KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Dainius Keizeris

ONTOLOGIJŲ VAIZDAVIMAS UML: TRANSFORMAVIMO TAISYKLĖS IR PROTOTIPAS

Baigiamasis magistro projektas

Vadovas
doc. dr. Rita Butkienė

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

ONTOLOGIJŲ VAIZDAVIMAS *UML*: TRANSFORMAVIMO TAISYKLĖS IR PROTOTIPAS

Baigiamasis magistro projektas
Informacinių sistemų inžinerijos studijų programa (kodas 621E15001)

Vadovas
doc. dr. Rita Butkienė
2017-05-17

Konsultantas
prof. dr. Lina Nemuraitė

Recenzentas
lekt. dr. Jaroslav Karpovič
2017-05-17

Projektą atliko
Dainius Keizeris
2017-05-17

KAUNAS, 2017

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

(Fakultetas)

Dainius Keizeris

(Studento vardas, pavardė)

Informacinių sistemų inžinerijos studijų programa, 621E15001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Ontologijų vaizdavimas *UML*: transformavimo taisyklės ir prototipas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2017 m. gegužės 17 d.

Kaunas

Patvirtinu, kad mano, **Dainiaus Keizerio**, baigiamasis projektas tema „Ontologijų vaizdavimas *UML*: transformavimo taisyklės ir prototipas“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

(vardą ir pavardę įrašyti ranka) *(parašas)*

Dainius Keizeris. Ontologijų vaizdavimas *UML*: transformavimo taisyklės ir prototipas. *Magistro baigiamasis projektas* / vadovas doc. dr. Rita Butkienė; Kauno technologijos universitetas, Informatikos fakultetas.

Reikšminiai žodžiai: Transformavimo taisyklės, ontologija, *OWL 2*, *UML*.

Kaunas, 2017. 57p.

SANTRAUKA

Šio darbo idėja kilo, kuomet nebuvo rasta tinkamų įrankių patogiai ir tiksliai vaizduoti ontologijas informacinių sistemų projektuose, kuriuose naudojami žinių modeliai. Nors yra sukurtas ontologijų vaizdavimo profilis, tačiau jis nėra realizuotas *UML CASE* įrankiuose ir nėra priemonių esamoms ontologijoms automatiškai perkelti į *CASE* įrankių aplinką. Todėl projektuotojai gaišta papildomą laiką ir turi lygiagrečiai atlikti pakeitimus dviejose aplinkose – ontologijų redaktoriuose ir *CASE* įrankiuose.

Šis darbas apima esamų sprendimų, transformavimo *OWL 2* į *UML* analizę, transformavimo taisykles, „MagicDraw“ įskiepio prototipą, bei išvadas. Šiame darbe buvo sukurtas „MagicDraw“ aplinkoje *OWL 2* ontologijos transformavimo į *UML* įskiepio prototipas. Eksperimentinis tyrimas parodė, kad siūlomas metodas transformuoti *OWL 2* ontologiją į *UML* yra efektyvus laiko išteklių atžvilgiu taip pat padeda išvengti žmogiškųjų klaidų. Sukurtas prototipas yra tik įrodymas, tačiau idėja gali būti panaudojama tolimesnei plėtrai. Transformacija *OWL 2* į *UML* pavaizduota pavyzdyje.

Dainius Keizeris. Representation of Ontology in *UML*: Transformation Rules and Prototype. *Final Degree Project of Master of Information Systems Engineering* / Supervisor doc. dr. Rita Butkienė; Kaunas University of Technology, Faculty of Informatics.

Keywords: Transformation Rules, ontology, *OWL 2*, *UML*.

Kaunas, 2017. 57p.

SUMMARY

The idea of master thesis was raised regarding the situation that there are no suitable tools for easily and precisely represent ontologies in projects of information technologies. Despite the fact that the profile of ontologies visualization is used, however, it is not realized in *UML CASE* tools. In addition, the profile does not have options to automatically transfer existing ontologies to the *CASE* tools program. Because of this reason engineers have to waste excessive amount of time and parallel make changes in two areas: editor of ontologies and *CASE* tools.

The master thesis covers analysis of existing decisions, transformation of *OWL 2* to *UML*, transformation rules, „MagicDraw“ prototype and conclusions. In the master thesis (in „MagicDraw“ area) a new prototype of ontologies transformation from *OWL 2* to *UML* was created. Experimental research has shown that the offered method to transform ontologies from *OWL 2* to *UML* can help not only to save time but avoid human errors. The created prototype is used as an idea, however, the same idea can be transformed for further development as well.

Transformation from *OWL 2* to *UML* is showed in the following example.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Web 2.0 - interaktyvus internetas.
Web 3.0 - semantinis internetas.
API - aplikacijų programavimo sąsaja.
OWL - ontologijos tinklo kalba. Ontologijos aprašymo kalba atitinkanti W3C Semantikos Tinklo standartams.
URI - universalus išteklių identifikatorius. Tai yra formatas, naudojamas semantiniame tinkle, kad paskirtų identifikuotus išteklius.
Ontologija - oficialus žodynas tinkamų sąvokų, ypatybių, kuriuos sieja tam tikros taisyklės.
Plug-in, įskiepis - sukompiliuotas programinis kodas, kuris padidina programos galimybes.
XML - duomenų struktūrų bei jų turinio aprašomoji kalba (angl. *eXtensibleMarkupLanguage*).
HTML (angl. *HypertextMarkupLanguage*) - tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete. Kalbą standartizuoja W3 konsorciumas.
Anotacija - dokumentas ar dalis dokumento kuriame yra informacija apie kitą dokumentą ar jo dalį, jungiantis metaduomenis su ištekliais, kad apibūdintų objektą.
Semantinė anotacija - anotacija, kurioje metaduomenys yra formaliai apibrėžti ir apdorojami kompiuteriu.
Metaduomenys - duomenys, kurie apibūdina kitus duomenis. Semantikos tinkle jie yra panaudoti, kad apibūdintų išteklius.
RDF - išteklių apibūdinimo struktūra. Tai yra W3C standartinė kalba tam, kad formaliai apibūdintų išteklius. Oficialus išteklių apibūdinimas formuoja Semantikos Tinklo pagrindą
RDF Schema - ontologijos aprašymo kalba W3C Semantikos Tinklo standartams. Ji turi mažiau išraiškingumo negu *OWL*.
Žiniatinklis - pasaulinis tinklas (angl. *WorldWideWeb* arba *WWW*) - interneto dalis, resursai, kuriuos internete galima pasiekti naudojant URL.
W3C - žiniatinklio konsorciumas. Tai yra organizacija, atsakinga už tinklo standartų išvystymą.
UML - modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.

ĮVADAS

Šis darbas – „Ontologijų vaizdavimas *UML*: transformavimo taisyklės ir prototipas“ – priklauso informacinių sistemų inžinerijos studijų programai. Darbo metu realizacijos klausimais taip pat buvo konsultuotasi su „NoMagic“ įmonės atstovu Justinu Bisikirsku.

Darbo problema ir aktualumas

Tyrimo problema – nėra tinkamų įrankių patogiai ir tiksliai vaizduoti ontologijas informacinių sistemų projektuose, kuriuose naudojami žinių modeliai. Dažniausiai ontologijos kuriamos ontologijų redaktoriais, o po to daromi jų grafiniai modeliai, pavyzdžiui, *UML* klasių diagramų pavidalu. Nors yra sukurtas ontologijų vaizdavimo profilis [14], tačiau jis nėra realizuotas *UML CASE* įrankiuose ir nėra priemonių esamoms ontologijoms automatiškai perkelti į *CASE* įrankių aplinką. Todėl projektuotojai gaišta papildomą laiką ir turi lygiagrečiai atlikti pakeitimus dviejose aplinkose – ontologijų redaktoriuose ir *CASE* įrankiuose.

Šio darbo tyrimo objektas – *OWL 2* ontologijų automatizuoto vaizdavimo *CASE* įrankiuose taikant *UML* profilį procesas. Tyrimo sritis – ontologijų ir objektinių modelių specifikavimo ir vaizdavimo kalbos *OWL 2* ir *UML*, *UML* profiliai, modeliais grindžiamos transformacijos, *CASE* įrankių plėtimo technologijos

Darbo tikslas ir uždaviniai

Darbo tikslas – sudaryti galimybę automatizuotai atvaizduoti *OWL 2* ontologijas *UML CASE* įrankiuose sukuriant *OWL 2* ontologijų transformavimo į *UML* profilį taisykles ir algoritmus bei juos realizuojančio įrankio prototipą.

Šiam tikslui pasiekti iškelti penki uždaviniai.

1. Išanalizuoti:
   1.1. ontologijų kalbos *OWL 2* sąvokas, metamodelį, redagavimo įrankius;
   1.2. *UML* kalbą, *CASE* įrankius;
   1.3. panašius sprendimus ir tyrimus, aprašytus mokslinėse publikacijose ir elektroninėje erdvėje;
   1.4. pasirinkti tinkamiausią sprendimą ir technologines priemones.
2. Sudaryti transformavimo taisykles ir algoritmus, suprojektuoti transformavimo įrankį;
3. Realizuoti įrankio prototipą, jį ištestuoti;
4. Atlikti eksperimentą, kuris leistų įvertinti sprendimo tinkamumą ir efektyvumą;
5. Apibendrinti tyrimo rezultatus.

Darbo rezultatai ir jų svarba

Sukurtas įskiepis sudaro galimybę įtraukti ontologijas į informacinių sistemų projektavimą *UML CASE* įrankyje („MagicDraw“). Tai suteikia galimybė įrankyje vizualizuoti ontologijas *UML* pagrindu. „MagicDraw“ [2] apima visą *UML*, kas leidžia modelius transformuoti ir į kitą formatą, taip pat susieti ontologiją su kuriamos sistemos reikalavimais ar sistemos architektūra.

Darbo struktūra

Skyriuje „Probleminės srities analizė“ suformuluoti darbo tikslai, apibrėžta tyrimo sritis, objektas ir problema. Pateikiamas ontologijos apibrėžimas ir pagrindinės sąvokos. Tolimesniuose skyriuose aprašoma semantinio tinklo technologijų dėklo pasirinkta technologija, artimiausia tyrimo sričiai. Atlikta *OWL 2* ontologijų kalba aprašytų ontologijų vizualizavimo įrankių analizė, iškeltas projekto tikslas. Skyriaus pabaigoje suformuotos analizės išvados.

Skyriuje „Ontologijos transformacijos įskiepio sprendimo reikalavimų specifikacija“ nustatyti transformacijos įskiepio nefunkciniai reikalavimai, išskirti sistemos panaudojimo atvejai ir sudarytos detalios jų specifikacijos.

Kituose „Ontologijos transformacijos įskiepio projektas“ ir „Ontologijos transformacijos įskiepio sprendimo realizacija ir testavimas“ skyriuose pateikiamas sukurtas įskiepio realizacijos projektas bei aprašyta įskiepio prototipo realizacija.

Skyriuje „Eksperimentinis ontologijos transformacijos įskiepio tyrimas“ sudaryta ontologija, kuri transformuojama į *UML*. Skyriaus pabaigoje pateikiama atlikta darbo rezultatų analizė.

1. ONTOLOGIJOS TRANSFORMAVIMO Į *UML* ANALIZĖ

1.1. Analizės tikslas

Pagrindinis analizės tikslas yra išsiaiškinti ir išanalizuoti tinkamiausius esamus sprendimus, priemones, leidžiančias ontologijas transformuoti bei atvaizduoti *UML* sukuriant taisykles, algoritmus ir juos realizuojančio įrankio prototipą.

1.2. Tyrimo objektas, sritis ir problema

Tyrimo problema – nėra tinkamų įrankių patogiai ir tiksliai vaizduoti ontologijas informacinių sistemų projektuose, kuriuose naudojami žinių modeliai. Dažniausiai ontologijos kuriamos ontologijų redaktoriais, o po to daromi jų grafiniai modeliai pavyzdžiui, *UML* klasių diagramų pavidalu. Nors yra sukurtas ontologijų vaizdavimo profilis [14], tačiau jis nėra realizuotas *UML CASE* įrankiuose ir nėra priemonių esamoms ontologijoms automatiškai perkelti į *CASE* įrankių aplinką. Todėl projektuotojai gaišta papildomą laiką ir turi lygiagrečiai atlikti pakeitimus dviejose aplinkose – ontologijų redaktoriuose ir *CASE* įrankiuose.

Šio darbo tyrimo objektas – *OWL 2* ontologijų automatizuoto vaizdavimo *CASE* įrankiuose taikant *UML* profilį procesas.

Tyrimo sritis – ontologijų ir objektinių modelių specifikavimo ir vaizdavimo kalbos *OWL 2* ir *UML*, *UML* profiliai, modeliais grindžiamos transformacijos, *CASE* įrankių plėtimo technologijos.

1.3. Ontologijos termino apibrėžimas

Terminas „Ontologija“ yra kilęs iš filosofijos srities ir yra sudarytas iš dviejų graikų kalbos žodžių: οντος („būtis“) ir λογος („teorija“, „mokslas“). Terminas pirmą kartą buvo paminėtas dar XVII a. filosofijoje. Šiame moksle ontologija yra metafizikos šaka, nagrinėjanti ryšius, tipus, kategorijas ir egzistencijos klausimus, pasireiškiančius būtyje. Ontologija turi didelę įtaką realybės koncepcijai, kadangi stengiasi kuo bendriau aprašyti viską, kas egzistuoja, neapsiribojant pavienių mokslų išvadomis ir galbūt peržengiant jas. Kiekviena ontologija turi atsakyti į klausimą, kurie žodžiai nurodo į esybes, o kurie – ne, kodėl taip yra ir kokios kategorijos iš to susidaro. Pagrindinis ontologijos klausimas – „Kas egzistuoja?“

Kitais žodžiais tariant, ontologija reiškia tam tikros dalykinės srities sąvokų visumos specifikavimą išreikštu pavidalu. Tai tarsi ypatinga žinių bazė, aprašanti faktus, kuriuos tam tikra naudotojų grupė laiko visada teisingais, remiantis sutartomis naudojamo žodyno terminų prasmėmis [16]. Šiame darbe naudojami taikomųjų ontologijų pavyzdžiai, kurie skirti siauresnių dalykinių sričių taikomiesiems uždaviniams spręsti.

Norint vizualizuoti *UML CASE* įrankyje ontologijas, jos turi būti aprašytos ontologijų kalba *OWL 2*.

