Detaliau realizacija aprašoma sekančiuose skyriuose.

### 4.5.1. Vartotojo sąsajos realizacija

Įskiepio vartotojo sąsaja yra vienas iš dviejų vartotojui matomos realizacijos komponentų. Šioje sąsajoje realizuotas funkcionalumas yra minimalus. Realizuojama tik pradinių nustatymų įvedimas. Todėl sąsają sudaro duomenų įvedimo, pasirinkimo bei patvirtinimo elementai (pav. 31).




Pav. 31 Įskiepio vartotojo sąsajos realizacija

Vartotojo sąsajoje reikia nurodyti susijungimo su veiklos žinių saugykla (VŽS) informaciją. T.y. nurodomas serveris ir vartotojo autentifikavimo informacija (vartotojo vardas bei slaptažodis). Nurodžius susijungimo informaciją atliekamas susijungimo patikrinimas. Ir esant sėkmingam susijungimui leidžiami tolimesni nustatymai.

Jei susijungimas buvo sėkmingas, vartotojui pateikiama generavimo nustatymų informacija. Čia vartotojas gali pasirinkti pagal kurį projektą bus generuojamas klasių modelis (klasių diagrama). Taip pat galima nurodyti, kad nebūtų generuojami materialios veiklos objektai (skyrius 3.9).

Realizacija atliekama standartinėmis „Java“ grafinių komponentų sudarymo priemonėmis.

### 4.5.2. Susijungimo su duomenų baze realizacija

Duomenų bazė (veiklos žinių bazė) gali būti realizuota įvairiuose duomenų bazės serveriuose (šiuo atveju „Microsoft SQL Server 2005“). Susijungimui su serveriu ir duomenų baze turi būti naudojama atitinkama tvarkyklė (skyrius 3.3).

Susijungimas su veiklos žinių baze bei visos duomenų operacijos (gavimas, įterpimas, keitimas, trynimas) vykdomos tvarkyklės dėka. Tvarkyklė, šiuo atveju – papildoma biblioteka, kurioje yra aprašytos bei realizuotos duomenų bazės operacijos.

„MagicDraw UML“ paketo įskiepis programuojamas „Java“ programavimo kalba. Tad tvarkyklė turi būti skirta šiai programavimo kalbai. Kadangi veiklos žinių bazė yra realizuota „Microsoft SQL Server 2005“ serveryje, naudojama oficiali „Microsoft“ išleista tvarkyklė, skirta „Java“ programas susieti su šio tipo duomenų bazės serveriu. Tvarkyklės naudojimas vaizduojamas modelyje (pav. 32).


Pav. 32 DB severio tvarkyklės naudojimas

Naudojant duomenų bazės serverio tvarkyklę komunikavimas vykdomas pagal standartines tvarkyklės operacijas. T.y. bendravimui naudojami specifiniai tvarkyklės elementai. Komunikavimas vykdomas standartinių SQL komandų pagalba. Taip užklausa ir atsakymas (pav. 32) yra standartinės formos. Užklausa yra bet kokia komanda (įtraukimo, gavimo, keitimo, trynimo) suformuota programos – tvarkyklei, o atsakymas – standartinės formos tvarkyklės pateikiamas atsakymas iš duomenų bazės. Operacijos pačioje tvarkyklėje, suformuotos užklausos ([užklausa2]) bei atsakymo ([atsakymas2]) (pav. 32) formos nėra žinomos. Bet ši informacija nėra reikalinga.

### 4.5.3. Klasių generavimo realizacija

Kaip ir susijungimas su veiklos žinių baze, klasių generavimo realizacija taip pat yra vartotojui nematoma. Pačio funkcionalumo vartotojas nemato, o matomas tik rezultatas - sugeneruotas klasių modelis (diagrama).

Realizuojamas algoritmas pagal atliktus projektavimo darbus. T.y. realizuojamas algoritmas atitinka naujus reikalavimus.

Realizuojamas visas planuotas funkcionalumas (pav. 23, pav. 24, pav. 25). Siekiant palengvinti realizacijos procesą, tam tikri žingsniai yra išskaidomi į smulkesnes funkcijas. T.y. vieną algoritmo žingsnį atitinka kelių funkcijų rinkinys. Funkcijos yra daugkartinio panaudojimo, tad tam tikros funkcijos pakartotinai naudojamos keliuose algoritmo žingsniuose. Tai algoritmo kodui suteikia glaustumą.

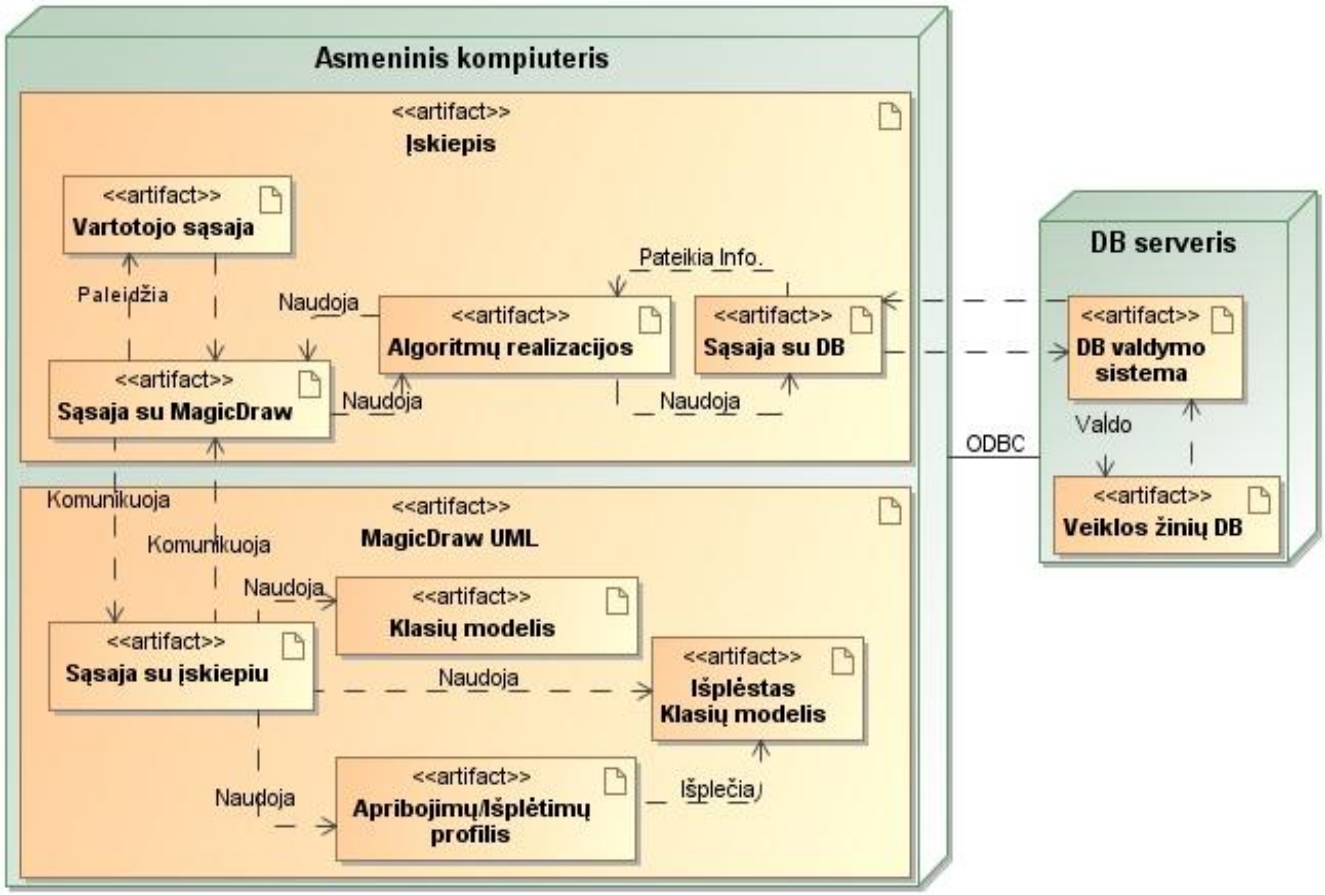
Šiuo metu nėra realizuota galimybė susijungti su įvairių tipų duomenų bazėmis. Galimas susijungimas tik su „Microsoft SQL Server 2005“ duomenų bazės serveriu, nes naudojama tik šio serverio tvarkyklė. Esant reikalui, įskiepį galima papildyti kitomis tvarkyklėmis. Tuo atveju teks realizuoti ir bendravimo su atitinkamais serveriai funkcionalumą.

Algoritme realizuotas klasių generavimas. Pagal veiklos žinių bazę sugeneruojama klasių aibė. Taip pat sugeneruotos klasės atvaizduojamos grafiškai. Tačiau klasės atvaizduojamos vienoje vietoje. Vartotojui, norint pamatyti informatyvų modelį, reiki atitinkamai išstumdyti klases (atitinkamai išdėstyti lange).

## 4.6.Realizuoto prototipo diegimas

Realizuotas prototipas – artefaktas. Jis yra realizuojamas komponentais pagal projektavime pateiktą schemą (pav. 15). T.y. komponentų išdėstymas bei jų aibė nesikeičia.

Realizuoti artefaktai įdiegiami į sistemą, kurioje bus atliekami jų tolimesni testavimai (eksperimentas). Diegimo struktūra pateikiama paveikslėlyje pav. 33.




Pav. 33 Įskiepio, CASE sistemos išplėtimui, diegimo modelis (schema)

Šioje schemoje pateikiami esami ir naujai – realizuoti artefaktai. Nauji artefaktai yra įskiepis, apribojimų/išplėtimų profilis ir veiklos žinių bazė. Pirmieji du privalo būti įdiegti vartotojo asmeniniame kompiuteryje. Veiklos žinių bazė gali būti įdiegta į bet kokį DB serverį, kuriame DB platforma yra „Microsoft SQL Server 2005“.

Tiek vartotojo, tiek ir DB serverio kompiuteriai privalo atitikti reikalavimus. Pagrindinis reikalavimas – privaloma programinė įranga (pav. 34). Ši įranga turi būti įdiegta kompiuteriuose prieš realizuoto prototipo diegimą ir naudojimą.

**Vartotojo kompiuteris**

**DB Serveris**


Pav. 34 Privalomos programos vartotojo ir serverio kompiuteriuose

Pagrindinė atliekama diegimo operacija yra įskiepio patalpinimas į standartinę įskiepiu rinkmeną (katalogą). Svarbu nurodyti įskiepio deskriptorių. Priešingu atveju šis artefaktas nebus aptinkamas „MagicDraw UML“ paketo.

Kitas įdiegiamas artefaktas – apribojimų/išplėtimų profilis. Šis profilis gali būti bet kurioje vietoje. Tačiau jis privalo būti įtrauktas į naujai kuriamą projektą. Neįtraukus artefakto, nebus vykdomas klasių modelio sudarymas pagal veiklos žinias.

Veiklos žinių bazė – artefaktas, diegiamas duomenų bazės serveryje. Rekomenduojama prijungti prieduose pateikiamą realizuotą fizinę DB. Joje jau yra sudarytos visos lentelės ir operacijų procedūros. Tokiu būdu vartotojas, prijungęs pateiktą DB, turės funkcionuojančią veiklos žinių saugyklą.

Išplėstas klasių modelis nėra privalomas. T.y. jis tik palengvina neautomatizuotą klasių modelio sudarymą. Šis modelis meniu laukuose pateikia klasių stereotipus ir jų komponentus (ryšius ir t.t.). Taip vartotojui palengvinamas neautomatizuotas procesas.