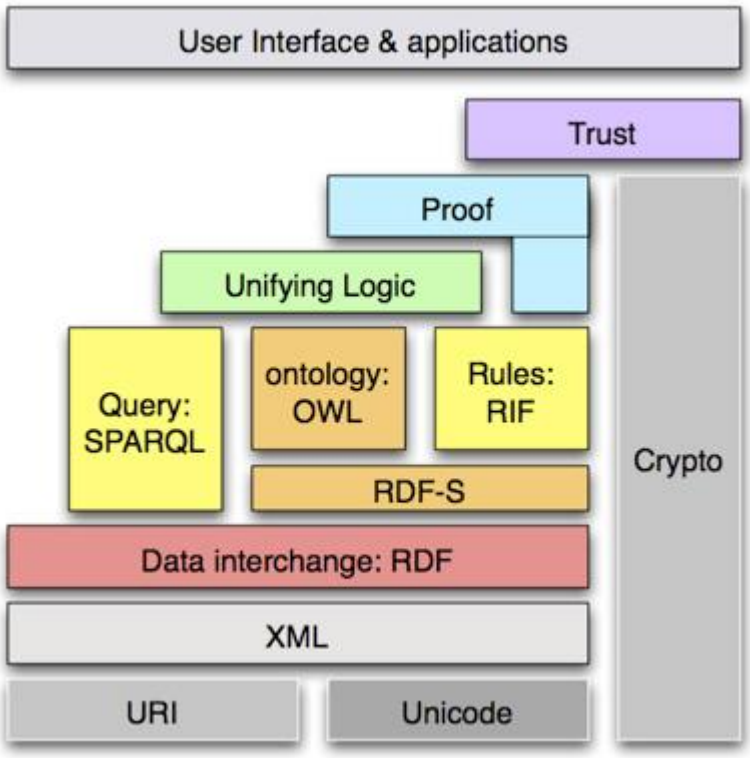
1.4. Semantinio tinklo technologijos

Semantinis tinklas paremtas naujomis pasaulinio tinklo kalbomis tokiomis kaip *XML*, *RDF* ir *OWL* bei įrankiais kurie naudoja šias kalbas.

Informacija pasauliniame tinkle tarpusavyje yra susieta nuorodomis, pagal kurių kontekstinę reikšmę informacijos šaltinių tarpusavio sąryšis dažniausiai lengvai suprantamas žmogui, tačiau nesuprantamas kompiuteriams [19]. Semantinio tinklo technologijų dėklas (1 pav.) atspindi naujosios kartos tinklo architektūrą. Joje hierarchiškai, skirtinguose sluoksniuose pateikiamos technologijos ir standartai būtini semantiniam tinklui įgyvendinti. Kiekvienas sluoksnis išnaudoja hierarchiškai žemesniame sluoksnyje esančių technologijų teikiamas galimybes. Visus steko sluoksnius, pradedant nuo žemiausio, galima išskaidyti į tris pagrindines dalis:

- saityno technologijos – dabartiniame pasauliniame tinkle naudojamos technologijos ir standartai (*Unicode, URI, XML);*

- semantinio tinklo technologijos – tai W3C konsorciumo standartizuotos technologijos, naudojamos specifiškai naujos kartos tinklo taikomiesiems uždaviniams spręsti (*RDF, RDFS, OWL, SPARQL*);
- koncepcinės semantinio tinklo technologijos – tai technologijos, kurios taip pat būtinos semantiniam tinklui įgyvendinti, tačiau kol kas nėra standartizuotos ir siekia tik idėjinį lygmenį (*Logic, Proof, Trust*).



1 pav. Semantinio tinklo technologijų dėklas [17]

Svarbiausios ir aktualiausios tyrimo sričiai semantinio tinklo duomenims aprašyti naudojamos kalbos apžvelgiamos tolimesniuose darbo skyreliuose.

1.4.1. *XML* kalba

XML – išplėstinė žymėjimo kalba (angl. *Extensible Markup Language*). Ši kalba yra rekomenduojama *W3C* konsorciumo kaip bendros paskirties duomenų struktūrų ir jų turinio aprašomoji kalba. Pagrindinis *XML* kalbos tikslas yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingo tipo sistemų, dažniausiai sujungtų internetu. *XML* kalbos vienetas yra elementas, kuris visada turi vardą, taip pat be jo gali turėti:

- reikiamą skaičių atributų, kurie turi savo vardus ir reikšmes;
- dukterinius šio elemento viduje esančius elementus;
- su elementu susijusį tekstą (elemento viduje).

Žemiau pavaizduotas *XML* kalbos pavyzdys:

```
<automobiliai>
  <automobilis atributas="reikšmė">
    <markė>Audi</markė>
    <modelis>A4</modelis>
    <kubatūra>4.2</kubatūra>
  </automobilis>
```

</automobiliai>

XML elementų atributuose galima įrašyti informaciją, panašiai kaip ir elementų turiniuose. Atributai negali apibrėžti dokumento struktūros, nėra lengvai išplečiami, yra sunkiau apdorojami programų. *XML* neturi apibrėžtų žymių, jas dokumente aprašo dokumento autorius, kurdamas *XML* dokumentą. *XML* formate žymės vardas turi keletą apribojimų:

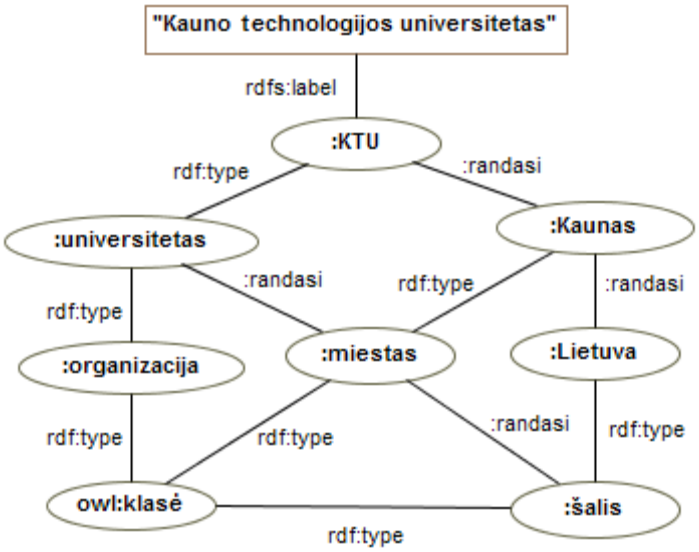
- negali prasidėti skaičiumi, simboliu '_',
- negali būti tarpų žymės varde,
- žymės vardas negali būti *XML*,
- negalima naudoti simbolio ':'.

XML žymės yra konteinerinės – turi žymės pradžią ir žymės pabaigą. *XML* formate žymėse yra skiriamos didžiosios ir mažosios raidės. *XML* vieno elemento pabaigos žymė negali būti kito elemento viduje, pvz.: *<e1><e2></e1></e2>*. Jei elementas yra kito elemento viduje, jis yra laikomas sūnumi to elemento, kuriame yra. Elementas, gaubiantis kitą elementą, yra laikomas viduje esančio elemento tėvu. Pavyzdžiui, *<e1><e2></e2></p1>* elementas *e1* yra elemento *e2* tėvas, o elementas *e2* yra elemento *e1* sūnus. *XML* dokumento elementų turinys gali būti: mišrus, paprastas, tuščias arba kiti dokumento elementai. Mišrus elemento turinys talpina kitus elementus ir tekstą. Paprastas elemento turinys yra tekstas. Tuščias elemento turinys netalpina nei teksto, nei kitų elementų.

1.4.2. *RDF* kalba

Semantinio tinklo infrastruktūrai aprašyti *W3C* konsorciumas rekomenduoja *RDF* (angl. *Resource Description Framework*) kalbą [18]. *RDF* yra paprasta kalba, išreiškianti duomenų modelius, kurie nurodo į objektus (resursus) ir jų ryšius. Šia kalba paremtas modelis gali būti aprašytas *XML* sintaksėje. Duomenys aprašomi naudojant trilypes struktūras, kurios susideda iš subjekto, teiginio ir reikšmės aprašymo. Trilypės struktūros subjektas ir reikšmė gali būti nusakoma tame pačiame dokumente, taipogi gali nurodyti ir į kitą internete esantį dokumento šaltinį. Teiginiu gali būti bet koks *XML* kalboje aprašomas vardas. Naudojant šią technologiją galima aprašyti ir duomenis, kurių šaltinis yra ir kiti *RDF* duomenys. *RDF* modelyje naudojamas įvairių išraiškų rinkinys vartoja konkretų žodyną, kuris nusako ypatybes ir duomenų tipus.

RDF kalbą, papildyta *RDF* schema, galima vadinti riboto išraiškingumo žinių ontologija. Žemiau paveiksle (2 pav.) pavaizduota supaprastinta universitetų klasifikavimo pagal vietovę ontologija panaudojant bazinius *RDF* klasifikavimo ir priskyrimo atributus*: rdfs:label, rdf:type.*



2 pav. Universitetų ontologijos klasių taksonimija RDFS kalba

1.4.3. *OWL* kalba

Viena iš populiariausių ontologijų kalbų yra būtent *OWL* 2 (angl. *Web Ontology Language*) [1]. Ši kalba yra *RDF* kalbos plėtinys, paremtas RDF/XML struktūra. *OWL 2* palaiko tokius sintaksės formatus: *RDF/XML, OWL/XML, OWL Functional Syntax, Manchester OWL syntax* ir *Turtle*. *OWL* naudojama norint aiškiai pateikti žodynuose esančias išraiškas, jų prasmę bei tarpusavio ryšius. Tai vieningas žinių apie tai, kas egzistuoja realiame pasaulyje, pateikimo internete formatas, kitaip vadinamas ontologija.

OWL 2 turi 3 dialektus, besiskiriančius išraiškos galimybėmis: *OWL Lite, OWL DL, OWL Full,* žemiau pateikiama detalesnė informacija apie kiekvieną iš jų.

- *OWL Lite* – pritaikyta pačioms paprasčiausioms užduotims atlikti ir turi mažiausias išraiškos galimybes. *OWL Lite* naudojama tokioms reikmėms, kaip klasifikavimo hierarchijos ir paprasti apribojimai. Pavyzdžiui, nors ji palaiko kardinalumo ribojimus, bet suteikia jiems 0 arba 1 reikšmę. Dėl to šiai kalbos rūšiai yra žymiai lengviau kurti įrankius nei kitoms;
- *OWL DL* – tinkamiausia daugeliui projektuojamų ontologijų kadangi suteikia maksimalų ekspresyvumą, išlaikant skaičiavimų baigtumą ir sprendimo baigtumą (visi skaičiavimai bus atlikti per baigtinį laiką). *OWL DL* gali būti naudojami visi *OWL* kalbos dariniai, bet juos galima naudoti su kai kuriais ribojimais (pavyzdžiui, nors klasė gali būti kelių klasių poklasis, bet klasė negali būti kitos klasės atskiras atvejis). *OWL DL* pavadinimas išplaukia iš jo suderinamumo su deskriptyviąja logika (angl. *description logics*) – tyrimų srities, kuri nagrinėjo logiką nuo pat formalaus *OWL* sukūrimo;
- *OWL Full* – labiausiai išplėtotas dialektas, turintis geriausias žinių išraiškos priemones, tačiau neteikiantis garantijos dėl samprotavimų proceso baigtinumo. Deja, panašu, kad jokia samprotaujanti programinė įranga negalėtų visiškai pagrįsti kiekvienos *OWL Full* savybės.

Kiekvienas iš šių dialektų yra prieš tai buvusio plėtinys, todėl paprastesnėje kalbos rūšyje aprašyta ontologija taip pat bus galiojanti sudėtingesnėje.

Kalbant apie ontologijos sąvokas išskiriami pagrindiniai ontologijos elementai kartu su būdingomis specifinėmis savybėmis kurie sutelpa į *OWL* Lite:

- Klasės (angl. *Class*), kurios gali turėti poklasių (angl. *Sub Class Of*) aksiomas, kurios nusako ryšius tarp deklaruotų klasių. Taip pat šios klasės gali turėti ir nesusikertančių sąjungų (angl. *Disjoint Union*) aksiomas, kurios nusako, kad klasės negali būti susikertančiomis. Pavyzdžiui, turime vieną klasę „Moteris“, kitą klasę „Vyras“ ir jos sudaro apibendrinamą rinkinį (angl. *Generaliuzation Set*) kaip nesusikertančią sąjungą, tai reiškia, jog asmuo bus arba moteris arba vyras bet kartu vienu metu;
- Objektų savybės (angl. *Object property*), būdingos funkcinės objektų savybės (angl. *Functional Object Property*), kurios nurodo, kad savybė gali turėti tik viena unikalią reikšmę kiekvienam individui. Taip pat ir atvirkštinės objektų savybės (angl. *Inverse Object Property*) – kai viena objektų savybė deklaruojama kaip priešinga kitai. Specializuojančios objektų savybės (angl. *Sub Object Property Of*) nurodo hierarchinį ryšį, kuris reiškia vienos savybės buvimą kitos posavybę, bei domenai (angl. *Domain*) - nurodo ribojimą, kuriems individams savybė gali būti taikoma, ir sritys (angl. *Range*) - nurodo ribojimą, kuriuos individus savybė gali turėti kaip reikšmę;
- duomenų savybės (angl. Data *property*). Šioms savybėms būdingos sąvokos nusakančios domenus ir sritis, kurios atitinkamai nurodo ribojimą, kuriems individams savybė gali būti taikoma ir nurodo ribojimą, kuriuos individus savybė gali turėti kaip reikšmę. Taip pat ir specializuojančios duomenų savybės kurios nurodo hierarchinį ryšį, kuris reiškia vienos savybės buvimą kitos posavybe;
- individai (angl. *Individuals*) yra klasių egzemplioriai;
- aksiomos (angl. *Axioms*).

1.4.4. Semantinio tinklo technologijų analizės apibendrinimas

Analizės metu nustatyta svarbiausia ir aktualiausia tyrimo sričiai semantinio tinklo technologija - *OWL 2* formali ontologijų kalba kurios paskirtis yra aprašyti ontologijas. *OWL* ontologijų aprašymo kalba turi galimybę aprašyti savybes, klases, taip pat ir ryšius tarp jų. *OWL* palengvina automatiškai interpretuoti interneto turinį, lyginant su tuo ką gali *XML* ir *RDF*. Taip pat *OWL* teikia papildomas informacijos aprašymo galimybes ir formalią semantiką. Ši *OWL* ontologijų aprašymo kalba yra laikoma itin perspektyvia kuriant semantinį tinklą, todėl yra gana platus ontologijų kūrimo įrankių pasirinkimas.

1.5. Esamų ontologijos vizualizacijos ir transformacijos įrankių analizė

Ontologijų įrankių analizė atliekama norint išsiaiškinti geriausią sprendimą ontologijoms vizualizuoti ir transformuoti į *UML*. Įrankių analizės tikslas išanalizuoti „OWLGrEd“ [5], „Protégé“ [4] ir „MagicDraw“ ir remiantis analizės rezultatais įvertinti išanalizuotus įrankius taip išsirenkant tinkamiausią sprendimą.

1.6. *UML* kalbos analizė

UML yra modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus [9]. Ši kalba paremta grafiniu modeliavimu, kuri apima keturiolika diagramų rūšių bei turi vieningą terminologiją. Šiuo metu *UML* yra labiausiai paplitęs programinės įrangos specifikavimo standartas, Lietuvoje yra kuriama visame pasaulyje žinoma „MagicDraw“ - *UML CASE* priemonė. *UML* pagalba galima specifikuoti, vizualizuoti ir dokumentuoti programinės įrangos sistemų modelius – jų struktūrą ir projektus. *UML* kalbos privalumai:

- *UML* leidžia kitiems kūrėjams greitai suvokti jūsų sistemą;
- tai visuotinai pripažintas standartas;
- supaprastėja komunikacija, visi kalba ta pačia kalba, iššvaistoma mažiau laiko;
- *UML* leidžia aprašyti sistemą norimu detalumu ir norimais pjūviais;
- reikalavimai lengviau apibrėžiami ir dokumentuojami, mažiau pamirštų vietų;
- vartotojai įtraukiami į programos kūrimą nuo pat pradžių, mažiau perdarymų pabaigoje;
- lengva nustatyti projektavimo klaidas;
- priemonė išsaugoti sukauptas žinias firmoje, net jei žmonės ją palieka;
- sutaupo laiko susipažįstant su jau sukurtomis sistemomis.

UML kalba turi 14 skirtingų diagramų tipų, suskirstytų į dvi kategorijas: struktūrines ir elgsenos. Pirmąją kategoriją sudaro septynios diagramos kurios apibūdina modelio objektų struktūras:

1. Klasių diagrama (angl. *class*) - skirsto daiktus į kategorijas. Klasė aprašo daiktų grupę, kurie turi panašius atributus ir vienodą elgseną;
2. Komponentų diagrama (angl. *component*) - aprašo sistemos fizinį vaizdą;
3. Objektų diagrama (angl. *object*) - visiškai ar iš dalies atvaizduoja modelio struktūrą tam tikru laiko momentu;
4. Sudedamųjų struktūrų diagrama (angl. *composite structure*) - apibūdina vidinę klasės struktūrą ir galimas šios struktūros sąveikas;
5. Paketų diagrama (angl. *package*) - parodo sistemos vidinę organizaciją. Sistemos struktūrines dalis atitinka paketai, į juos dedami tų dalių modeliai, priklausomybės tarp paketų modeliuoja dalių sąryšius;
6. Profilių diagrama (angl. *profile*) - naudojama meta modelio lygmenyje ir parodo klasių stereotipus ir profilius;
7. Realizacijos diagrama (angl. *component, deployment*) – nusako sistemos komponentus ir parodo fizinę sistemos struktūrą.

Būsenas ir sąveikas apibūdina kitos septynios diagramos:

1. Veiklos diagrama (angl. *activity*) - apibūdina veiksmų sekas;
2. Sekos diagrama (angl. *sequence*) – apibūdina objektų sąveiką laike;

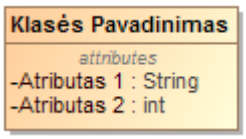
3. Panaudos atveju diagrama (angl. *usecase*) - apibūdina sistemos elgseną iš vartotojo pozicijų;
4. Būsenų diagrama (angl. *state*) - nusako objektų būsenas ir jų pasikeitimus laike;
5. Bendradarbiavimo diagrama (angl. *collaboration, communication*) - apibūdina objektų sąveiką;
6. Išdėstymo diagrama (angl. *deployment)* - aprašo fizinį sistemos diegimą;
7. Laiko diagrama (angl. *timing*) - specifinė diagrama, skirta apibūdinti laiko apribojimus.

1.7. Klasių diagramos analizė

Kadangi šio darbo tikslas yra sudaryti galimybę įtraukti ontologijų kūrimą į informacinių sistemų projektavimą, kur informacinių sistemų kūrimas yra modeliais grindžiamas ir yra naudojamos *UML* diagramos, todėl logiška, kad ir grafinis sprendimas turėtų būti paremtas *UML*, kuris leistu integracija su kitais *UML* modeliais kuriamais informacines sistemos kūrimo etape.

Šiuo atveju iš visų *UML* siūlomų diagramų aktualiausia yra klasių diagrama, kadangi ji gali pasiūlyti daug atitikmenų su ontologijų elementais jų vizualizavimui.

Pati *UML* klasių diagrama susideda iš stačiakampių – klasių, kurios apima atributus ir operacijas. Klasės pavyzdys pateiktas 3 pav.

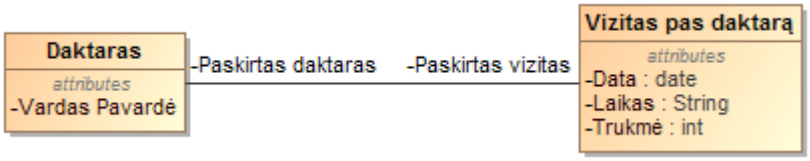


3 pav. Klasės pavyzdys

Taip pat klasių diagramoje naudojami ryšiai kurie gali būti kelių rūšių:

- asociacija;
- priklausomybė;
- generalizacija.

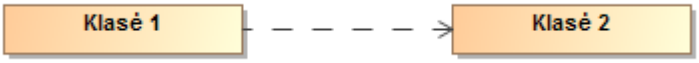
Asociacija skirta specifikuoti dvipusį ryšį tarp klasių. Asociacijos pabaiga yra vadinama vaidmeniu (angl. *role*) kuriam galima suteikti pavadinimą. Asociacijos vaidmenų pavyzdys pateiktas 4 pav. Daktaras vizite užima paskirto daktaro vaidmenį, o vizitas pas daktarą vaidina paskirto vizito vaidmenį.



4 pav. Asociacijos pavyzdys

Generalizacija tai ryšys, kuriame viena klasė yra kitos klasės konkretizacija (5 pav.). Priklausomybė tai vienkryptis ryšys, nurodantis, kad vienas elementas naudoja kitą elementą. Priklausomybės pavyzdys pateikiamas 6 pav.

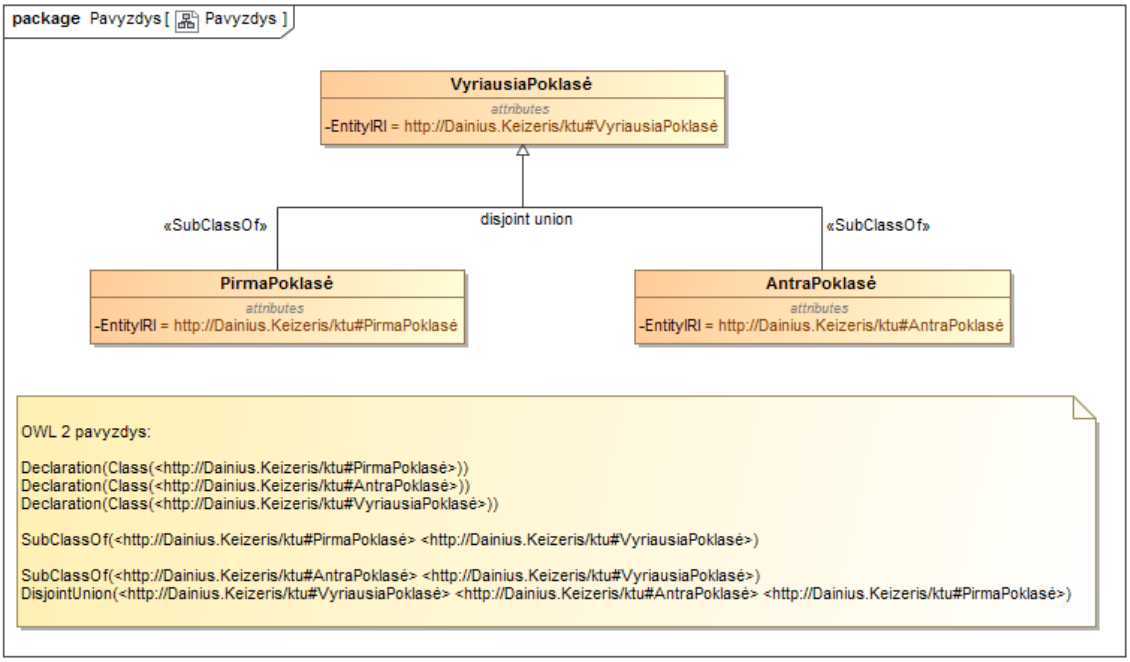
5 pav. Generalizacijos pavyzdys



6 pav. Priklausomybės pavyzdys

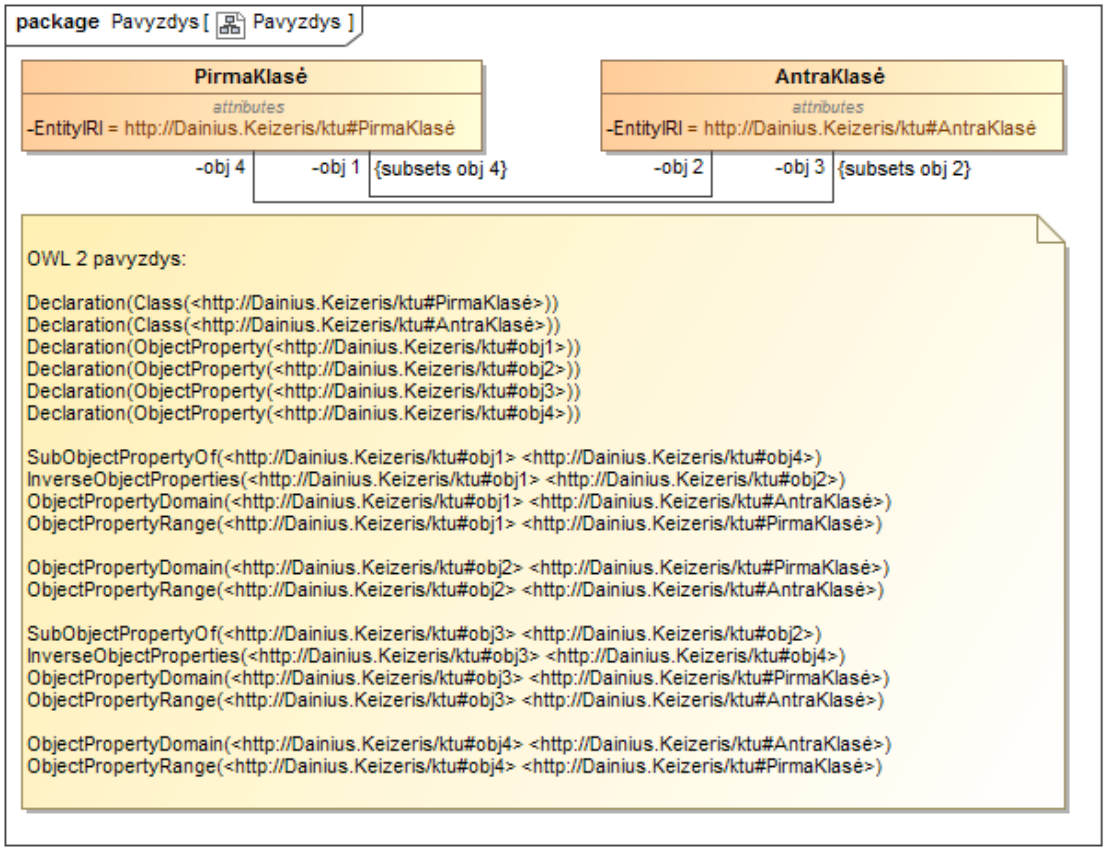
Taigi *UML* klasių diagrama turi pakankamai elementų atvaizduoti ontologijų elementus. Šiuo atveju mums aktualūs prieš tai analizėje jau minėti *OWL* Lite pagrindiniai komponentai: klasės, objektų savybės, duomenų savybės, individai ir visų jų aksiomos.

UML klasių diagrama leidžia atvaizduoti visus šiuos elementus panaudojus darbo metu sukurtą *OWL* profilį. Profilis stereotipų pagalba leidžia susieti *UML* klases su *OWL 2* ontologijų kalbos klasėmis per UML klasę su stereotipu „<<OWL 2 class>>“ iš *OWL* profilio. Poklasės vizualizuojamos naudojant generalizacijos (angl. *Generalization*) ryšį su *OWL* profilio „<<SubClassOf>>“ stereotipu. Nepersikertanti sąjunga (angl. *Disjoint Union*) *UML* ontologijos vizualizavime galima nurodyti sujungus du generalizacijos ryšius ir susidariusio bendro ryšio specifikacijoje nustačius „is disjoint“ reikšmę į „true “. Bendras klasių, poklasių ir nepersikertančios sąjungos pavyzdys parodytas 7 pav.



7 pav. Klasių ir klasės aksiomų pavyzdys *UML* ir *OWL 2*

Objektų savybėms nurodyti *UML* klasių diagramoje puikiai tinka asociacijų ryšys. Šiuo ryšiu sujungus du elementus ir uždėjus jam stereotipą „<<ObjectProperty>>“ iš sukurto *OWL* profilio, transformacijos metu jis bus atpažintas kaip objektų savybė. Asociacija patogi tuo, jog priešingai nei „Protégé OntoGraf“, leidžia nurodyti domeną (angl. *Domain*) ir sritį (angl. *Range*) ant vieno ryšio. Taip vizualizuota diagrama nebūna perkrauta pertekliniais ryšiais. Įskiepis taip pat geba nustatyti ar objektų savybė yra atvirkštinė. Objektų savybių ir jų aksiomų pavyzdys *UML* ir *OWL 2* parodytas 8 pav.



8 pav. Objektų savybių ir jų aksiomų *UML* ir *OWL 2* pavyzdys

Duomenų savybės *UML* klasių diagramoje prilygsta atributams. Atributo tipas nusako ar atributas yra „String“ tipo, ar „Integer“, ar „Boolean“ ir t.t.

Žemiau pateikiama apibendrinanti lentelė (Lentelė 1), kurioje pateikiamos taisyklės, kaip asocijuojasi *OWL 2* ir *UML* elementai.