Artefaktų diegimas aprašomas tam tikrais algoritmais (žingsnių seka).

Įskiepio įdiegimui užtenka pateiktas bylas („ECMgenByEM“ katalogą) įkelti į katalogą „plugins“, esantį įdiegto „MagicDraw UML“ paketo kataloge („...\MagicDraw UML“). Standartinio diegimo „MS Windows“ sistemoje atveju, įskiepio katalogas turi būti patalpintas atitinkamame kataloge, adresu - C:\Program Files\MagicDraw UML\plugins\eclipse\plugins.

Profilis diegiamas pagal tam tikrą žingsnių seką („MagicDraw UML“ versija 15.5):

1. Paleidžiamas „MagicDraw UML“ paketas.
2. Sukuriamas naujas projektas: File->New Project.
3. Sukurtame projekte atidaromas nustatymų langas: Options->Modules.
4. Atsidariusiame lange nurodome profilį. Ir profilis įtraukiamas į projektą.

Veiklos žinių bazė gali būti įdiegta prijungiant pateiktą (prieduose) veiklos žinių bazę prie DB serverio.

Įskiepis savyje turi visą reikiamą informaciją (visas bibliotekas) skirtą funkcionalumo realizacijai, įskaitant susijungimui su veiklos žinių baze reikalingas bibliotekas. Taip minimizuojamas visų diegimo operacijų kiekis. Įdiegus pateiktus tris artefaktus (įskiepį, profilį, VŽ saugyklą), vartotojas tūrės funkcionuojančią CASE sistemą, kuri suteiks galimybę sudarinėti klasių modelius pagal veiklos žinias.

### 4.7.Vartotojo vadovas

Prototipas, skirtas klasių modelio sudarymui pagal veiklos žinias, gali būti naudojamas tik įdiegus visus privalomus komponentus (skyrius: 4.6 Realizuoto prototipo diegimas). Sudarytas įskiepis gali būti naudojamas tik „MagicDraw UML“ pakete.

Prognozuojama, kad tikslinė vartotojų grupė yra IS projektuotojai, kurie turi aukšto lygio kompiuterinį raštingumą bei patirties, darbui su nurodytu paketu. Tokiu būdu, nurodytos diegimo ir darbo su įskiepiu operacijos yra aiškios.

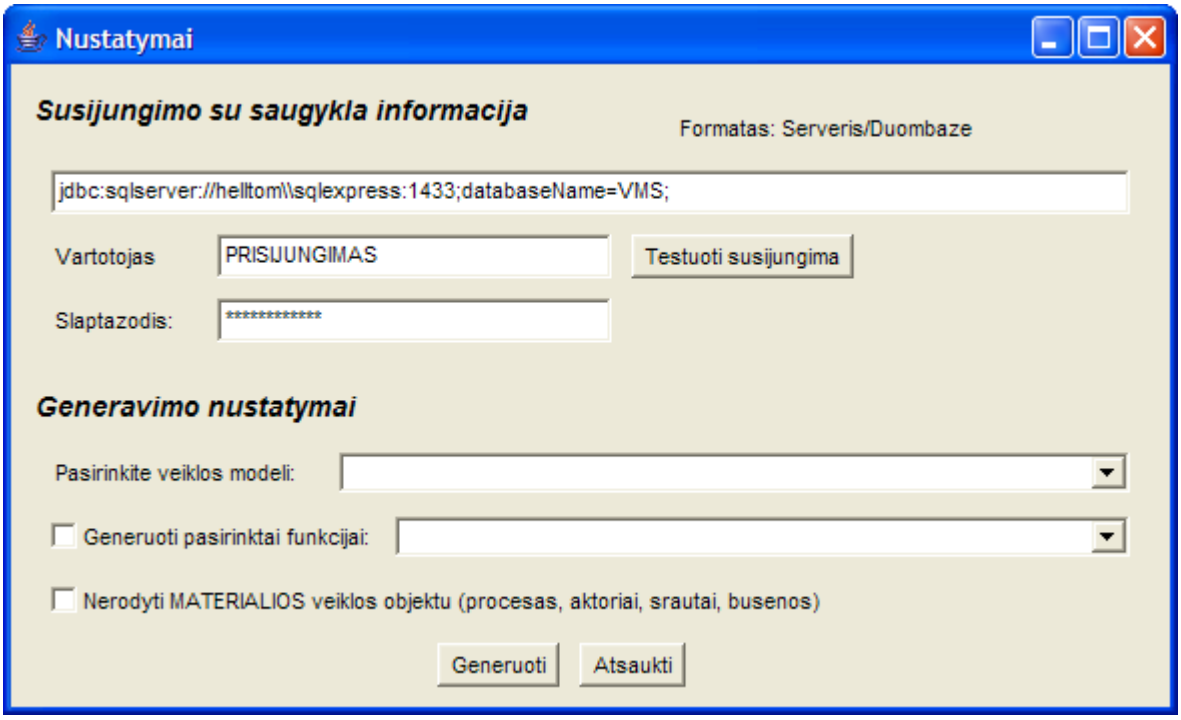
Norint sudaryti klasių modelį reikia atlikti atitinkamą žingsnių seką:

**Žingsnis 1.** Paleidžiamas paketas ir jame sukuriamas arba atidaromas projektas.

**Žingsnis 2.** Projekte reikia susikurti naują klasių modelį (Diagrams->Class Diagrams...).

**Žingsnis 3.** Jei modelyje nėra profilio, įtraukiamas profilis (Options->Modules, atsidariusiame lange nurodomas profilis).

**Žingsnis 4.** Sudarius naują modelį kreipiamasi į įskiepį (File->My Action). Vartotojui pateikiama įskiepio vartotojo sąsaja (pav. 35).



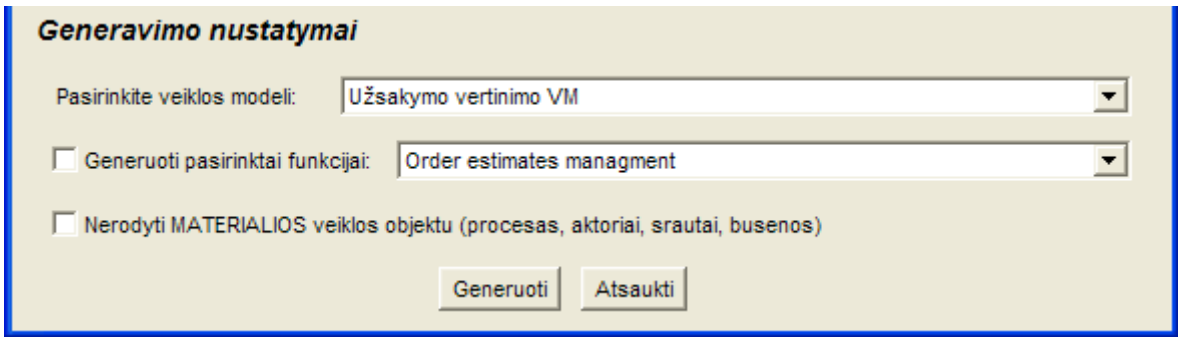

Pav. 35 Įskiepio vartotojo sąsaja

**Žingsnis 5.** Sąsajos lange privaloma nurodyti susijungimo su veiklos žinių baze informaciją. Saugyklos nurodymo eilutėje rekomenduojama keisti atitinkamas vietas. T.y. nerekomanduojama eilutės rašyti pačiam. Vartotojui reikia nurodyti serverio kompiuterio vardą ar adresą, DB serverio vardą kompiuteryje ir pačios veiklos žinių bazės pavadinimą. Susijungimo „porto“ galima nenurodyti arba galima palikti nurodytą.

Lauke „Vartotojas“ – nurodomas vartotojo, kuris jungiasi atitinkamu slaptažodžiu (laukas „Slaptažodis“) vardas. Tai vartotojo autentifikavimui bei atitinkamų teisiu serveryje suteikimui (autorizavimui) skirta informacija.

Nurodžius šią informaciją spaudžiamas susijungimo testavimo mygtukas („Testuoti susijungimą“).

**Žingsnis 6.** Sėkmingo susijungimo atveju, vartotojui pateikiama serveryje saugoma informacija (veiklos modelių pasirinkimas, modelių funkcijų pasirinkimas). Informacijos pateikimas vaizduojamas paveikslėlyje 36.




Pav. 36 Sėkmingo susijungimo atveju pateikiami generavimo nustatymai.

Vartotojas privalo pasirinkti veiklos modelį (standartiškai parenkamas pirmas). Kiti nustatymai nėra privalomi. Tačiau vartotojui suteikiama galimybė generuoti klasių modelį pasirinktai modelio

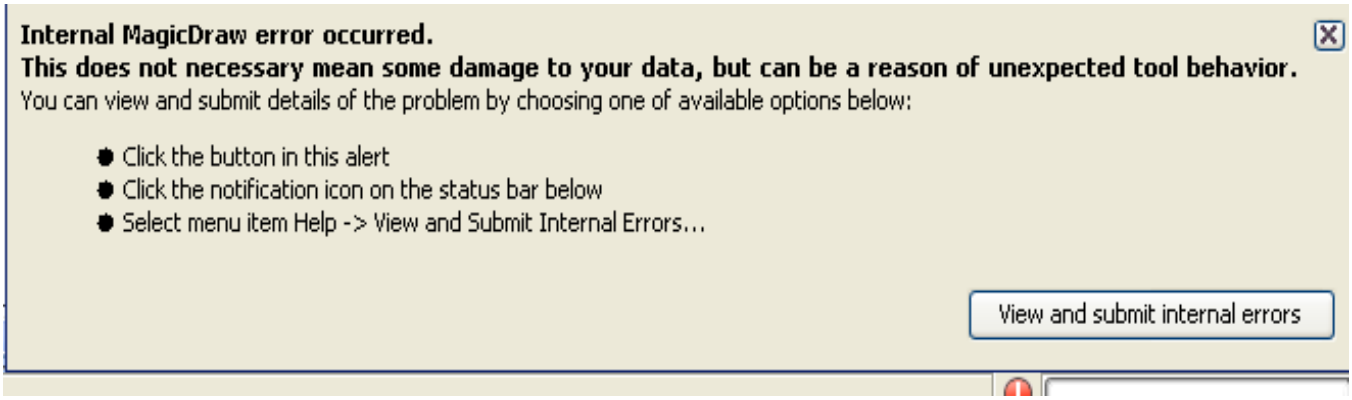
funkcijai (uždedama varnelė – „Generuoti pasirinktai funkcijai“ ir iš iškrentančio meniu parenkama valdančioji funkcija) ir galimybė nevaizduoti materialios veiklos objektų (uždedama varnelė – „Nerodyti MATERIALIOS veiklos objektų...“).

**Žingsnis 7.** Įvedus nustatymus spaudžiamas mygtukas „Generuoti“. Norint atsisakyti operacijos, spaudžiamas mygtukas „Atsaukti“.

Griežtai nerekomenduojamas sąsajos išjungimas, naudojant išjungimo mygtuką. Šis mygtukas išjungia ne tik sąsają, bet ir patį „MagicDraw UML“ paketą.

### 4.7.1. Įspėjimai ir klaidos

Tam tikrais atvejais (nesėkmingas susijungimas su serveriu ir t.t.) vartotojui pateikiamas klaidos pranešimas. Pranešime nurodoma, kad projekto informacija nėra sugadinta, tačiau įskiepis negalėjo atlikti tam tikrų operacijų (pav. 37).