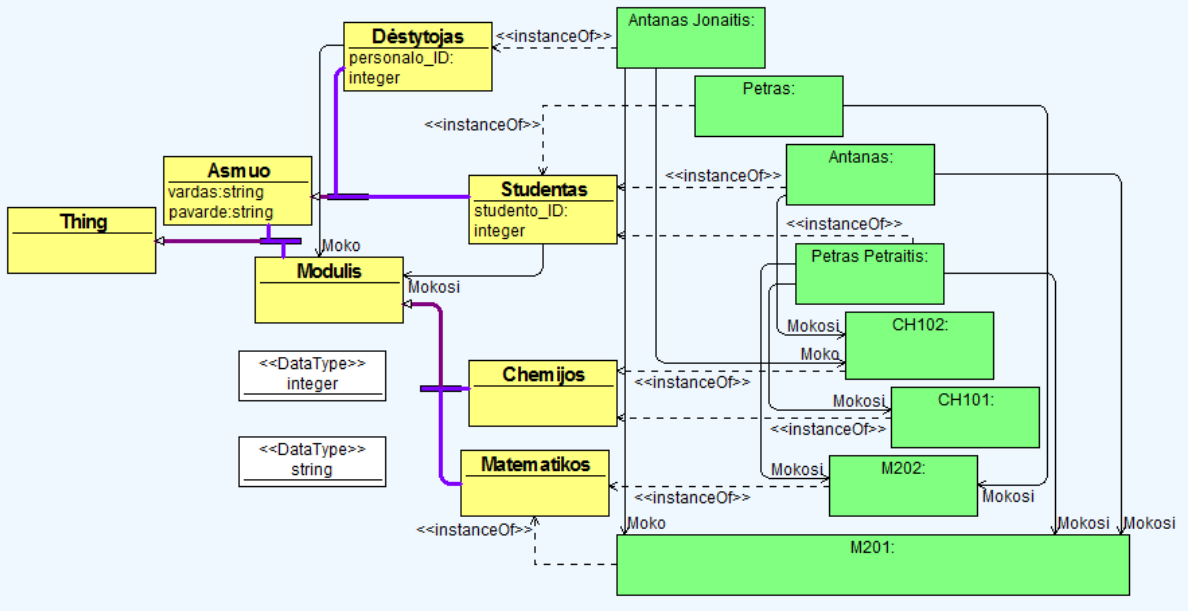
Lentelė 1. *OWL 2* ir *UML* sugretinti elementai

	***OWL 2* esybė**	**Atvaizdavimas *ULM***
1.	Annotation Property	UML class
2.	Annotation Property Assertion	Property of UML class
3.	Asymmetric Object Property	UML association constraint
4.	boolean	UML data type
5.	Class	UML class
6.	Class Assertion (the type of the Individual)	UML dependency with stereotype
7.	Data All Values From	UML data type
8.	Data Complement Of	UML datatype constraint
9.	Data Exact Cardinality	UML attribute multiplicity
10.	Data Intersection Of	UML datatype constraint
11.	Data Max Cardinality	UML attribute multiplicity
12.	Data Min Cardinality	UML attribute multiplicity
13.	Data One Of (enumeration of Individuals)	UML datatype constraint
14.	Data Property	UML attribute
15.	Data Property Assertion	UML slot value
16.	Data Property Domain	UML class of attribute
17.	Data Property Range	UML datatype
18.	Data Type Restriction	UML datatype constraint
19.	Data Union Of	UML attribute datatype
20.	Data Some Values From	UML attribute multiplicity 0..1
21.	Decimal	–
22.	Different Individuals	UML dependency with stereotype
23.	Disjoint Classes	UML dependency with stereotype
24.	Disjoint Data Properties	UML constraint on attribute
25.	Disjoint Object Properties	UML dependency with stereotype
26.	Disjoint Union	UML dependency with stereotype
27.	Entity IRI	Hidden

	***OWL 2* esybė**	**Atvaizdavimas *ULM***
28.	Equivalent Classes	UML dependency with stereotype
29.	Equivalent Data Properties	UML attribute constraint
30.	Equivalent Object Properties	UML association constraint
31.	Functional Object Property	UML association constraint
32.	Functional Data Property	UML attribute constraint
33.	Has Key	UML class constraint
34.	Imported ontology	–
35.	Inverse Functional Object Property	UML association constraint
36.	Inverse Object Property	UML association constraint
37.	Irreflexive Object Property	UML association constraint
38.	Language for Annotations	Part of UML class property, specifying Annotation Property Assertion
39.	Named Individual	UML object
40.	Negative Data Property Assertion	UML object slot value constraint
41.	Negative Object Property Assertion	UML object link constraint
42.	Object All Values From	UML association constraint
43.	Object Complement Of	UML class constraint
44.	Object Exact Cardinality	Multiplicity of Association End
45.	Object Has Self	UML class constraint
46.	Object Intersection Of	UML class constraint
47.	Object Max Cardinality	Multiplicity of Association End
48.	Object Min Cardinality	Multiplicity of Association End
49.	Object One Of	UML class constraint
50.	Object Property	UML association
51.	Object Property Assertion	UML link
52.	Object Property Domain	UML class
53.	Object Property Range	UML class
54.	Object Some Values From	UML constraint on class
55.	Object Union Of	UML constraint on class
56.	Ontology	UML package
57.	Reflexive Object Property	UML constraint on association
58.	Same Individuals	Dependency
59.	string	Datatype
60.	SubClass Of	Class specialization
61.	SubData PropertyOf	Attribute constraint
62.	SubObject Property Of	Association constraint
63.	Symmetric Object Property	Association constraint
64.	Transitive Object Property	Association constraint
65.	xsd:datetime	Datatype
66.	xsd:integer	Datatype
67.	xsd:nonnegative integer	Datatype
68.	xsd:positive integer	Datatype

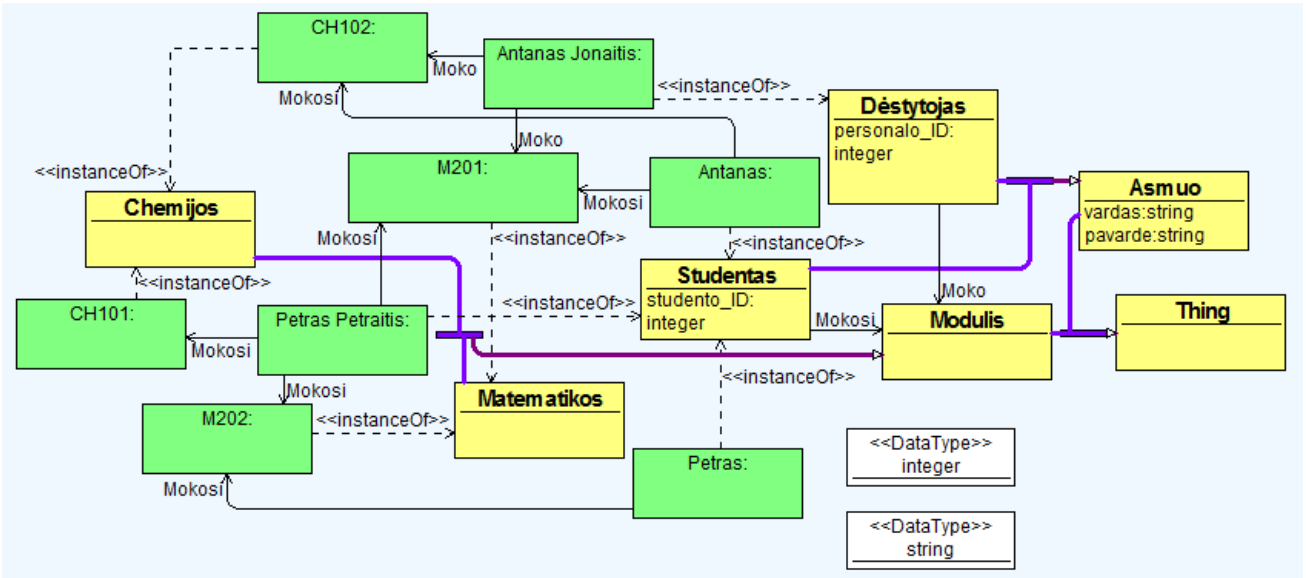
1.7.1. „OWLGrEd“ analizė

„OWLGrEd“ įrankis buvo pasirinktas analizei kadangi leidžia grafiniu būdu atvaizduoti, kurti, redaguoti bei išsaugoti ontologijas *UML* stiliumi. Šio įrankio analizės metu buvo nustatyta, kad pradinis ontologijos vaizdas atvaizduojamas ganėtinai tvarkingai (9 pav.), tačiau norint rankiniu būdu pakeisti schemą pagal savo poreikius į kitokį vaizdą yra susiduriama su nepatogumais ir sugaištama kur kas daugiau laiko keičiant ontologijos išdėstymą nei su „No Magic“ įmonės kuriamu produktu „MagicDraw“ ar „Protégé“.



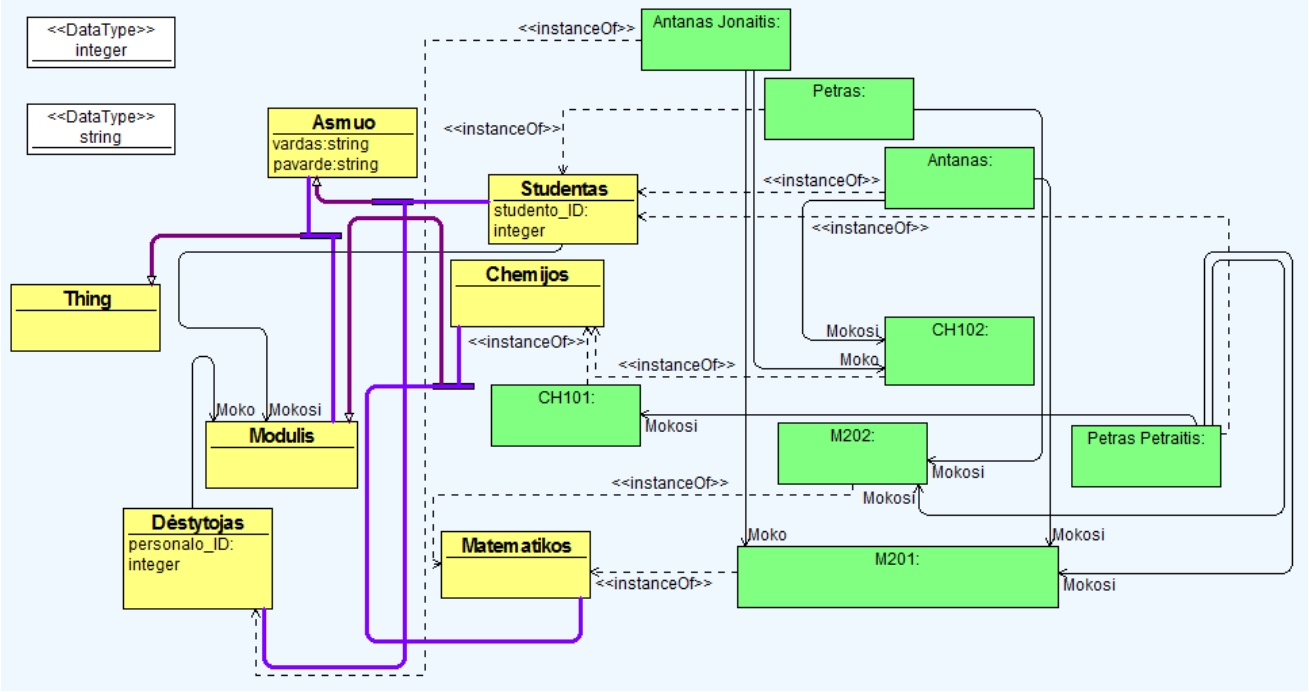
9 pav. „OWLGrEd“ vizualizuota ontologija

„OWLGrEd“ turi funkciją kuri leidžia pakeisti schemos atvaizdavimą į vertikalų arba horizontalų išdėstymą, tačiau išbandžius šią funkciją su vertikaliu ir horizontaliu atvaizdavimu sugrįžti atgal prie buvusio prieš funkcijos panaudojimą schemos vaizdo jau neturime galimybės. Schemoje atsiranda chaosas, nutrūkę ryšiai, bei vėl pakitęs išdėstymas. Adaptavus ontologiją gautas vaizdas pateikiamas 10 pav.



10 pav. „OWLGrEd“ vizualizuojama adaptuota ontologija

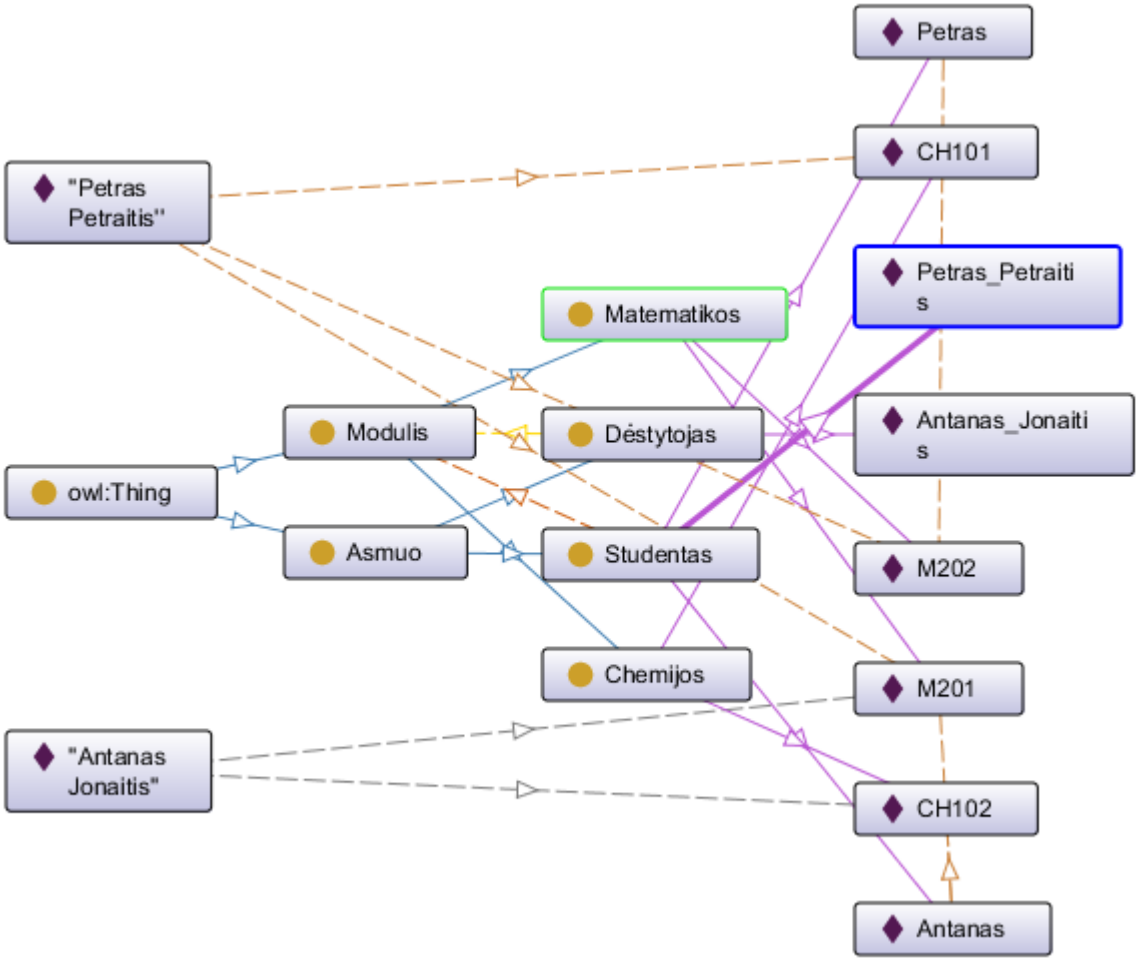
Adaptavus pavyzdinę ontologijos schemą labiausiai užkliuvo nelankstūs ryšiai, perkeliant klases ryšiai darosi sunkiai kontroliuojami. Ryšius reikia kilnoti atskirai, kadangi jie lieka savo vietose, o tai užima daug papildomo laiko. Patys ryšiai yra labai nelankstūs, norint sutvarkyti ryšį jį gali tekti keisti per kelias vietas, o jei dar ryšys priklausomas ir nuo kitų ryšių tada situacija tampa dar sudėtingesnė, kadangi pajudinus vieną ryšį ankstesnių vieta schemoje gali taip pat pasikeisti. Ryšių problemos pavyzdys pavaizduotas 11 pav. kuomet yra perkeliamos kelios klasės, o ryšiai lieka savo vietoje.