Pav. 37 „MagicDraw UML“ įskiepio klaidos pranešimas

Klaidos pranešimas yra bendras visoms situacijoms. Jis gali būti pateikiamas esant skirtingoms aplinkybėms. Pagrindinė aplinkybė – nesėkmingas susijungimas su serveriu.

Kadangi tai tik metodo prototipas, jame nėra numatyta visos galimos ekstremalios situacijos. Dėl šios priežasties pranešimas gali būti pateiktas ir atliekant kitas operacijas. Pvz.: generavimo metu gauti netinami duomenys, duomenys visai negauti, nutrūko ryšys su DB ir t.t.

Jei pranešimas pateikiamas generavimo metu, rekomenduojama modelio generavimą pradėti iš naujo. Esant pakartotinam klaidos pranešimui, rekomenduotina patikrinti veiklos žinių bazės įrašus.

## 4.8.Eksperimentas

### 4.8.1. Eksperimento tikslas ir kriterijai

Pagal atliktus analizės, projektavimo bei realizacijos darbus atliekamas realizuoto „MagicDraw UML“ paketo įskiepio, skirto klasių modelio generavimui pagal veiklos žinių bazę, prototipo bandymas (eksperimentas).

Eksperimentas vykdomas eksperimentinio pavyzdžio metodu. T.y. paimamas konkretūs pavyzdiniai duomenys, kurie yra naudojami realizuoto prototipo testavimui.

Eksperimento metu tikimasi nustatyti, ar prototipas veikia korektiškai. Tai nustatoma palyginus planuojamus gauti rezultatus su gautais. Jei jie atitiks, bus daroma prielaida, kad algoritmas veikia korektiškai.

Tikimasi nustatyti našumo parametrus. T.y. bus nustatomas algoritmo našumas (projektavimo proceso efektyvumo pagerėjimas/pablogėjimas).

Eksperimentas nėra pilnai objektyvus, nes atliekamas su konkrečiais pavyzdiniais duomenimis, vykdant automatizuota bei neautomatizuotą klasių modelio sudarymą. Eksperimento metu vertinami kriterijai yra subjektyvus, nes priklauso nuo skirtingų aplinkybių. Pavyzdžiui neautomatizuoto proceso charakteristikos priklauso nuo vartotojo kompetencijos, bendros emocinės būklės. Tačiau eksperimento tikslas yra nustatyti automatizuoto metodo efektyvumą, korektiškumą bei bendrą IS kūrimo proceso pagerėjimą.

Bendrai eksperimento vykdymo kriterijai, kriterijų aprašymas ir vertinimo sistema yra pateikiami lentelėje (lent. 8).

Lentelė 8 Eksperimento kriterijai

| Kriterijus | Aprašymas | Vertinimas |
|---|---|---|
| **Korektiškumas** | Eksperimento metu nustatoma, ar realizuotas įskiepis veikia pagal apibrėžtus reikalavimus (algoritmas nustatytas projektavime). | Eksperimento metu gauti rezultatai sutikrinami su analizėje sudarytais teoriniai rezultatais. Nustatomas atitikimas (%). |
| **Projektavimo kokybės pagerėjimas (sparta)** | Eksperimento metu nustatoma ar realizuotas įskiepis pagerina projektavimo kokybę, konkrečiai projektavimo proceso greitaveiką. | Lyginama automatizuoto ir neautomatizuoto projektavimo greitaveika. Palyginimas (%). |
| **Modelio suprantamumas** | Nustatoma, ar gautas sugeneruotas modelis yra taip pat suprantamas, kaip ir neautomatizuotai sudarytas klasių modelis (standartinis). | Nustatomas stereotipų suprantamumas, gauto modelio bendras suprantamumas. Įvertinimas balais. |
| **Modelio parengimas pristatymui** | Nustatoma, kiek laiko užtrunka galutinio, vartotojui suprantamo klasių modelio parengimas. | Nustatomas papildomas laikas, skirtas modelio vizualiniai daliai parengti. Laikas lyginamas su bendru neautomatizuoto proceso laiku. |
| **Įskiepio suprantamumas** | Eksperimento metu nustatomas ar realizuotas įskiepis yra patogus, t.y. ar naudojimas yra suprantamas vartotojui. | Balais (1-10) įvertinamas įskiepio naudojimosi suprantamumas. |

Eksperimentas vykdomas pagal apibrėžtus kriterijus. Tačiau, kaip buvo minėta, eksperimento rezultatai priklauso nuo pasirinktų pradinių duomenų ir tiriamų subjektų. Tad ir pats eksperimentas yra dalinai subjektyvus.

Kriterijus „sparta“ apibūdina proceso pagreitėjimą ar sulėtėjimą. Eksperimento metu nustatoma laiko trukmė reikalinga automatizuotam ir neautomatizuotam klasių modelio gavimui. Abu gauti laikai palyginami ir gaunamas proceso trukmės pagerėjimas ar pablogėjimas vertinant „%”. Automatizuoto proceso metu procesas skaidomas į tris dalis (pasiruošimo generavimui trukmė, generavimo trukmė, modelio paruošimo pristatymui trukmė). Tačiau bendras laikas gaunamas susumavus tris dedamąsias.

Modelio suprantamumas taip pat yra gan subjektyvi sąlyga. Konkrečiu atveju yra labai svarbi, nes lyginamas išplėstas ir neišplėstas klasių modeliai. Kaip apsibrėžta analizės dalyje generuojamas modelis nėra tikras klasių modelis (tai veiklos objektų modelis PIM lygmenyje). Siekiant šį modeli padaryti suprantamu sudarytos stereotipizuotos klasės. Tokiu būdu lengvai identifikuojami įvairių tipų veiklos objektai. Nepaisant to išlieka ir standartiniai (neišplėsto klasių modelio) elementai. Dėl šios priežasties svarbu įvertinti modelio suprantamumo (informatyvumo) kriterijus.

Modelio parengimas pristatymui kriterijus reikalingas įvertinti prototipo trūkumus. Kriterijus padės nustatyti poreikį tobulinti atitinkamus prototipo funkcionalumo elementus (atvaizdavimo mechanizmą).

Kriterijus – įskiepio suprantamumas, taip pat, kaip ir trukmės kriterijus, priklauso nuo tiriamojo subjekto (sistemos naudotojo). Siekiant optimizuoti rezultatus, tiriamas subjektas turi tinkama kompiuterinį raštingumą. Tokiu būdu daroma prielaida, kad rezultatai bus analogiški tiriant ir kitus subjektus.

Iš to seka, kad užsibrėžtų kriterijų pakanka nustatytiems eksperimento tikslams realizuoti.

### 4.8.2. Eksperimento eiga

Eksperimento metu atliekami realizuojamo klasių modelio generavimo pagal veiklos žinių bazę „MagicDraw UML“ paketo įskiepio testavimai. Antroji eksperimento dalis yra neautomatizuotas klasių modelio sudarymas pagal turimas veiklos žinias. Ši dalis skirta rezultatų palyginimui.

Tyrimas atliekamas naudojant šiuolaikiškas priemones (Intel Core 2 Duo 1,8GHz procesorius, 1Gb RAM, “Microsoft Windows XP Pro Sp3”, “MagicDraw UML 15.5”), kurios atitinka vidutinio lygio kompiuterines sistemas. Taip tikimasi gauti neiškreiptus rezultatus. Tikimasi, kad analogiškas sistemas naudoja IS projektuotojai.

Automatizuoto klasių modelio sudarymo metu atliekamos pagrindinės pasiruošimo operacijos. Į veiklos žinių bazę yra surašytas veiklos modelis (pav. 22), naudotas atliekant projektavimo darbus. SQL komandų pagalba pateiktas modelis surašytas į realizuotą veiklos žinių bazę (duomenų bazė „Microsoft SQL Server 2005“ serveryje). Detaliau duomenų bazės pildymas aprašytas skyriuje (skyrius 4.4.2).

Imituojant realias sąlygas, vykdomos standartinės projektavimo operacijos. „MagicDraw UML“ pakete sukuriamas naujas projektas. Į projektą įtraukiamas profilis, skirtas klasės stereotipizavimui (UML meta-modelio išplėtimui). Žinant, kad jau yra sudaryti veiklos modeliai (turima veiklos žinių ba-

zė) – CIM lygio modeliai, sudaromas naujas klasių modelis. T.y. sukuriamas naujas „Išplėstas klasių modelis“. Tai atlikus, kreipiamasi į klasių modelio generavimo įskiepį. Įvedus nustatymus paleidžiamas klasių modelio generavimas.

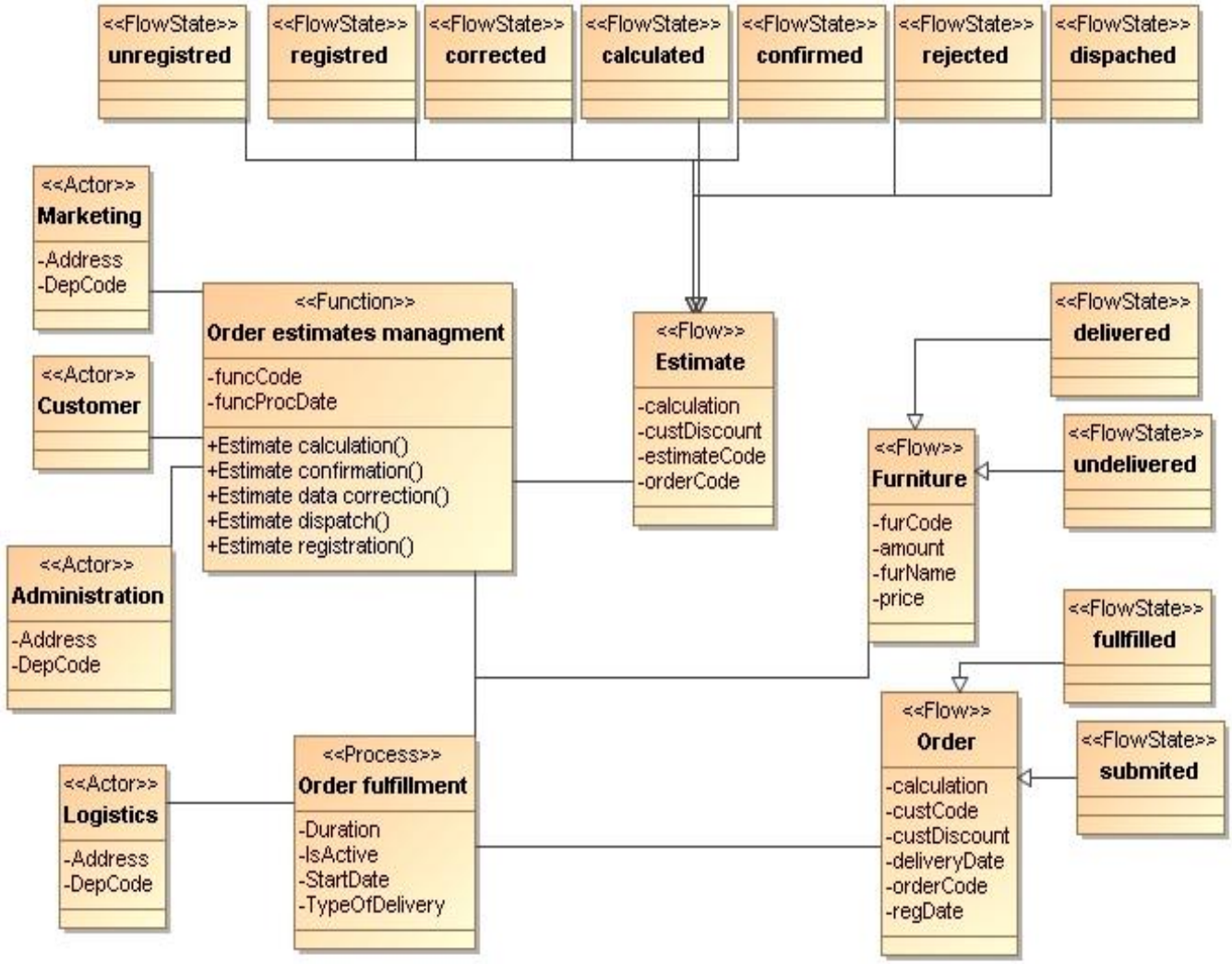
Sugeneruotas klasių modelis atvaizduojamas „MagicDraw UML“ paketo grafiniame pateikimo lange. Tačiau vartotojui tenka paruošti modelį, kad jis vizualiai būtų suprantamas visiems vartotojams. Taip vartotojui tenka paruošti modelį pristatymui (tinkamai išdėstyti komponentus), kad modelis būtu lengvai suprantamas.

Neautomatizuoto proceso metu subjektui (vartotojui) pateikiamas „MagicDraw UML“ paketas ir veiklos modelis. Vartotojas pagal pateiktą veiklos modelį sudaro klasių modeli. Eksperimento metu stebima proceso trukmė ir eiga.

Visa vykdymo eiga bei gauti rezultatai yra tiriami pagal apsibrėžtus eksperimento kriterijus.

### 4.8.3. Eksperimento rezultatai

Eksperimento metu gaunamas rezultatas yra „MagiwDraw UML“ pakete sugeneruotas klasių modelis pagal veiklos žinių bazę (pav. 38) bei analogiškas modelis, gautas neautomatizuotai. Šis rezultatas atitinka planuotus, užsibrėžtus reikalavimus.




Pav. 38 Sugeneruotas klasių modelis

Sugeneruotame klasių modelyje tiek elementų medyje, tiek prezentacijos lange (angl. prezentation level) gaunami visi reikiami elementai. Prezentacijos lange visi elementai randasi toje pačioje vietoje. Tad siekiant modelį padaryti suprantamu visiems vartotojams, elementai rankiniu būdu yra išstumdomi į atitinkamas prezentacijos lauko vietas.

Taip pat ne visi atributai ar ryšiai yra nurodyti (ne visi sudaryti arba nėra parinktas tinkamas). Atributai gali būti įtraukti rankiniu būdu. Atitinkami ryšiai (ryšio tipas) gali būti pakeičiami per nustatymo lauką. Gautas modelis yra gan tikslus, ir pakeitimų kiekis yra minimalus. Todėl atlikus visus grafinius pakeitimus, galutinis modelis pateikiamas paveikslėlyje (pav. 38).

### 4.8.4. Eksperimento išvados

Pagal gautus eksperimento rezultatus atliekamas eksperimento vertinimas. T.y. atliekamas apsibrėžtų kriterijų reikšmių nustatymas. Eksperimentas buvo atliekamas su pateiktu pavyzdiniu veiklos modeliu iš kurio gaunama 20 klasės egzempliorių. Šie rezultatai pateikiami lentelėje (lentelė 9).

Lentelė 9 Eksperimento kriterijų įvertinimas

| Nr. | Kriterijus | Vertinimas |
|---|---|---|
| **1.** | Korektiškumas | 95% atitikimas |
| **2.** | Sparta | 35% greičiau |
| **2.1** | Sparta – neautomatizuoto proceso | ~20min. |
| **2.2** | Sparta – automatizuoto proceso | ~7min |
| **3.** | Suprantamumas (modelio) | Pagerėjimas 4 balai. |
| **3.1** | Suprantamumas (neišplėsto klasių modelio) | 6/10 |
| **3.2** | Suprantamumas (išplėsto klasių modelio) | 10/10 |
| **4.** | Modelio parengimas pristatymui | 8/10 |
| **4.1** | Neautomatizuoto proceso | ~20min. |
| **4.2** | Automatizuoto proceso | ~4min |
| **5.** | Įskiepio suprantamumas | 7/10 |
| **6.** | Galimybė panaudoti modelį projektuojant | 5/10 |
| **6.1** | Galimybė panaudoti sugeneruotą modelį (PIM->PSM) | 8/10 |
| **6.2** | Galimybės naudoti algoritmą dėl neturimų VŽ saugyklų | 2/10 |

Eksperimentas buvo atliekamas su konkrečiu modeliu. Tad esant kitiems modeliams rezultatai gali skirtis. T.y. spartos bei korektiškumo rezultatai priklauso nuo esamo veiklos modelio bei projektuotojo darbo savybių. Tačiau tendencingai automatizuotas procesas bus spartesnis bei tikslesnis.

Įskiepio našumą labai mažina laikas skirtas modelio pristatymo parengimui. T.y. šis procesas išlieka neautomatizuotu. Vartotojui tenka pačiam išdėstyti sugeneruotas klases. Šis procesas užtrunka daugiau nei pusę viso modelio sudarymo laiko. Siekiant optimizuoti procesą, reikia automatizuoti ir modelio parengimo pristatymui vykdymą.

Planuojama, kad sparta eksponentiškai didės sudėtingėjant modeliams. T.y. paprastų modelių sparta gali būti didesnė net neautomatizuoto proceso, nes automatizuotam procesui tenka atlikti daug pasiruošimo darbų. Tačiau esant sudėtingiems modeliams automatizuotas procesas bus ženkliai spartesnis, nes automatizuotame procese daugiausiai laiko atliekamas modelio parengimas pristatymui. Kai kuriais atvejais šios operacijos net nereikia atlikti.

Kaip buvo minėta, automatizuoto klasių modelio generavimo atveju atliekama daug pasiruošimo darbų. Šis būdas reikalauja kiekvieną kartą kuriant naują projektą įtraukti išplėsto klasių modelio profilį. Rekomenduojama susikurti ar įtraukti modeliavimo aplinką. Šis procesas yra vienkartinis. Pasirengimo darbai yra paprasti. Privalomas darbas – nurodyti generavimo parametrus. Čia gali iškilti problemų su susijungimu su veiklos žinių baze. Nepavykus susijungti, generavimas nebus vykdomas. Šie pasirengimo darbai apsprendžia generavimo trukmę (šiuo atveju 3 min.). Likusią laiko dalį atliekamas modelio parengimas pristatymui.

Neautomatizuoto proceso atveju modelio parengimas pristatymui atliekamas nuolat. T.y. šis procesas ir yra modeliavimo procesas, nes vartotojas iš kart modelyje vaizduoja klases ir ryšius atitinkamose vietose siekiant, kad modelis būtų suprantamas.

Svarbu paminėti, kas išplėsti klasių modeliai yra lengviau suprantami nes klasės yra stereotipizuotos. Todėl lengviau išsisavinama, kur ir kokio tipo klasės yra sudarytos bei kaip jos siejamos. Materialios veiklos atskyrimas nuo informacinės veiklos taip pat pagerina šią modelio savybę.

Informacinės veiklos atskyrimas nuo materialios pagerina modelio savybes, leidžiančias atlikti (PIM->PSM) transformacijos operacijas. Taip atsiranda galimybės panaudoti modelį projektuojant kitus modelius (PSM lygio). Šiai charakteristikai neigiamą įtaką daro veiklos žinių bazės poreikis, nes nėra realizuotų priemonių veiklos žinių bazės pildymui. Eksperimento metu veiklos modelio surašymas į veiklos žinių bazę užtruko ~2 valandas. Tad šis laiko terminas yra ilgesnis nei neautomatizuotas klasių modelio generavimas. Tokiu atveju, neesant automatizuotoms priemonėms veiklos žinių bazės pildymui, klasių modelio generavimas veiklos žinių bazės pagrindu, tampa labai neefektyvus.

### 4.9. Realizacijos ir eksperimento apibendrinimas

Atlikta visų suprojektuotų elementų realizacija. T.y. realizuotas „MagicDraw UML“ paketo įskiepis skirtas klasių modelio generavimui veiklos žinių bazės pagrindu. Taip pat realizuota veiklos žinių bazė bei modeliavimo aplinka.

Realizuota veiklos žinių bazė „Microsoft SQL server 2005“ duomenų bazės serveryje. Toks sprendimas suteikia galimybę duomenų bazę turėti „Windows“ sistemoje. Tai suteikia paprasto administravimo galimybes.

Projektavime numatyta, kad duomenų bazė gali būti įvairiose sistemose. Tačiau realizuotas prototipas leidžia panaudoti tik „Microsoft SQL server 2005“ duomenų bazės serveryje laikomą veiklos žinių bazę. Norint išplėsti funkcionalumą, reikia papildyti susijungimo bei informacijos srautų tarp programos ir serverio metodus (algoritmo papildymas).

Realizuotas algoritmas generuoja klasių modelį. Tačiau gautą modelį reikėtų įvardyti kaip veiklos objektų modelį. Sugeneruotas modelis nėra pilnas PIM lygio klasių modelis. Šiuo atveju nėra sudaromi ryšiai tarp srautų. Kiti sudaryti srautų ryšiai yra asociacijos tipo ir nėra apibrėžti kardinalumais. Siekiant optimizuoti IS projektavimo procesą rekomenduotina realizuoti šiuos trūkstamus papildymus.

Realizavus visus komponentus buvo atliktas eksperimentas realizuoto įskiepio prototipo funkcionalumo efektyvumui bei korektiškumui nustatyti.

Eksperimentas parodė, kad realizuotas įskiepis veikia teisingai. Tai suteikia IS projektuotojui galimybę automatizuotai gauti klasių modelį pagal turimas veiklos žinias. Tačiau gaunamas modelis nėra galutinis ir įskiepio vartotojui tenka atlikti papildomus darbus. Nepaisant to, IS projektavimo proceso kokybė pagerėja, nes automatizuotai gauto klasių modelio sudarymo trukmė yra mažesnė. Taip pat pagerėja ir pačio modelio kokybė, nes sumažinama klaidos pasirodymo tikimybė.

### 4.10. Realizacijos išvados

1. Realizuotas „MagicDraw UML“ paketo funkcionalumą išplečiantis profilis (klasių modelio išplėtimui stereotipizuotomis klasėmis). Pagal realizuotą profilį sudaryta modeliavimo aplinka (naujas – išplėstas klasių modelis). Tai leidžia sudarinėti informatyvesnius klasės modelius (veiklos objektų atvaizdavimas).
2. „Microsoft SLQ Server 2005“ serveryje realizuota suprojektuota veiklos žinių bazė ir į ją sėkmingai surašytas eksperimentinis veiklos modelis. Bazė sėkmingai panaudota klasių modelio sudarymui. Toks DB realizavimas suteikia galimybę vartotojui patogiomis priemonėmis valdyti turimas žinias (DB administravimas).
3. Realizuotas „MagicDraw UML“ paketo įskiepio prototipas. Ši realizacija apima projekte numatytos vartotojo sąsajos bei algoritmo funkcionalumo realizaciją.
4. Atlikti bandymai (eksperimentas) parodė, kad algoritmo prototipo bei veiklos žinių bazės funkcionalumas atitinka projektuose numatytą funkcionalumą.