11 pav. „OWLGrEd“ vizualizuojamos ontologijos ryšiai

1.7.2. „Protégé“ analizė

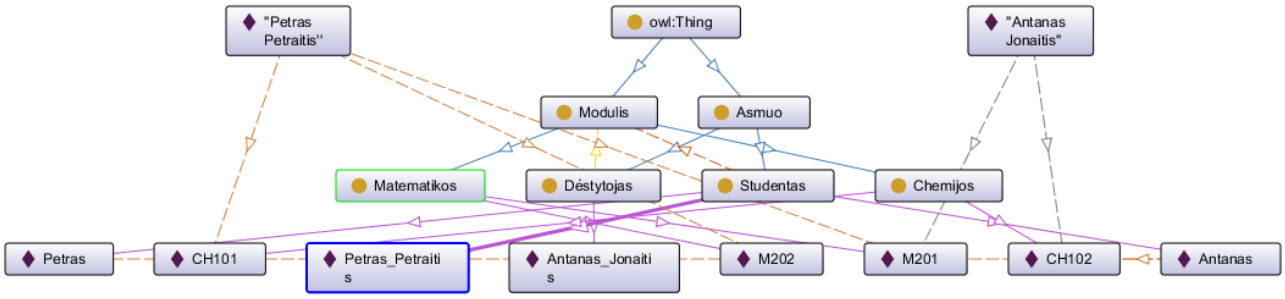
Naudojant „OntoGraf“ įskiepį „Protégé“ įrankis turi galimybę grafiškai atvaizduoti ontologijas. „OntoGraf“ įskiepis yra skirtas tik ontologijos atvaizdavimui bet ne jos kūrimui ar redagavimui. Pradinis išskleistos ontologijos vaizdas parodytas 12 pav. Ontologija yra gražiai išdėstoma grafiškai ryšių sąskaita. Tai reiškia, kad klasių hierarchija išdėstoma taip, kad visos klasės išsirikiuotų pagal paveldimumą, tačiau dėl to nukenčia ryšiai, kurie pradeda pintis bei uždengti vienas kitą. Įrankyje galima išdėstyti ontologiją rankiniu būdu pagal poreikius, tačiau išsaugoti schematiškai galimybės nėra. Ontologija „Protégé“ įrankyje pradedama vaizduoti suskleista - rodoma tik bazinė klasė su galimybe ją išskleisti. Perstumdžius ir adaptavus ontologiją į patogesnį vaizdą, norit išskleisti ar suskleisti papildomai dar bent vieną klasė, ontologijos schemos vaizdas yra iš naujo išrikiuojamas pagal klasių hierarchiją, o tai reiškia, kad reiks iš naujo rankiniu būdu adaptuoti schemą kaskart kai išskleidžiame ar suskleidžiame nors vieną klasę.



12 pav. „OntoGraf“ vizualizuota ontologija

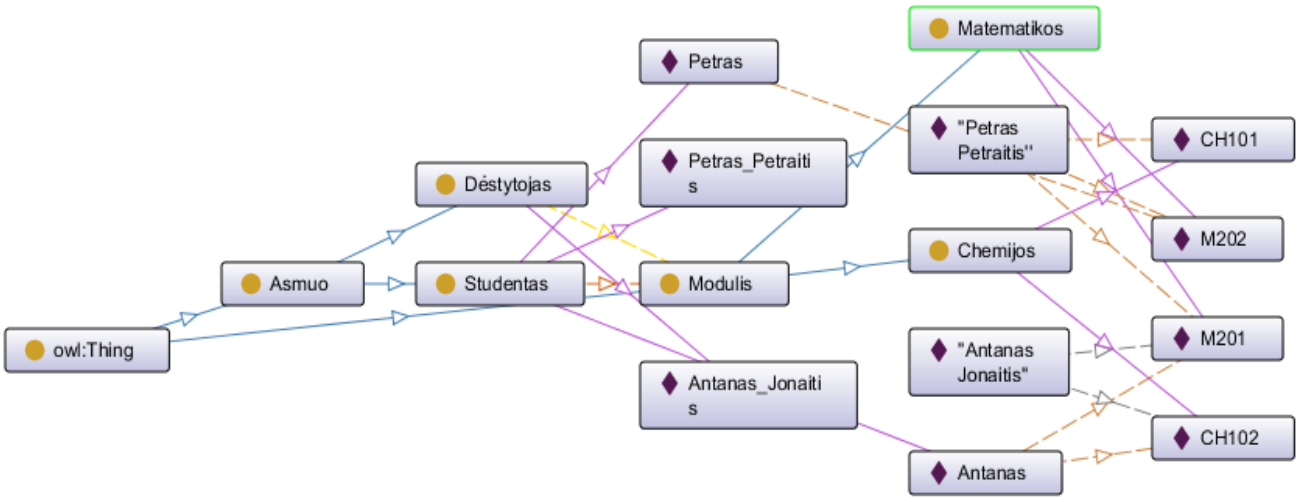
Įskiepis leidžia pasirinkti kokiu būdu ontologiją išdėstyti schematiškai, tačiau ir tai neišsprendžia ryšių problemos, kadangi visais atvejais schema rikiuojama pagal klasių hierarchiją. Žemiau esančiame paveikslėlyje (13 pav.) pateikiamas schemos atvaizdavimo pavyzdys pasirinkus galimą automatinį rikiavimo metodą iš įskiepio siūlomų.

Dar vienas aktualus trūkumas šiame įrankyje - atvirkštinės objektų savybės kurios yra dvipusės tačiau atvaizduojamos atskirtai. Dėl šių atvirkštinių objektų savybių sprendimo išauga ryšių kiekis, kadangi vietoje vieno ryšio atvaizduojami du. Dėl šios priežasties schema tampa sunkiai skaitoma, apkrauta pertekliniais duomenimis. Šiame darbe naudojamame „MagicDraw“ įrankyje ši problema išspręsta atvirkštines objektų savybes atvaizduojant ant to paties ryšio galų.



13 pav. „OntoGraf“ vizualizuota ontologija pakeitus rikiavimo metodą

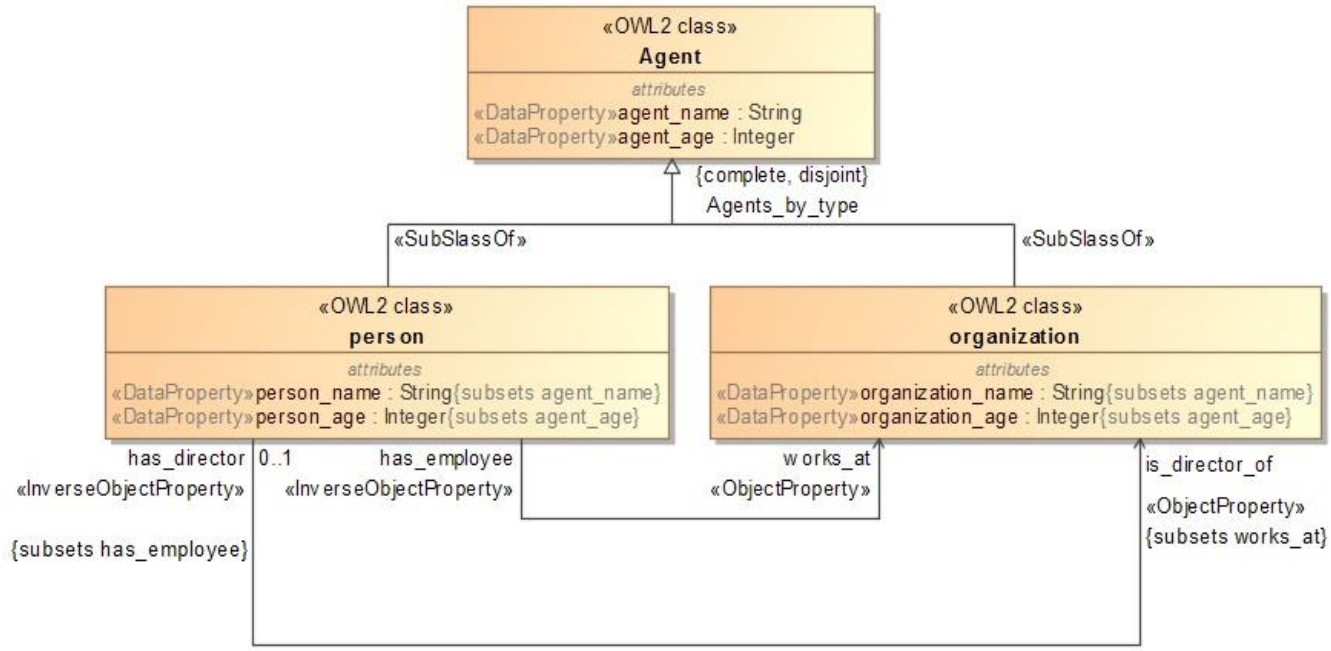
Išbandžius šį ontologijų vizualizavimo sprendimą galima teigti jog tai nėra labai patogus įrankis schematiškai atvaizduoti ontologijas, tačiau ignoruojant kai kurias ryšių problemas ontologiją gan nesunkiai galima adaptuoti į skaitymui patogesnę schemą, kaip parodyta 14 pav.



14 pav. „OntoGraf“ vizualizuota diagrama perstumdyta skaitymo patogumui

1.7.3. „MagicDraw“ įrankio analizė

„MagicDraw“ įrankis buvo pasirinktas analizei kadangi palaiko *UML* klasių diagramą. Šio įrankio analizės metu buvo nustatyta, kad galima grafiškai kurti, redaguoti ir išsaugoti *UML* diagramas. Įrankis turi modeliais grindžiamų sistemų projektavimo galimybę taip pat įrankyje galima inversines objektų savybes atvaizduoti vienu ryšiu. Žemiau paveikslėlyje (15 pav.) pavaizduotas *UML* klasių diagramos vaizdas. „MagicDraw“ įrankio analizės metu nustatytas trūkumas aktualus darbo sričiai – nėra galimybės transformuoti *OWL 2* ontologijos į *UML* klasių diagramą.



15 pav. „MagicDraw“ vizualizuota klasių diagrama

1.8. Esamų ontologijos vizualizacijos ir transformacijos įrankių analizės apibendrinimas

Analizuotiems ontologijos vizualizacijos ir transformacijos įrankiams buvo nustatyti palyginimo kriterijai:

1. Galimybė redaguoti ontologiją diagramoje;
2. Galimybė kurti ontologiją braižant diagramą;
3. Galimybė išsaugoti ontologijos diagramą;

4. *UML* klasių diagramų palaikymo galimybė;
5. Galimybė vizualizuoti inversines objektų savybes vienu ryšiu;
6. Modeliais grindžiamų sistemų projektavimo galimybė;
7. Galimybė vizualizuoti *OWL 2* ontologiją.

Lentelė 2. Esamų sprendimų palyginimas

Palyginimo kriterijus	**„Protégé“**	**„OWLGrEd“**	**„MagicDraw“**
Grafinis redagavimas	-	+	+
Grafinis kūrimas	-	+	+
Grafiniai elementai objektų savybėms	Du ryšiai	Vienas ryšys	Vienas ryšys
UML klasių diagramų palaikymas	-	+-	+
Grafinio projekto išsaugojimas	-	+	+
Modeliais grindžiamas sistemų projektavimas	-	-	+
OWL 2 ontologijos vizualizavimas *UML*	-	+	-

Palyginus įrankius pagal įvardintus kriterijus (Lentelė 2) galima daryti išvadą, jog dėl savo savybių tinkamiausias įrankis iškeltos problemos sprendimui yra „MagicDraw“. Problemos sprendimui reikia sukurti *OWL* profilį, kuris savyje turėtu visus reikalingus stereotipus ontologijos pavaizduotos *UML* klasių diagrama susiejimui su ontologijų kalba *OWL 2*.

1.9. Darbo tikslas, uždaviniai ir siekiami privalumai

Šio darbo tikslas yra sukurti transformavimo įskiepio prototipą leidžiantį automatizuoti ontologijų vaizdavimą *UML CASE* įrankiuose, *OWL 2* profilį ir ontologijų transformavimo į *UML* profilį taisykles ir algoritmus.

Šiam tikslui pasiekti iškelti uždaviniai:
1. Išanalizuoti:
   1.1. Ontologijų kalbos *OWL 2* sąvokas, redagavimo įrankius;
   1.2. *UML* kalbą, *CASE* įrankius;
   1.3. Panašius sprendimus ir tyrimus, aprašytus mokslinėse publikacijose ir elektroninėje erdvėje;
   1.4. Pasirinkti tinkamiausią sprendimą ir technologines priemones.
2. Sudaryti transformavimo taisykles ir algoritmus, suprojektuoti transformavimo įrankį;
3. Realizuoti įrankio prototipą, jį ištestuoti;
4. Atlikti eksperimentą, kuris leistų įvertinti sprendimo tinkamumą ir efektyvumą;
5. Apibendrinti tyrimo rezultatus.