# 5. DARBO APIBENDRINIMAS IR IŠVADOS

## 5.1.Darbo apibendrinimas

Darbo metu įvykdyti visi numatyti uždaviniai. Taip pat yra apibrėžti nauji reikalavimai, kuriuos įvykužius galima optimizuoti gautą „MagicDraw UML“ paketo įskiepį.

Darbo metu atlikta išsami esamos situacijos analizė. Kurios metu nustatyta kaip veiklos modeliai yra taikomi IS kūrimo procese. Analizė parodė, kad yra daugybė grafinių IS projektavimo bei verslo procesų modeliavimo įrankių. Taip pat yra daug veiklos modeliavimo metodikų, siekių, priemonių, standartų, tačiau IS klasių modelio generavimas pagal veiklos žinias nėra realizuotas. Dauguma paketų yra perpildyti nereikalingu funkcionalumu. Dėl šių priežasčių, realizuotas įskiepis turi pagerinti IS projektavimo procesą.

Išanalizavus Informacinių sistemų katedros (ISK) ir kitus vykdomus tyrimus veiklos žiniomis grindžiamų IS kūrime, buvo pasirinktas pagrindas, tyrimo metu sudaryto algoritmo projektavimui. Algoritmas buvo pakeistas, pagal pasikeitusius reikalavimus (skyrius 3.8).

Taip pat pagal pasirinktą loginę veiklos žinių bazės schemą buvo sudaryta nauja veiklos žinių bazė. Pasiūlyta duomenų bazės schema buvo pakeista siekiant optimizuoti informacijos įrašų kiekį. T.y. likviduota tam tikra perteklinė informacija. Atlikti pakeitimai leidžia saugykloje laikyti daug veiklos modelių.

Išsamios „MagicDraw UML“ paketo analizės metu buvo išsiaiškintos paketo išplėtimo galimybės. Tai leido realizuoti algoritmą šiai modeliavimo sistemai. Klasių modelio generavimo algoritmo realizavimas, naudojant paketo įskiepį, leidžia apsibrėžtą funkcionalumą naudoti pasirinktinai.

Suprojektuotas ir realizuotas sistemos prototipas leidžia generuoti vieną iš svarbiausių IS modelių (klasių modelį). Tačiau šis modelis įvardijamas kaip veiklos objektų modelis. Šiuo atveju, klasių modelis turi operacijas, tad iš to seka, kad ir veiklos modelis turi operacijas. Taip jis ne visiškai atitinka veiklos modelio koncepciją.

Sudarytas metodo prototipas realizuoja automatizuotą perėjimą nuo CIM lygmens modelių prie PIM lygmens modelių. Perėjimas nėra visiškai pilnas. T.y. nors ir generuojamas vienas iš svarbiausių IS modelių, informacija nėra pakankama, kad būtų realizuotas pilnas PIM lygmens modelis. Šiuo atveju nėra generuojami ryšiai tarp srautų. Skyriuje (skyrius 2.7) aprašomas šios informacijos poreikis. Taip pat vien šio gauto modelio nepakaks PIM->PSM perėjimui.

Kitas trūkumas – nėra generuojami objektų ryšių kardinalumai. Šiuo atveju visi klasių egzemplioriai jungiami asociacijos ryšiu, nenaudojant kardinalumų. Taip prarandama tam tikra dalykinė informacija.

Realizuoto prototipo funkcionalumas platesnis nei buvo užsibrėžta pradiniuose uždaviniuose ar planuota analizės metu. Realizuoti tokie papildymai, kaip galimybė atskirti informacinę veiklą nuo materialios veiklos bei galimybė generuot informaciją pasirinktai funkcijai. Tai suteikia IS projektuotojui galimybę pasirinkti tik reikiamą informaciją. Taip klasių modelis tampa informatyvesnis.

Apsibrėžti papildymai, kuriuos įgyvendinus generuojamas klasių modelis bus informatyvesnis. Vienas iš papildymų – galimybė generuoti klasių modelį pasirinktam procesui. Taip pat rekomenduojama realizuoti papildymus, kurie užtikrins, kad generuojamas klasių modelis būtų pilnas PIM lygio modelis (nustatomi ryšiai tarp klasių bei jų kardinalumai, ryšiai tarp srauto tipo klasių).

### 5.2.Išvados

1. Vartotojų analizė parodė, kad IS kūrimo procesas yra neoptimalus ir egzistuoja didelis proceso automatizavimo poreikis. Iš vartotojų bei esamos situacijos analizės (kaip veiklos modeliai yra taikomi IS kūrimo procese, bendras IS kūrimo procesas) išplaukia darbe kuriamo prototipo poreikis. Analizė atlikta analizuojant informacijos šaltinius.
2. Išanalizuoti Informacinių sistemų katedros (ISK) vykdomi tyrimai veiklos žiniomis grindžiamų IS kūrime ir kitų autorių tyrimai leido pasirinkti tinkamą pagrindą algoritmo projektavimui.
3. CASE priemonių (įskaitant „MagicDraw UML“) analizė parodė, kad „MagicDraw UML“ yra vienas iš populiariausių IS projektavimo paketų, kurio funkcionalumas ir pateikta dokumentacija suteikia galimybę realizuoti metodo prototipą šiai sistemai. Platus paketo naudojimas gali užtikrinti pasiūlyto metodo tolimesnį vartotojų pripažinimą ir vystymą.
4. Remiantis bendra analize, suprojektuotas klasių modelio sudarymo pagal veiklos žinias algoritmo prototipas. Taip pat sudaryta algoritmo panaudojimo metodika, kurią sudaro reikiamų pakeitimų (UML išplėtimas) ir papildomų paketų panaudojimo sąrašas.
5. Sukurtos realizacijos („MagicDraw UML“ įskiepis ir veiklos žinių saugykla „Microsoft SQL Server 2005“) testavimas patvirtino, kad prototipai veikia kokybiškai.
6. Atliktas eksperimentas (eksperimentinio pavyzdžio metodu) ir projektavimas parodė, kad prototipą reikia vystyti, norint, kad jis įgytų vartotojų pripažinimą. Pagrindiniai vystymo uždaviniai – atvaizdavimo mechanizmo vystymas, modelio generavimo nustatymų išplėtimas, ryšių sudarymo mechanizmo tobulinimas.
7. Darbo metu įgyta patirtis (analizė) leidžia teigti, kad tokios priemonės poreikis įtakos prototipo pripažinimą, tolimesnį vystymą ir naudojimą. Tokiu būdu algoritmo prototipas tampa svarbiu artefaktu, kurį reikia plėsti ir optimizuoti.

# LITERATŪRA


[1.] Skersys T., Gudas S.: Enterprise Model-based Generation of the Class Model.: Straipsnis. Kaunas. birželis 2007.

[2.] Schekkerman J.: How to Survive in the Jungle of Enterprise Architecture Frameworks. Trafford. Knyga. Canada. 2003.

[3.] Coad P., Yourdon E.: Object-oriented analysis, 2nd ed.: Knyga. Yourdon Press, New Jersey 1991.

[4.] Gudas S., Lopata A., Skersys T.: Approach to Enterprise Modelling for Information Systems Engineering. Informatica, Vol. 16, No. 2. Institute of Mathematics and Informatics, Vilnius 2005. psl. 175-192.

[5.] Gudas S.: A framework for research of information processing hierarchy in enterprise. Mathematics and Computers in Simulation, Vol. 33, No. 4. North Holland. 1991.

[6.] ENV 12204 – Systems Architecture: Language Constructs for Enterprise Modelling. Revision of ENV 12204, CEN/TC 310/WG1 2002

[7.] Thomas D.: UML - Unified or Universal Modeling Language?. Žurnalas - Object Technology, Vol. 2, No. 1. 2003

[8.] Skersys T., Gudas S.: Class model development using business rules. In: Advances in Information Systems Development: Bridging the Gap between Academia and Industry. Publikacija (ISD'2005). New York. 2006. Psl. 203-215

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

[10.] Gupta M.M., Sinha N.K. Intelligent Control Systems; Theory and Applications. The Institute of Electrical and Electronic Engineers, Inc., New York. 1996. Psl.: 856.

[11.] Skersys T., Gudas S.: Approach to Enterprise modelling for Information Systems engineering. (ISD), INFORMATICA 2005 04 03.

[12.] UEML Group.: Universal Enterprise Modelling Language, IFAC-IFIP Task Force, 1990. Prieiga internetu.: http://www.cit.gu.edu.au/~bernus/taskforce/archive/UEML-TF-IG.ppt. [Peržiūrėta 2007.09-10]

[13.] ENV 40 003. Computer Integrated Manufacturing Systems Architecture - Framework for Enterprise Modelling. CEN/CENELEC. 1990.

[14.] GERAM.: GERAM:Generalised Enterprise Reference Architecture and Methodology. Version 1.6.3. IFIP–IFAC Task Force on Architectures for Enterprise Integration. 1999. Preieiga intenetu: http://www.cit.gu.edu.au/~bernus/taskforce/geram/versions/geram1-6-3/v1.6.3.html. [Peržiūrėta 2007.09-10]

[15.] Schekkerman J. How to Survive in the Jungle of Enterprise Architecture Frameworks. Trafford, Canada 2003. Psl. 222.

[16.] Totland T. Enterprise modelling as a means to support human sense-making and communication in organizations. Norwegian University of Science and Technology (NTNU), Trondheim IDI-raport. 1997:8.

[17.] No Magic, Inc.: MagicDraw OpenAPI UserGuide. 2007 October. Prieiga internete: http://www.magicdraw.com/main.php?ts=navig&NMSESSID=a6b07b6051d48f05d46935666e0b15f5&cmd_show=1&menu=download_manual&NMSESSID=a6b07b6051d48f05d46935666e0b15f5 [Peržiūrėta 2008.01]

[18.] Internetinė prieiga: http://www.magicdraw.com pateikiama informacija [Peržiūrėta 2008.01 – 2008.06]

[19.] Subramaniam K., Dong Liu, Behrouz H. Far, Eberlein A.: Automating transition from use-cases to class model. Department of Electrical and Computer Engineering University of Calgary. 2003

[20.] Liu D., Subramaniam K., Far B.H., Eberlein A.: Automating transition from use-cases to class model**.** Straipsnis: Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference 2003.

Preiga internete: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/8688/27521/01226023.pdf Psl. 830-835. [Peržiūrėta 2008.01]

[21.] Dong Liu.: Automating Transition from Use Cases to Class Model. Publikuotas baigiamasis darbas. Department of Electrical and Computer Engineering University of Calgary. 2003.

[22.] No Magic, Inc.: UML profiling and DSL. 2003. [Peržiūrėta 2008.05-2008.06]

[23.] Pender T., UML Bible. John Wiley & Sons. 2003 [Peržiūrėta 2008.05]

[24.] OMG.: OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2. 2007. [Peržiūrėta 2008.05]

## Terminų žodynas

CIM – angl. computation independant model – nuo realizacijos nepriklausomas modelis.

CASE – angl. computer-aided system enginering.

CMM – angl. class meta-model – klasių modelio meta-modelis (saugykla).

ECM – angl. extended class model – išplėstas klasių modelis. Kuris gaunamas išplečiant OMG pasiūlytą klasių modelio meta-modelį komponentais, reikalingais šio darbo realizavimui. Išplėtimas atliekamas panaudojant profiliavimo mechanizmą.

ECMM – angl. extended class meta-model – išplėstas OMG pasiūlytas klasių modelio meta-modelis.

EMM – angl. enterprice meta-model – veiklos žinių meta-modelis. Veiklos žinių modelių saugykla. Saugomi visi komponentai.

IS – informacinė sistema.

ISD – angl. Information System Development – Informacinių sistemų kūrimas. Darbe bus naudojamas angliškas terminas ISD, nes lietuviškai trumpinys ISK (informacinių sistemų kūrimas) jau panaudotas (žiūrėti terminų žodynas: ISK).

ISK – Informacinių sistemų katedra, KTU.

Įskiepis – (plug-ins) – paketų papildai, kurie įneša naują ar pakeičia seną paketų funkcionalumą. Paprastai laisvai įdiegiami ir pašalinami.

OMG – angl. Object managment group – tarptautinis kompiuterių industrijos konsorciumas kurio siekis – pagerinti visuotinį integracijos standartų vystymą...

PIM – angl. platform independant model – nuo platformos nepriklausomas modelis – sudarinėjamas pagal turimas specifikacijas bei CIM modelį.

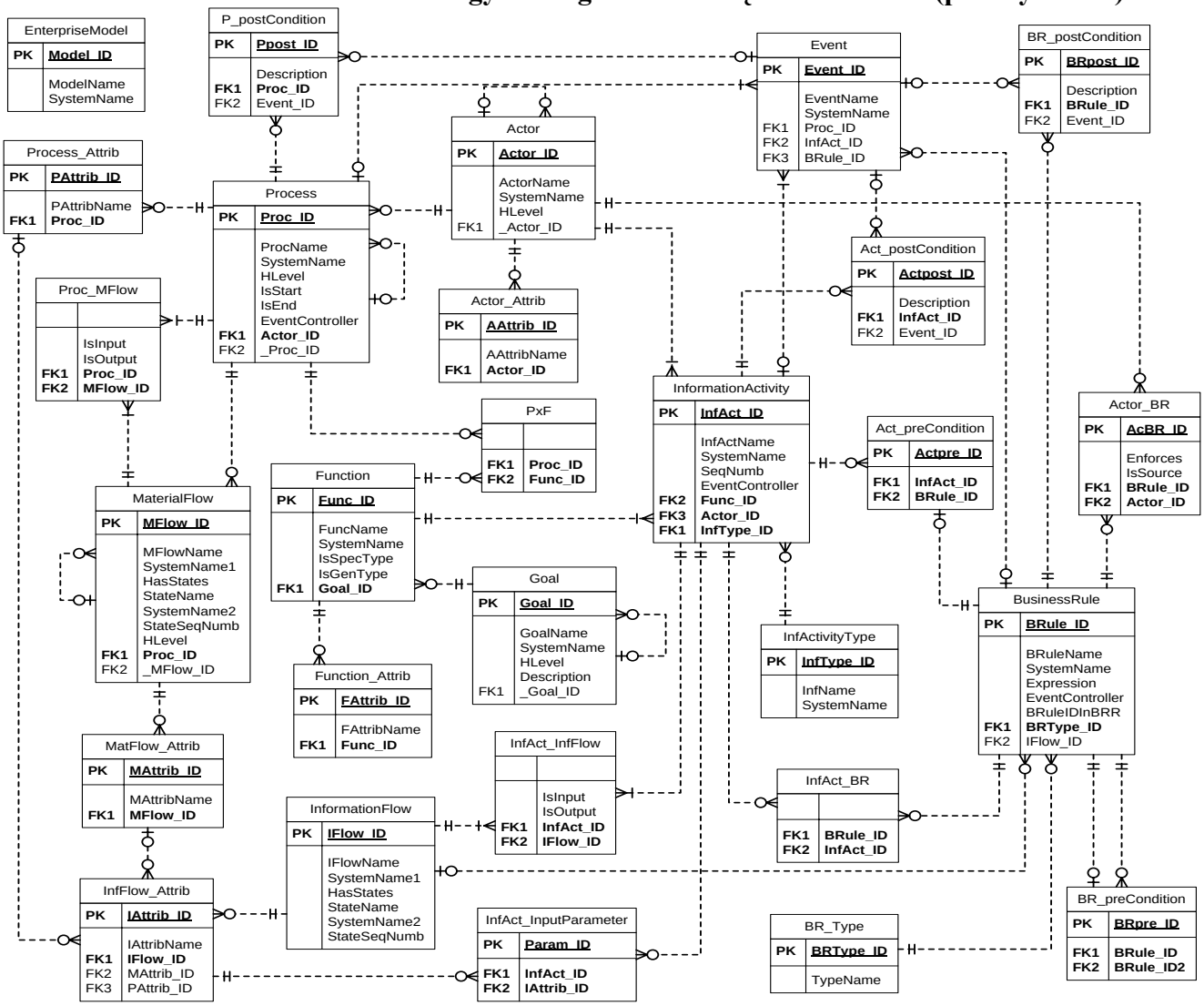
Profilis – angl. profile – „MagicDaw UML“ pakete esantis profiliavimo mechanizmas leidžia naudoti atitinkamus profilius – notacijas įvairių diagramų braižymui. Tai yra darbinė aplinka suteikianti atitinkamą funkcionalumą ir apribojimus.

PSM – angl. platform spacific model – tam tikrai platformai skirtas modelis, sudarinėjamas pagal PIM modeliuose sukauptas žines.

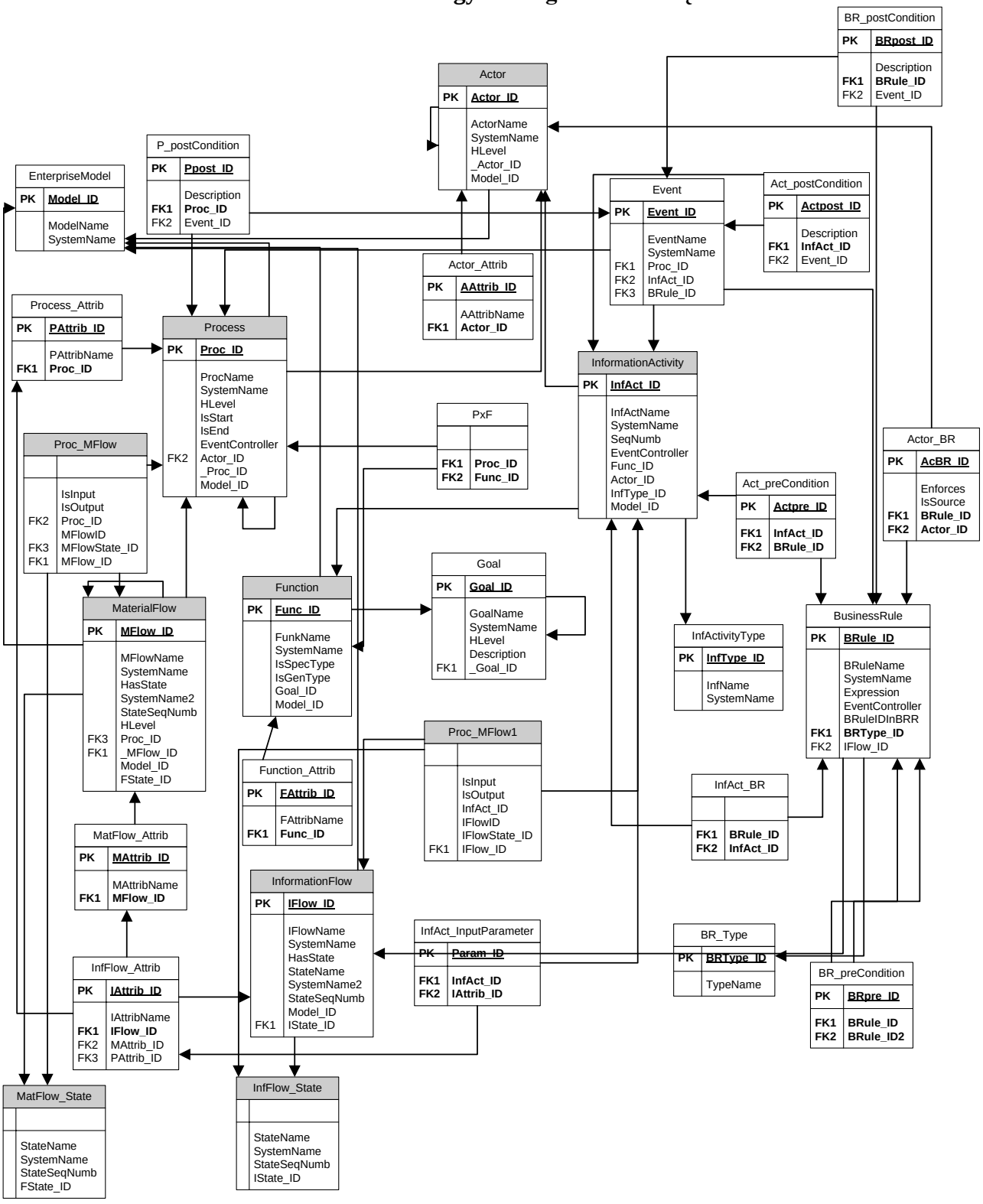
Validavimas – angl. valinate – patikra. Proceso metu patikrinama ar objektas, sritis atitinka tam tikrus apibrėžimus.