1.10. Siekiamo sprendimo apibrėžimas

Norint panaudoti ontologijas informacinių sistemų kūrime yra labai svarbu turėti įrankį leidžiantį projektuoti sistemą ir pačias ontologijas. Ontologijų inžinieriai daug sėkmingiau suderintų vienodą požiūri į dalykinę sritį jeigu naudotų grafines priemones ontologijoms vizualizuoti. Ontologijų perkėlimas į *UML* leistu sukurtus modelius transformuoti ir į kitas formas, taip pat susieti ontologiją su kuriamos sistemos reikalavimais ar sistemos architektūra.

Siekiamas galutinis rezultatas yra „MagicDraw“ aplinkoje veikiantis įskiepis kurio pagalba *OWL 2* kalba aprašyta ontologija būtų transformuota į *UML*.

1.11. Analizės išvados

1. Atlikus analizę buvo išanalizuoti pagrindiniai *OWL 2* kalbos elementai kurie turi asociaciją su *UML* elementais ir kuriuos reikėtų transformuoti - klasės, objektų savybės, duomenų savybės, individai bei jų visų aksiomos – domenai, duomenų tipai, sritys, atvirkštinės savybės,

nepersikertančių elementų sąjungos, poklasės, specializuojančios objektų ir duomenų savybės nes visi šie išvardinti baziniai elementai yra visuose kalbos lygiuose.

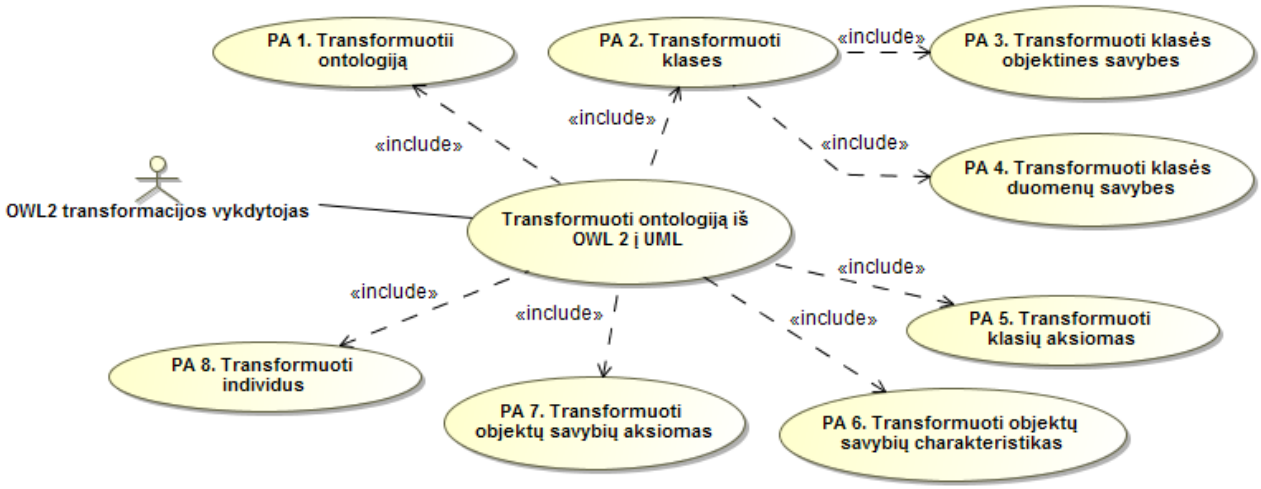
2. Atlikus įrankių analizę ir palyginus juos pagal išsikeltus kriterijus, problemai išspręsti buvo pasirinktas „MagicDraw“ įrankis. Vienas pagrindinių kriterijų lėmusių sprendimą buvo modeliais grindžiamo sistemų projektavimo galimybė. Tai suteikia galimybę įtraukti ontologijų kūrimą į informacinių sistemų projektavimą.

2. ONTOLOGIJOS TRANSFORMACIJOS ĮSKIEPIO SPRENDIMO REIKALAVIMŲ SPECIFIKACIJA

2.1. Reikalavimų specifikacija

Ontologijos transformavimo įskiepio funkciniai reikalavimai aprašomi panaudojimo atvejų diagrama Įskiepis turėtų nuskaityti *OWL 2* kalba aprašytą ontologijos failą, atlikti eilę vidinių transformacijų į ontologijos metamodelio (toliau - OM) - klases, objektines savybes, duomenų savybes, klasių aksiomas, duomenų tipus, objektų savybių charakteristikas, objektų savybių aksiomas. Pagrindinė įskiepio savybė – *OWL 2* ontologijos transformavimas į ontologijos modelį OM.

UML diagrama modeliuojama naudojant „No Magic“ kompanijos kuriamu įrankiu „MagicDraw“, darbo metu buvo naudota 18.4 versija. *UML* modeliavimui naudojamas magistrinio darbo metu sukurtas *OWL* profilis. Žemiau pateiktoje panaudojimo atvejų diagramoje (16 pav..) matoma sukurto įskiepio apimtis.



16 pav. Transformacijos įskiepio panaudojimo atvejų diagrama

Žemiau esančiose 2-8 lentelėse pateikiami kiekvieno atvejo aprašymai.

Lentelė 3. PA 1 aprašymo lentelė

ID	PA 1
Pavadinimas	Transformuoti ontologiją
Aprašymas	Įskiepis transformuoja ontologijos aksiomą į ontologijos diagramos paketą.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	Ontology(<*IRI*>)
UML atitikmuo	Paketas su ontologijos stereotipu ir *IRI*

Lentelė 4. PA 2 aprašymo lentelė

ID	PA 2
Pavadinimas	Transformuoti klases
Aprašymas	Įskiepis transformuoja klases kurios turi stereotipą „<<OWL 2 class>>“ ir unikalų IRI į *UML* klasių diagramą.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	Declaration(Class(IRI))
UML atitikmuo	*UML* klasių diagramos klase su stereotipu ir IRI

Lentelė 5. PA 3 aprašymo lentelė

ID	PA 3
Pavadinimas	Transformuoti klasės objektines savybes
Aprašymas	Įskiepis transformuoja ontologijos kalvos objektų savybes į asociacijos ryšius su jų rolėmis.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	Declaration(ObjectProperty(*IRI*))
UML atitikmuo	Asociacija su rolėmis ir objektų savybių stereotipu.

Lentelė 6. PA 4 aprašymo lentelė

ID	PA 4
Pavadinimas	Transformuoti klasės duomenų savybes
Aprašymas	Įskiepis transformuoja *OWL 2* kalboje aprašytas duomenų savybes į *UML* klasių diagrama su klasėse aprašytais atributais kurie turi stereotipą „<<DataProperty>>“.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	Declaration(DataProperty(*IRI*))
UML atitikmuo	Klasės atributas su stereotipu ir *IRI*

Lentelė 7. PA 5 aprašymo lentelė

ID	PA 5
Pavadinimas	Transformuoti klasių aksiomas
Aprašymas	Įskiepis transformuoja poklasių aksiomas į poklasių ryšius kurie turi stereotipą „<<SubClassOf>>“ bei nepersikertančių elementų sąjunga (angl. *disjoint union*) į generalizacijos ryšius.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	DisjointUnion(*IRI1 IRI2 IRI3*) SubClassOf(*IRI1 IRI2*)
UML atitikmuo	Generalizacijos rinkinys nustatytas kaip nepersikertančių elementų sąjunga. Generalizacija su poklasės stereotipu.

Lentelė 8. PA 6 aprašymo lentelė

ID	PA 6
Pavadinimas	Transformuoti objektų savybių charakteristikas
Aprašymas	Įskiepis transformuoja *OWL 2* objektų savybės į asociacijos ryšius su jų rolėmis.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	Declaration(ObjectProperty(*IRI*))
UML atitikmuo	Asociacija su rolėmis ir objektų savybių stereotipu.

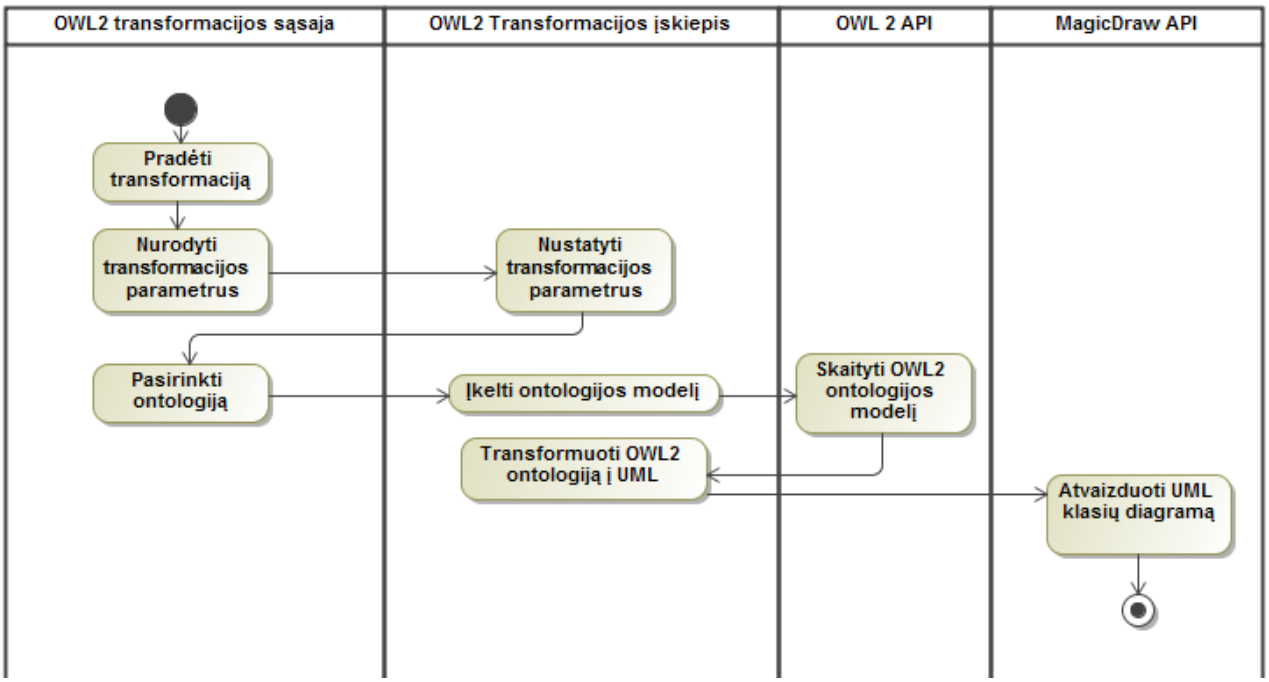
Lentelė 9. PA 7 aprašymo lentelė

ID	PA 7
Pavadinimas	Transformuoti objektų savybių aksiomas
Aprašymas	Įskiepis transformuoja funkcines objektų savybes į kardinalumus, objektų savybių domenus ir sritis į asociacijų ryšių roles, atvirkštines objektų savybes į atvirkštines asociacijų ryšių roles bei specializuojančias objektų savybes į specializuojančių asociacijų ryšius.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	ObjectPropertyDomain(*IRI1 IRI2*) ObjectPropertyRange(*IRI1 IRI2*) SubObjectPropertyOf(*IRI1 IRI2*) InverseObjectProperties(*IRI1 IRI2*) FunctionalObjectProperty(*IRI*)
UML atitikmuo	Asociacijos ryšio rolės poaibis. Atvirkštinė asociacijos ryšio rolė. Asociacijos ryšio rolė su „<<Functional>> “ stereotipu.

Lentelė 10. PA 8 aprašymo lentelė

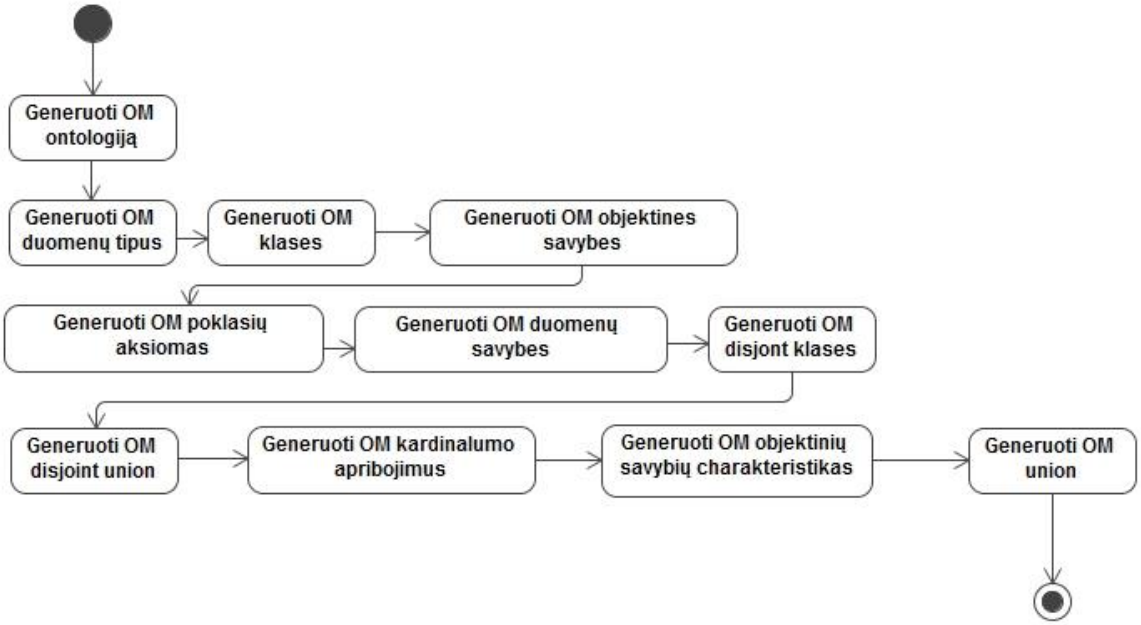
ID	PA 8
Pavadinimas	Transformuoti individus
Aprašymas	Įskiepis transformuoja individus *OWL* ontologijų kalboje į egzempliorius.
Aktoriai	*OWL 2* transformacijos vykdytojas
OWL atitikmuo	Declaration(NamedIndividual(IRI))
UML atitikmuo	Egzempliorius su stereotipu ir aprašytomis individo reikšmėmis.