## Priedai

# Priedas Nr.1. Veiklos modelio saugyklos loginė duomenų bazės schema (pasiūlyta ISK)

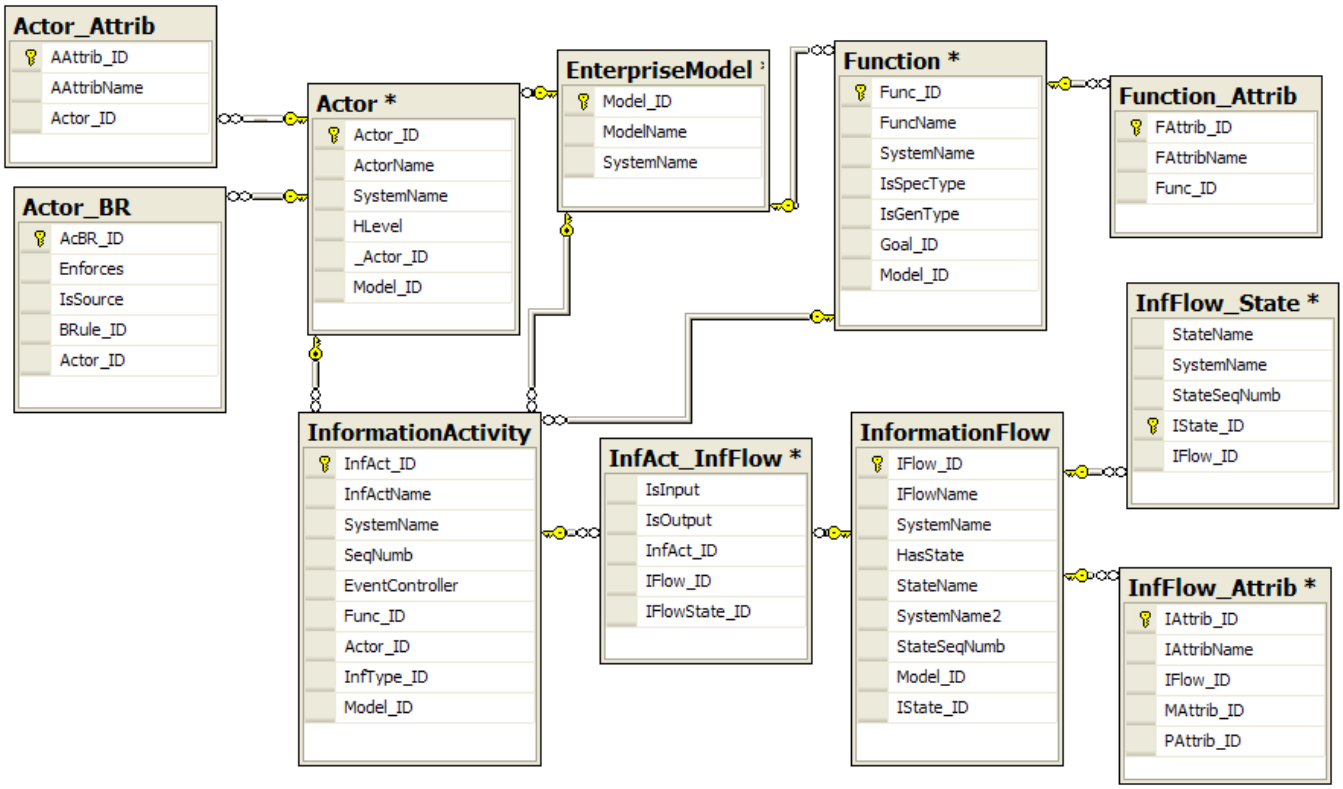


## Priedas Nr.2. Pakeisto veiklos modelio saugyklos loginė duomenų bazės schema

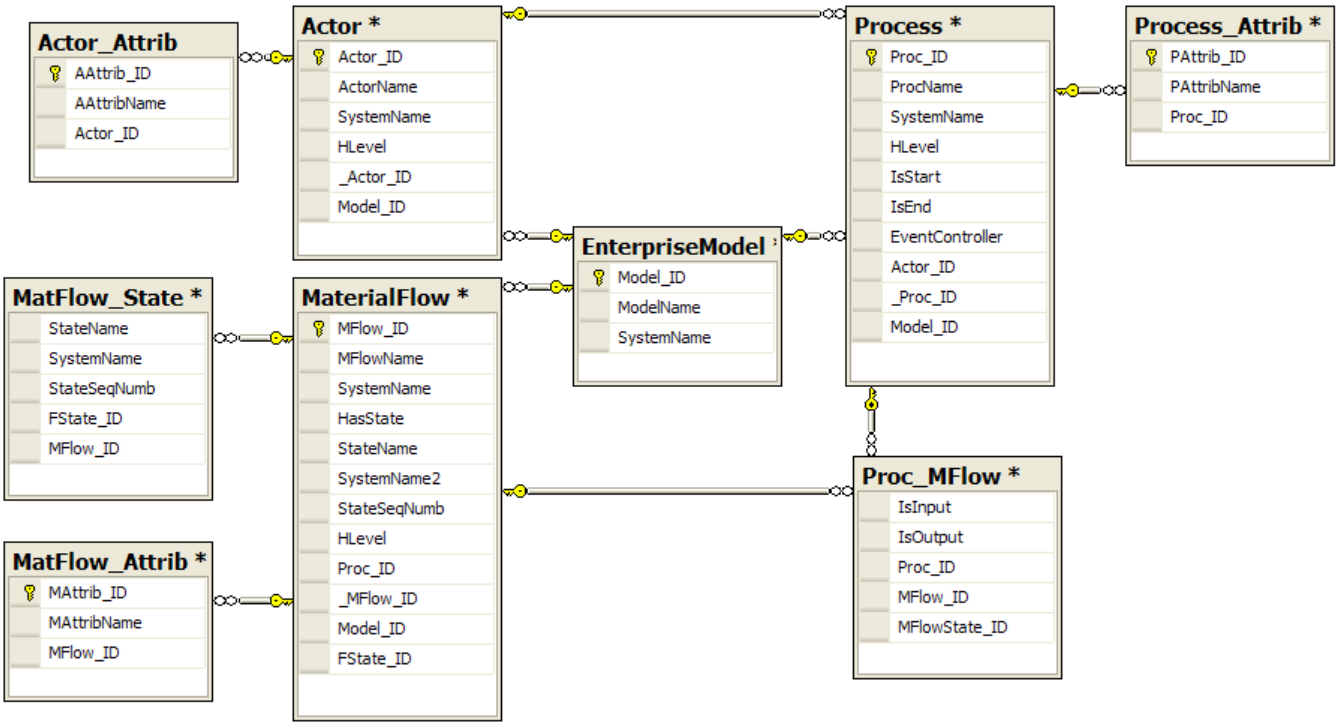


**Priedas Nr.3. Pakeisto veiklos modelio (VM) saugyklos fizinė duomenų bazės schema**

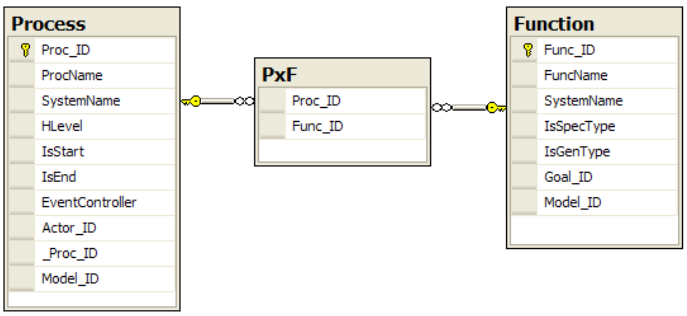
**Priedas Nr.3.1 – Pakeisto VM saugyklos fizinė duomenų bazės schema (Informacinė veikla)**