Panaudojimo atvejis „Transformuoti ontologiją *OWL 2* į *UML*“ (16 pav.) šioje schemoje funkcionuoja taip: projektuotojas inicijuoja veiksmą *OWLToUML*. Atidaromas *OWL 2* ontologijos bylos pasirinkimo langas. Transformavimo iš *OWL 2* į ontologijos modelį OM valdiklis iškviečia Transformacijos parametrų valdiklį, kuris atidaro transformacijos parametrų įvedimo langą. Projektuotojas nustato paketo bei diagramos pavadinimus. Transformavimo iš *OWL 2* ontologijos į ontologijos modelį OM valdiklis kreipiasi į valdiklius: transformavimo į OM ontologiją, transformavimo į OM klases, transformavimo į OM klasės objektines savybes, transformavimo į klasės duomenų savybes, transformavimo į OM klasių aksiomas transformavimo į OM objektų savybių aksiomas, transformavimo į objektų savybių charakteristikas. Atitinkami modelių valdikliai įgyvendina transformacija.



17 pav. Transformavimo įskiepio veikimo modelis

Ontologijos transformavimo algoritmas (18 pav.) prasideda nuo ontologijos modelio generavimo tuomet sugeneruojami duomenų tipai su klasėmis. Sekančiu etapu yra generuojamos objektinės, duomenų savybės, poklasių aksiomos ir tik tuomet yra generuojamos *disjoint* klasės, *disjoint union, union*, kardinalumo apribojimai, objektinių savybių charakteristikos.



18 pav. Transformavimo procesas, pavaizduotas *UML* veiklos diagrama

2.1.1. Transformacijos įskiepio nefunkciniai reikalavimai

Transformacijos įskiepiui buvo iškelti šie nefunkciniai reikalavimai:

1. Sistema turi veikti patikimai, t.y. ontologijos failo klaidingi duomenys neturi nutraukti įskiepio darbo;
2. Įskiepis turi veiki visose operacinėse sistemose kurias palaiko „MagicDraw“ modeliavimo įrankis;
3. Įskiepis privalo suprantamai informuoti kūrėją kokius duomenis suvesti;
4. Įskiepis turi būti lengvai atnaujinamas;
5. Įskiepis turi atlikti transformacija neprarandant duomenų bei sumodeliuoti *UML* diagramą;
6. Įskiepio vartotojo sąsaja turi būti draugiška informatyvi kūrėjui;
7. Transformavimas turi trukti kiek įmanoma trumpiau.

2.2. Ontologijos transformacijos įskiepio reikalavimų apibendrinimas

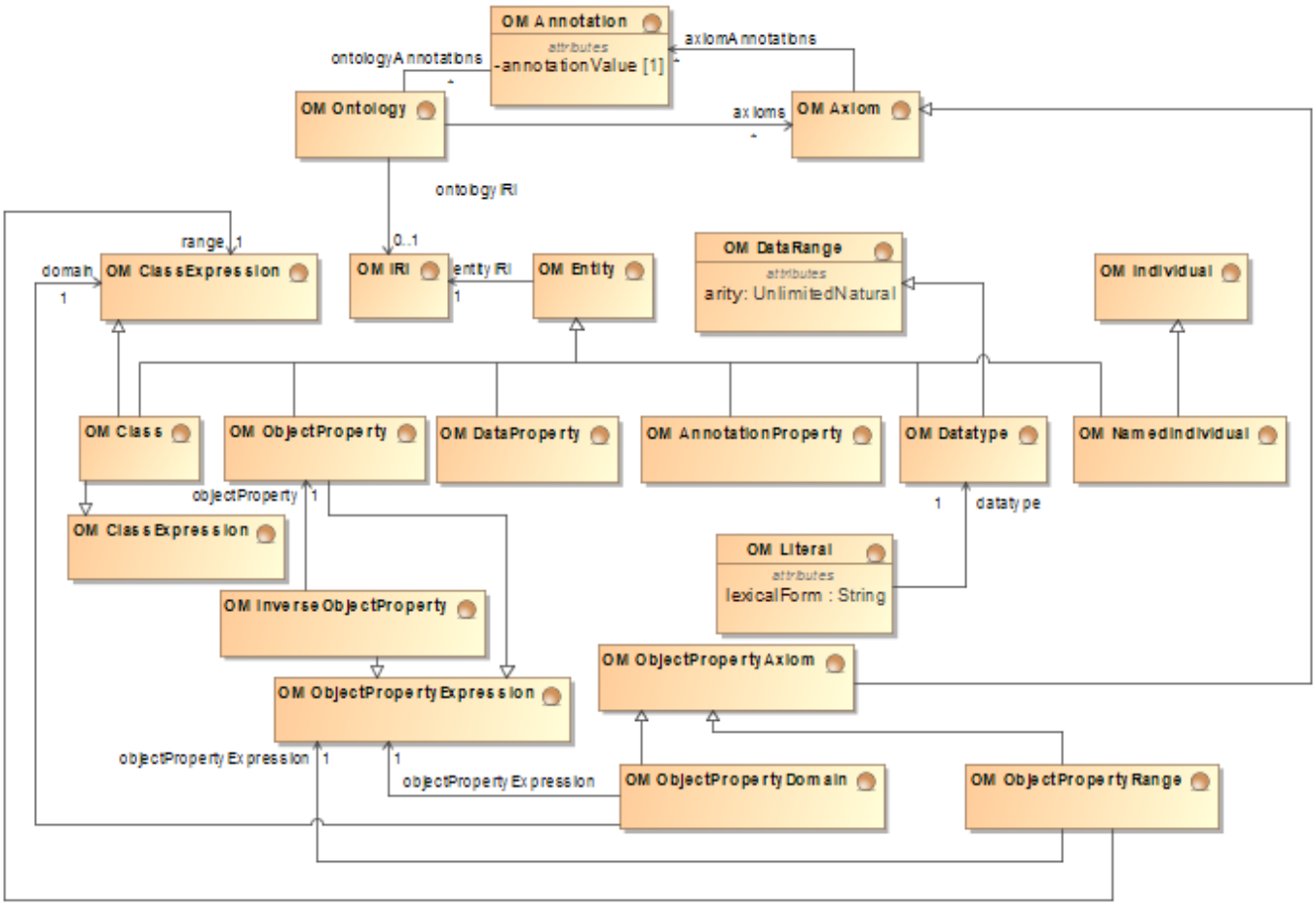
Apibendrinant reikalavimus įskiepis turi veikti „MagicDraw“ modeliavimo įrankyje, nepriklausomai kokia yra operacinė sistema, kurią palaiko modeliavimo įrankis. Įskiepis turi gebėti „MagicDraw“ aplinkoje pasirinkus *OWL 2* ontologijos kalba aprašytą failą transformuoti į *UML* elementus.

3. ONTOLOGIJOS TRANSFORMACIJOS ĮSKIEPIO PROJEKTAS

Projekto tikslas yra sukurti „MagicDraw“ įskiepį, kuris leistų transformuoti ontologijos failą aprašytą *OWL 2* kalba į *UML* elementus „MagicDraw“ aplinkoje egzistuojantį projektą.

3.1. *OWL 2* metamodelis

Transformavimo algoritmo aprašymui naudosime *OWL 2* metamodelį. Žemiau pateikiami (19 pav.) *OWL 2* metamodelio elementai. Sukurtas įskiepis padengia pagrindinius elementus, kuriuos turint galima plėsti ontologiją toliau.



19 pav. Pagrindiniai *OWL 2* metamodelio elementai

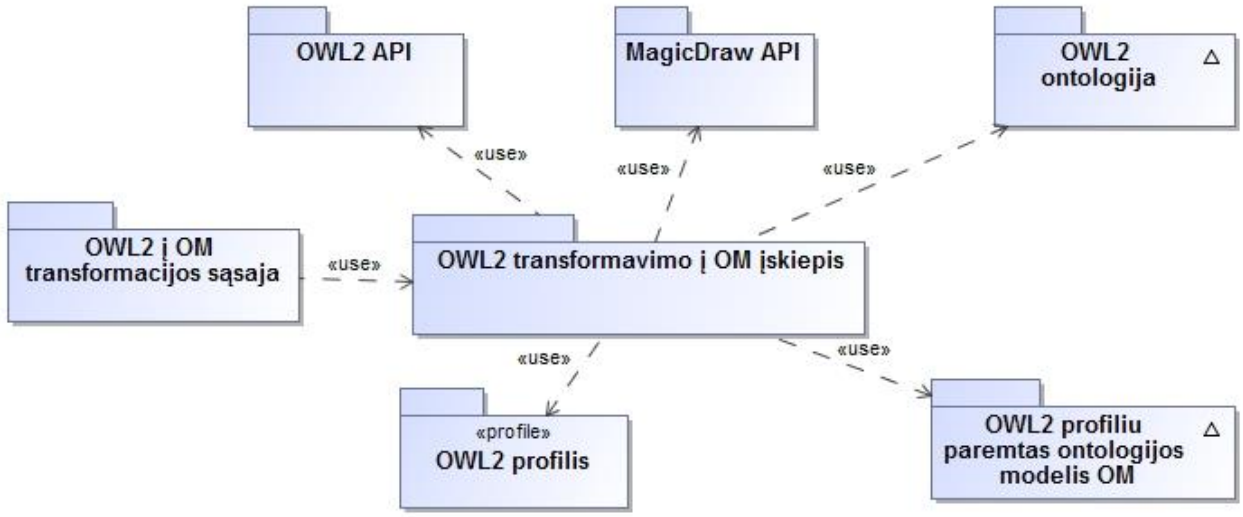
3.2. Sistemos architektūra ir realizacijos modeliai

Realizuotas transformavimo įskiepis turi susidėti iš vartotojo sąsajos modulio ir *OWL 2* transformacijos į *UML* modelio. Vartotojo sąsajos modulis skirtas palaikyti ryšį tarp projektuotojo ir modeliavimo aplinkos „MagicDraw“, leidžia atlikti veiksmus: pasirinkti transformacijos langą, šiame pasirinkti ontologijos *OWL 2* aprašo failą, įvesti paketo ir diagramos pavadinimus. *OWL 2* transformacijos modulis nagrinėja ontologijos objektus, apdoroja ir transformuoja į atitinkamus objektus *UML CASE* įrankio aplinkoje.

Tam, kad paprasčiau atlikti veiksmus su ontologijos objektais transformavimo įskiepis naudoja *OWL 2 API* bibliotekas. „MagicDraw“ aplinkoje programuoti įskiepi naudojama „MagicDraw“ *Open API*.

Pradinis transformavimo rezultatas yra saugomas objektinėje klasių struktūroje. Objektinė struktūra yra paverčiama į *UML*.

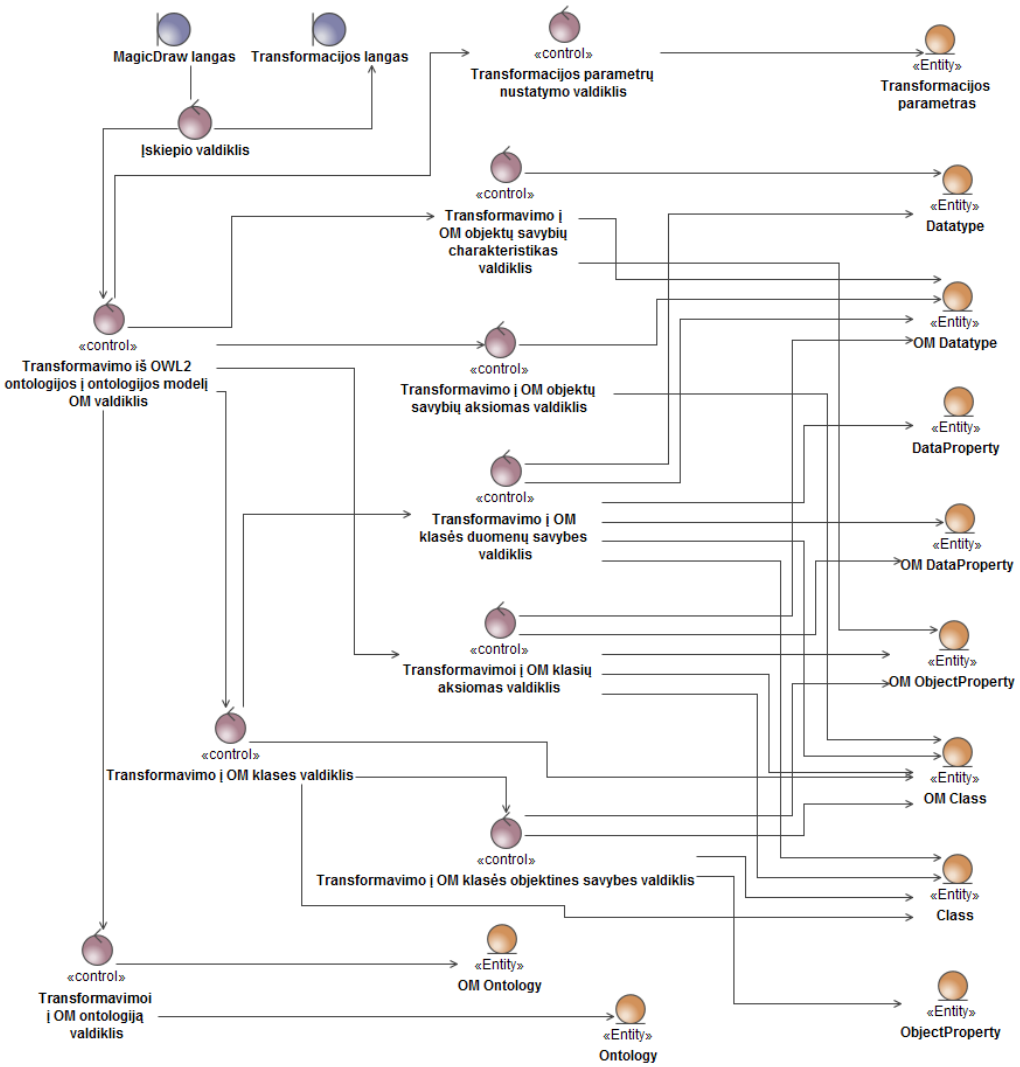
Žemiau (20 pav.) pateikiama loginė architektūra, kurią sudaro „MagicDraw“ *API*, *OWL 2* ontologija, *OWL 2* profilis ir *OWL 2* profiliu paremtu ontologijos modelis ir *OWL 2 API* loginiai elementai.



20 pav. Transformacijos įskiepio loginė architektūra

3.3. Reikalavimų analizės modelis

Transformavimo įskiepiui keliamiems reikalavimams, konceptams sudarytas panaudojimo atvejų analizės klasių modelis – robastiškumo diagrama (21 pav.)