**Priedas Nr.3.2 - Pakeisto VM saugyklos fizinė duomenų bazės schema (Materiali veikla)**




**Priedas Nr.3.3 - Pakeisto VM saugyklos fizinė duomenų bazės schema (Veiklų susiejimas)**



## Priedas Nr.4. Veiklos žinių bazės pildymo komandos (pavyzdinis modelis)

```
insert into EnterpriseModel (Model_ID, ModelName, SystemName) VALUES (1, 'Uþsakymo vertinimo VM', 'Uþsakymo
vertinimo VM');
--======================================== --
-- ACTORIAI --
--======================================== --
INSERT INTO [VMS].[dbo].[Actor] ([Actor_ID], [ActorName], [SystemName], [HLevel], [_Actor_ID], [Model_ID])
     VALUES (1, 'Administration', 'Administration', 0, 0, 1);
INSERT INTO [VMS].[dbo].[Actor] ([Actor_ID], [ActorName], [SystemName], [HLevel], [_Actor_ID], [Model_ID])
     VALUES (2, 'Marketing', 'Marketing', 0, 0, 1);
INSERT INTO [VMS].[dbo].[Actor] ([Actor_ID], [ActorName], [SystemName], [HLevel], [_Actor_ID], [Model_ID])
     VALUES (3, 'Customer', 'Customer', 0, 0, 1);
INSERT INTO [VMS].[dbo].[Actor] ([Actor_ID], [ActorName], [SystemName], [HLevel], [_Actor_ID], [Model_ID])
     VALUES (4, 'Logistics', 'Logistics', 0, 0, 1);

INSERT INTO [VMS].[dbo].[Actor_Attrib] ([AAttrib_ID], [AAttribName], [Actor_ID]) VALUES (1, 'DepCode', 1);
INSERT INTO [VMS].[dbo].[Actor_Attrib] ([AAttrib_ID], [AAttribName], [Actor_ID]) VALUES (2, 'Address', 1);
INSERT INTO [VMS].[dbo].[Actor_Attrib] ([AAttrib_ID], [AAttribName], [Actor_ID]) VALUES (3, 'DepCode', 2);
INSERT INTO [VMS].[dbo].[Actor_Attrib] ([AAttrib_ID], [AAttribName], [Actor_ID]) VALUES (4, 'Address', 2);
INSERT INTO [VMS].[dbo].[Actor_Attrib] ([AAttrib_ID], [AAttribName], [Actor_ID]) VALUES (5, 'DepCode', 4);
INSERT INTO [VMS].[dbo].[Actor_Attrib] ([AAttrib_ID], [AAttribName], [Actor_ID]) VALUES (6, 'Address', 4);
--======================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--======================================== --
--======================================== --
-- FUNKCIJA - Order estimates managment --
--======================================== --
INSERT INTO [VMS].[dbo].[Function]
           ([Func_ID]
           ,[FuncName]
           ,[SystemName]
           ,[IsSpecType]
           ,[IsGenType]
           ,[Goal_ID]
           ,[Model_ID])
     VALUES (1,'Order estimates managment','Order estimates managment',0,1,1,1);
-- Sudaromi Funkcijos atributai --
INSERT INTO [VMS].[dbo].[Function_Attrib] ([FAttrib_ID],[FAttribName],[Func_ID]) VALUES (1, 'funcCode', 1);
INSERT INTO [VMS].[dbo].[Function_Attrib] ([FAttrib_ID],[FAttribName],[Func_ID]) VALUES (2, 'funcProcDate',
1);
-- Sudaromas GOAL - TIKSLTAS --
INSERT INTO [VMS].[dbo].[Goal] ([Goal_ID],[GoalName],[SystemName],[HLevel],[Description],[_Goal_ID])
     VALUES (1, 'Order estimation', 'Order estimation', 0, 'Description...', 1);
--======================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--======================================== --
--======================================== --
-- PROCESAS - Order fulfillment --
--======================================== --
INSERT INTO [VMS].[dbo].[Process]
([Proc_ID],[ProcName],[SystemName],[HLevel],[IsStart],[IsEnd],[EventController]
           ,[Actor_ID],[_Proc_ID],[Model_ID])
     VALUES (1, 'Order fulfillment', 'Order fulfillment',0,0,0,0,4,0,1)

INSERT INTO [VMS].[dbo].[Process_Attrib] ([PAttrib_ID],[PAttribName],[Proc_ID]) VALUES (1, 'TypeOfDelivery',
1);
INSERT INTO [VMS].[dbo].[Process_Attrib] ([PAttrib_ID],[PAttribName],[Proc_ID]) VALUES (2, 'IsActive', 1);
INSERT INTO [VMS].[dbo].[Process_Attrib] ([PAttrib_ID],[PAttribName],[Proc_ID]) VALUES (3, 'StartDate', 1);
INSERT INTO [VMS].[dbo].[Process_Attrib] ([PAttrib_ID],[PAttribName],[Proc_ID]) VALUES (4, 'Duration', 1);

--======================================== --
-- Materialus srautai PROCESUI - Order fulfillment --
--======================================== --
-- ========== 1 ================ --
-- Iejimas --
-- FlowState --
INSERT INTO [VMS].[dbo].[MatFlow_State] ([StateName],[SystemName],[StateSeqNumb],[FState_ID])
     VALUES ('undelivered', 'undelivered', 1, 1);
-- Flow --
INSERT INTO [VMS].[dbo].[MaterialFlow]
([MFlow_ID],[MFlowName],[SystemName],[HasState],[HLevel],[Proc_ID],[_MFlow_ID]
           ,[Model_ID],[FState_ID])
     VALUES (1, 'Furniture', 'Furniture', 1, 0, 1, 1, 1, 1);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (1, 'furCode', 1);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (2, 'furName', 1);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (3, 'amount', 1);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (4, 'price', 1);
-- SUSIEJAM SRAUTA SU PROCESU --
INSERT INTO [VMS].[dbo].[Proc_MFlow] ([IsInput],[IsOutput],[Proc_ID],[MFlow_ID]) VALUES (1, 0, 1, 1);

-- Isejimas --
-- FlowState --
```

```
INSERT INTO [VMS].[dbo].[MatFlow_State] ([StateName],[SystemName],[StateSeqNumb],[FState_ID])
     VALUES ('delivered', 'delivered', 1, 2);
-- Flow --
INSERT INTO [VMS].[dbo].[MaterialFlow]
([MFlow_ID],[MFlowName],[SystemName],[HasState],[HLevel],[Proc_ID],[_MFlow_ID]
           ,[Model_ID],[FState_ID])
     VALUES (2, 'Furniture', 'Furniture', 1, 0, 1, 1, 1, 2);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (5, 'furCode', 2);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (6, 'furName', 2);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (7, 'amount', 2);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (8, 'price', 2);
-- SUSIEJAM SRAUTA SU PROCESU --
INSERT INTO [VMS].[dbo].[Proc_MFlow] ([IsInput],[IsOutput],[Proc_ID],[MFlow_ID]) VALUES (0, 1, 1, 2);
-- ========== 2 ================ --
-- Iejimas --
-- FlowState --
INSERT INTO [VMS].[dbo].[MatFlow_State] ([FState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (3, 'submited', 'submited', 1);
-- Flow --
INSERT INTO [VMS].[dbo].[MaterialFlow]
([MFlow_ID],[MFlowName],[SystemName],[HasState],[HLevel],[Proc_ID],[_MFlow_ID]
           ,[Model_ID],[FState_ID])
     VALUES (3, 'Order', 'Order', 1, 0, 1, 1, 1, 3);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (9, 'orderCode', 3);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (10, 'custCode', 3);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (11, 'regDate', 3);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (12, 'deliveryDate',
3);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (13, 'custDiscount',
3);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (14, 'calculation',
3);
-- SUSIEJAM SRAUTA SU PROCESU --
INSERT INTO [VMS].[dbo].[Proc_MFlow] ([IsInput],[IsOutput],[Proc_ID],[MFlow_ID]) VALUES (1, 0, 1, 3);

-- Isejimas --
INSERT INTO [VMS].[dbo].[MatFlow_State] ([FState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (4, 'fullfilled', 'fullfilled', 2);
-- Flow --
INSERT INTO [VMS].[dbo].[MaterialFlow]
([MFlow_ID],[MFlowName],[SystemName],[HasState],[HLevel],[Proc_ID],[_MFlow_ID]
           ,[Model_ID],[FState_ID])
     VALUES (4, 'Order', 'Order', 1, 0, 1, 1, 1, 4);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (15, 'orderCode', 4);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (16, 'custCode', 4);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (17, 'regDate', 4);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (18, 'deliveryDate',
4);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (19, 'custDiscount',
4);
INSERT INTO [VMS].[dbo].[MatFlow_Attrib] ([MAttrib_ID],[MAttribName],[MFlow_ID]) VALUES (20, 'calculation',
4);
-- SUSIEJAM SRAUTA SU PROCESU --
INSERT INTO [VMS].[dbo].[Proc_MFlow] ([IsInput],[IsOutput],[Proc_ID],[MFlow_ID]) VALUES (0, 1, 1, 4);
----============================================================== -------
--======================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--======================================== --
-- Procesas "Order FullFillMent" susiejamas su FUNKCIJA "Order estimates managment" --
--======================================== --
INSERT INTO [VMS].[dbo].[PxF] ([Proc_ID],[Func_ID]) VALUES (1, 1);
--======================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--======================================== --
-- UZPILDYMAS InformacinesVeiklos --
--======================================== --
-- ========== 1 ========== --
-- Sudarom InfAct TIPA --
INSERT INTO [VMS].[dbo].[InfActivityType] ([InfType_ID],[InfName],[SystemName])
VALUES (1, 'Estimate registration', 'Estimate registration');
-- Sudarom Informacine veikla --
INSERT INTO [VMS].[dbo].[InformationActivity]
           ([InfAct_ID]
           ,[InfActName]
           ,[SystemName]
           ,[SeqNumb]
           ,[EventController]
           ,[Func_ID]
           ,[Actor_ID]
           ,[InfType_ID])
     VALUES (1, 'Estimate registration', 'Estimate registration', 1, 0, 1, 1, 1);
```

```
--======================================== --
-- Informaciniai srautai Veiklai - Estimate Registration --
--======================================== --
-- ========== 1.1 ================ --
-- Iejimas --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (1, 'unregistred', 'unregistred', 1);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (1, 'Estimate', 'Estimate', 1, '', '', 0, 1, 1);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (1, 'estimateCode', 1, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (2, 'orderCode', 1, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (3, 'custDiscount', 1, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (4, 'calculation', 1, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (1, 0, 1, 1);
----============================================================ ---- --
-- Isejimas --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (2, 'registred', 'registred', 2);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (2, 'Estimate', 'Estimate', 1, '', '', 0, 1, 2);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (5, 'estimateCode', 2, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (6, 'orderCode', 2, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (7, 'custDiscount', 2, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (8, 'calculation', 2, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 1, 2);
----============================================================ ---- --
--======================================== --
--<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>> --
--======================================== --
-- ========== 2 ========== --
-- Sudarom InfAct TIPA --
INSERT INTO [VMS].[dbo].[InfActivityType] ([InfType_ID],[InfName],[SystemName])
VALUES (2, 'Estimate calculation', 'Estimate calculation');
-- Sudarom Informacine veikla --
INSERT INTO [VMS].[dbo].[InformationActivity]
           ([InfAct_ID]
           ,[InfActName]
           ,[SystemName]
           ,[SeqNumb]
           ,[EventController]
           ,[Func_ID]
           ,[Actor_ID]
           ,[InfType_ID])
     VALUES (2, 'Estimate calculation', 'Estimate calculation', 2, 0, 1, 2, 2);
--======================================== --
-- Informaciniai srautai Veiklai - Estimate Calculation --
--======================================== --
-- ========== 1 ================ --
-- Iejimas --
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (1, 0, 2, 2);
----============================================================ ---- --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (3, 'corrected', 'corrected', 6);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (3, 'Estimate', 'Estimate', 1, '', '', 0, 1, 3);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (9, 'estimateCode', 3, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (10, 'orderCode', 3, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (11, 'custDiscount', 3, 0, 0);
```

```
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (12, 'calculation', 3, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (1, 0, 2, 3);
----============================================================ ---- --
-- Isejimas1 --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (4, 'calculated', 'calculated', 3);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (4, 'Estimate', 'Estimate', 1, '', '', 0, 1, 4);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (13, 'estimateCode', 4, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (14, 'orderCode', 4, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (15, 'custDiscount', 4, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (16, 'calculation', 4, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 2, 4);
----============================================================ ---- --
--====================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--====================================== --
-- ========== 3 ========== --
-- Sudarom InfAct TIPA --
INSERT INTO [VMS].[dbo].[InfActivityType] ([InfType_ID],[InfName],[SystemName])
VALUES (3, 'Estimate confirmation', 'Estimate confirmation');
-- Sudarom Informacine veikla --
INSERT INTO [VMS].[dbo].[InformationActivity]
           ([InfAct_ID]
           ,[InfActName]
           ,[SystemName]
           ,[SeqNumb]
           ,[EventController]
           ,[Func_ID]
           ,[Actor_ID]
           ,[InfType_ID])
     VALUES (3, 'Estimate confirmation', 'Estimate confirmation', 3, 0, 1, 3, 3);
--====================================== --
-- Informaciniai srautai Veiklai - Estimate confirmation --
--====================================== --
-- ========== 1 ================ --
-- Iejimas --
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (1, 0, 3, 4);
----============================================================ ---- --
-- Isejimas1 --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (5, 'confirmed', 'confirmed', 4);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (5, 'Estimate', 'Estimate', 1, '', '', 0, 1, 5);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (17, 'estimateCode', 5, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (18, 'orderCode', 5, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (19, 'custDiscount', 5, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (20, 'calculation', 5, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 3, 5);
----============================================================ ---- --
-- Isejimas2 --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (6, 'rejected', 'rejected', 5);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (6, 'Estimate', 'Estimate', 1, '', '', 0, 1, 6);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (21, 'estimateCode', 6, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (22, 'orderCode', 6, 0, 0);
```

```
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (23, 'custDiscount', 6, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (24, 'calculation', 6, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 3, 6);
----================================================================ ---- --
--======================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--======================================== --
-- ========== 4 ========== --
-- Sudarom InfAct TIPA --
INSERT INTO [VMS].[dbo].[InfActivityType] ([InfType_ID],[InfName],[SystemName])
VALUES (4, 'Estimate data correction', 'Estimate data correction');
-- Sudarom Informacine veikla --
INSERT INTO [VMS].[dbo].[InformationActivity]
           ([InfAct_ID]
           ,[InfActName]
           ,[SystemName]
           ,[SeqNumb]
           ,[EventController]
           ,[Func_ID]
           ,[Actor_ID]
           ,[InfType_ID])
     VALUES (4, 'Estimate data correction', 'Estimate data correction', 4, 1, 1, 3, 4);
--======================================== --
-- Informaciniai srautai Veiklai - Estimate data correction --
--======================================== --
-- ========== 1 ================ --
-- Iejimas --
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (1, 0, 4, 6);
----================================================================ ---- --
-- Isejimas1 --
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 4, 3);
----================================================================ ---- --
--======================================== --
--<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>> --
--======================================== --
-- ========== 5 ========== --
-- Sudarom InfAct TIPA --
INSERT INTO [VMS].[dbo].[InfActivityType] ([InfType_ID],[InfName],[SystemName])
VALUES (5, 'Estimate dispatch', 'Estimate dispatch');
-- Sudarom Informacine veikla --
INSERT INTO [VMS].[dbo].[InformationActivity]
           ([InfAct_ID]
           ,[InfActName]
           ,[SystemName]
           ,[SeqNumb]
           ,[EventController]
           ,[Func_ID]
           ,[Actor_ID]
           ,[InfType_ID])
     VALUES (5, 'Estimate dispatch', 'Estimate dispatch', 4, 1, 1, 1, 5);
--======================================== --
-- Informaciniai srautai Veiklai - Estimate dispatch --
--======================================== --
-- ========== 1 ================ --
-- Iejimas --
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 5, 5);
--========================================== --
-- Isejimas --
-- FlowState --
INSERT INTO [VMS].[dbo].[InfFlow_State] ([IState_ID],[StateName],[SystemName],[StateSeqNumb])
     VALUES (7, 'dispached', 'dispached', 7);
-- Flow --
INSERT INTO [VMS].[dbo].[InformationFlow] ([IFlow_ID],[IFlowName],[SystemName],[HasState],[StateName]
           ,[SystemName2],[StateSeqNumb],[Model_ID],[IState_ID])
     VALUES (7, 'Estimate', 'Estimate', 1, '', '', 0, 1, 7);
-- Srauto atributai --
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (25, 'estimateCode', 7, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (26, 'orderCode', 7, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (27, 'custDiscount', 7, 0, 0);
INSERT INTO [VMS].[dbo].[InfFlow_Attrib] ([IAttrib_ID],[IAttribName],[IFlow_ID],[MAttrib_ID],[PAttrib_ID])
     VALUES (28, 'calculation', 7, 0, 0);
-- SUSIEJAM SRAUTA SU Veikla --
INSERT INTO [VMS].[dbo].[InfAct_InfFlow] ([IsInput],[IsOutput],[InfAct_ID],[IFlow_ID])  VALUES (0, 1, 5, 7);
```