21 pav. Ontologijos transformacijos analizės klasių diagrama

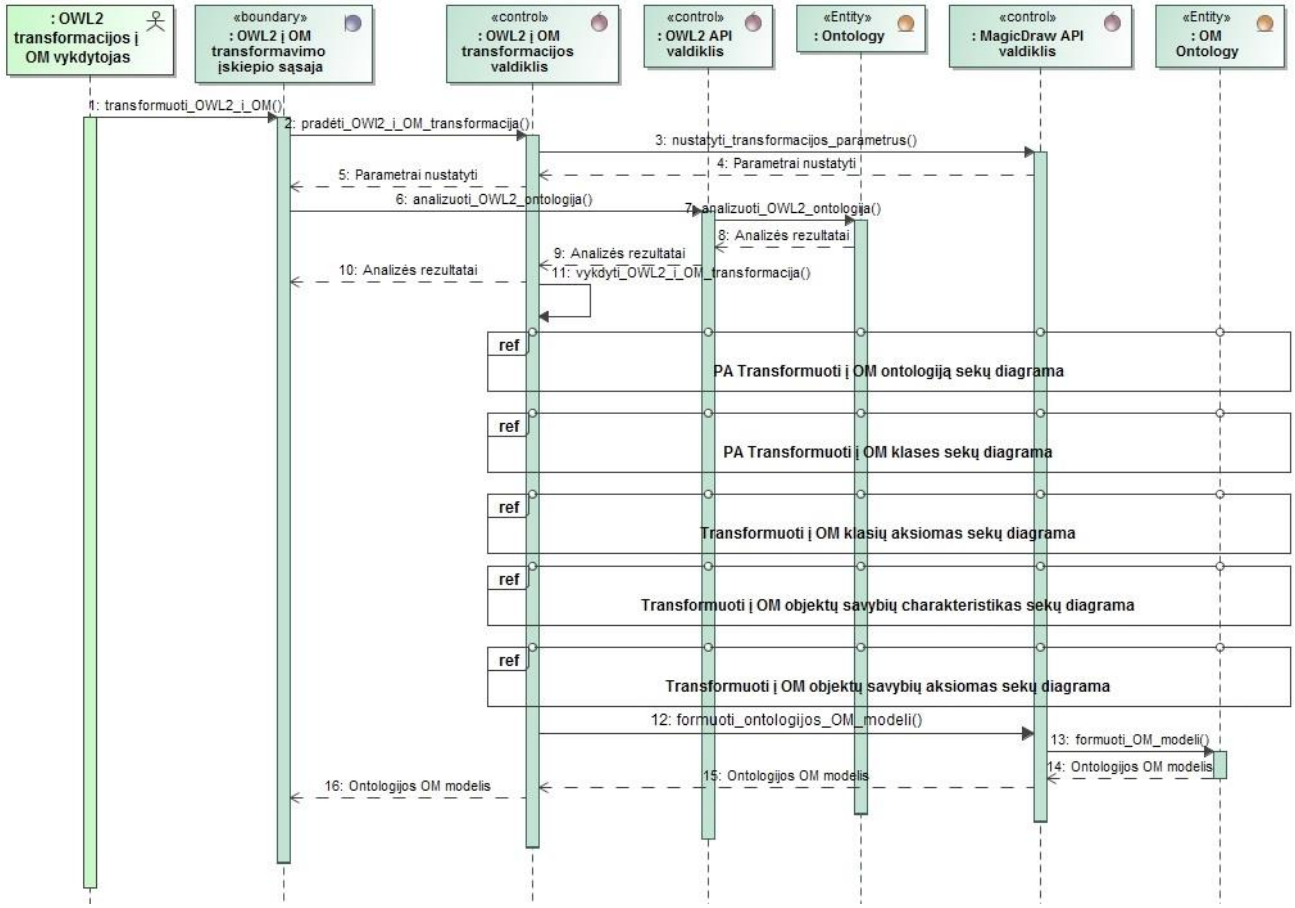
Diagramoje (21 pav.) galima matyti, kad MagicDraw lange vartotojas gali iškviesti įskiepio transformacijos langą, dėka įskiepio valdiklio. Transformavimo *OWL 2* i *UML* valdiklis komunikuoja su atskirais valdikliais kurie yra atsakingi už atskirus elementų transformavimo tipus.

3.4. Sistemos veiksena

Transformavimo įskiepio elgsena modeliuojama sekų diagramomis (22 pav. - 27 pav.). Analizuojant panaudojimo atvejų modelį (16 pav.) bei panaudojimo specifikacijose (Lentelė 3 - Lentelė 10) aprašytus žingsnius, nustatytos analizės klasių operacijos bei detalizuota sąveika tarp valdiklių ir vartotojo sąsajos komponentų.

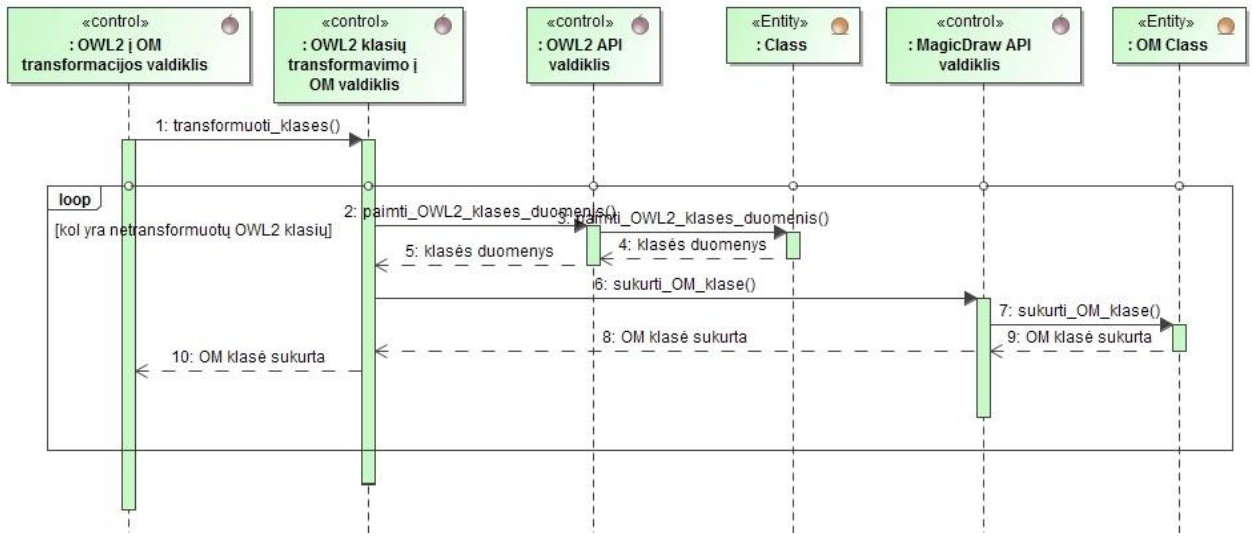
Panaudojimo atvejo „Transformuoti *OWL 2* ontologiją į ontologijos modeli OM“ realizacijai pateikiama sekų diagrama (22 pav.)

Sekų diagramoje (22 pav.) matoma IS kūrėjo, įskiepio, valdiklių , ontologijos objektų sąveika ir jų elgsena transformacijos situacijoje. Informacinės sistemos kūrėjas kreipiasi į įskiepio sąsają, įkeliamas ontologijos failas. Įskiepio sąsaja savo ruožtu kreipiasi į transformacijos valdiklį, kuris tarpininkaudamas tarp *OWL 2 API* ir „MagicDraw“ *API* valdiklių nustato reikiamus transformacijos parametrus, atlieka analizę bei vykdo *OWL 2* į OM transformaciją.

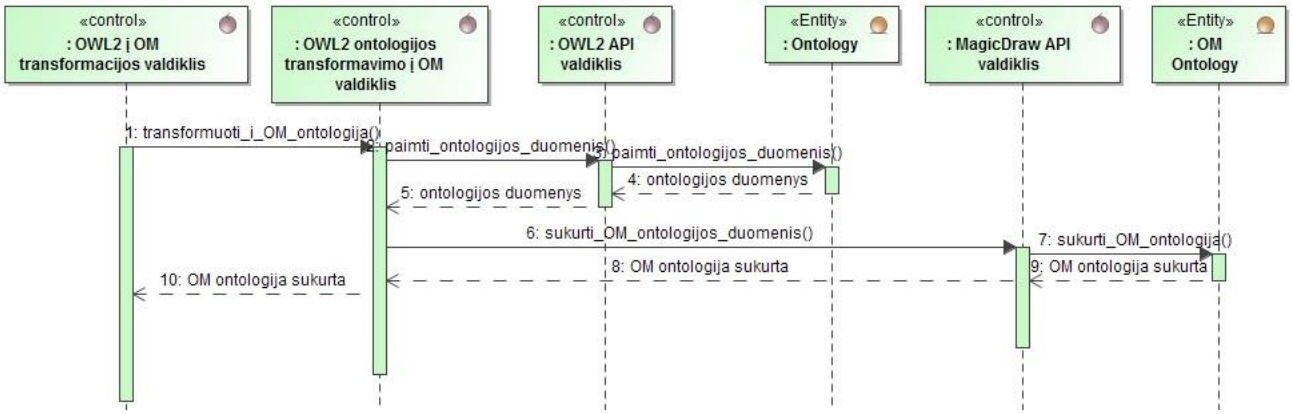


22 pav. PA Transformuoti *OWL 2* ontologiją į ontologijos modeli OM sekų diagrama

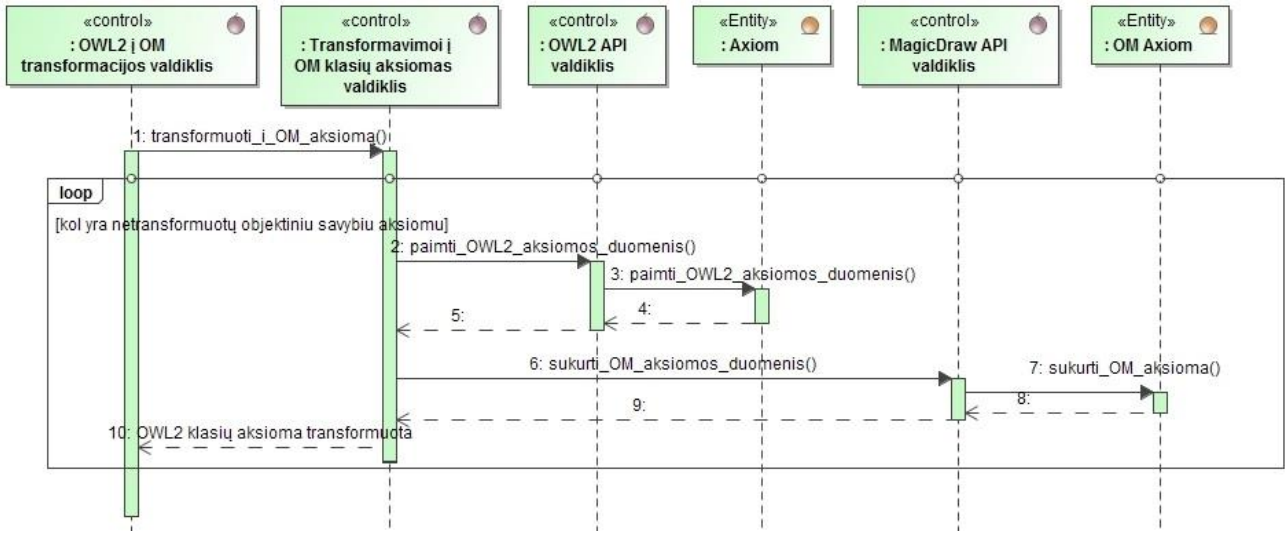
Sekų diagrama aprašo pagrindinį transformacijos įskiepio elgsenos scenarijų, kurį sudaro kiti panaudojimo atvejų elementai: „Transformuoti į OM ontologiją“, „Transformuoti į OM klases“, „Transformuoti į OM klasių aksiomas“, „Transformuoti į OM objektų savybių charakteristikas“, „Transformuoti į OM objektų savybių aksiomas“. Projektuotojui inicijavus veiksmą „OWLToUML“ ir pasirinkus ontologijos *OWL* bylą, transformacijos įskiepio valdiklis kreipiasi į „*OWL 2* į OM transformacijos valdiklį“ kuris nuosekliai inicijuoja transformacijos panaudojimo atvejus: Transformuoti į OM klases (23 pav.), Transformuoti į OM klasių aksiomas (25 pav.), Transformuoti į OM objektų savybių charakteristikas (27 pav.), Transformuoti į OM objektų savybių aksiomas (26 pav.)



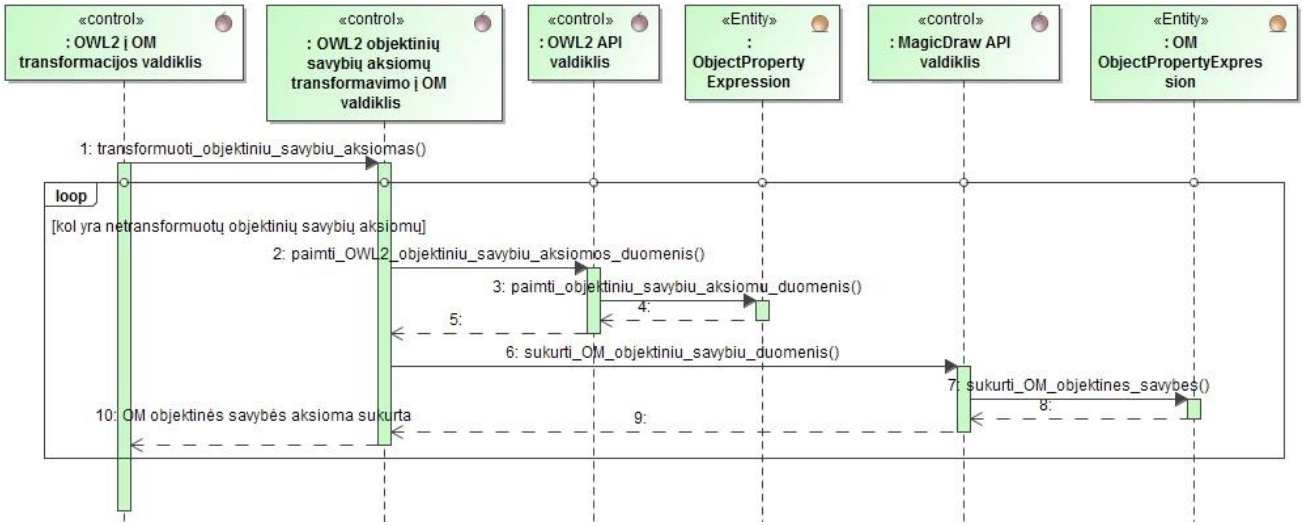
23 pav. PA Transformuoti į OM klases sekų diagrama



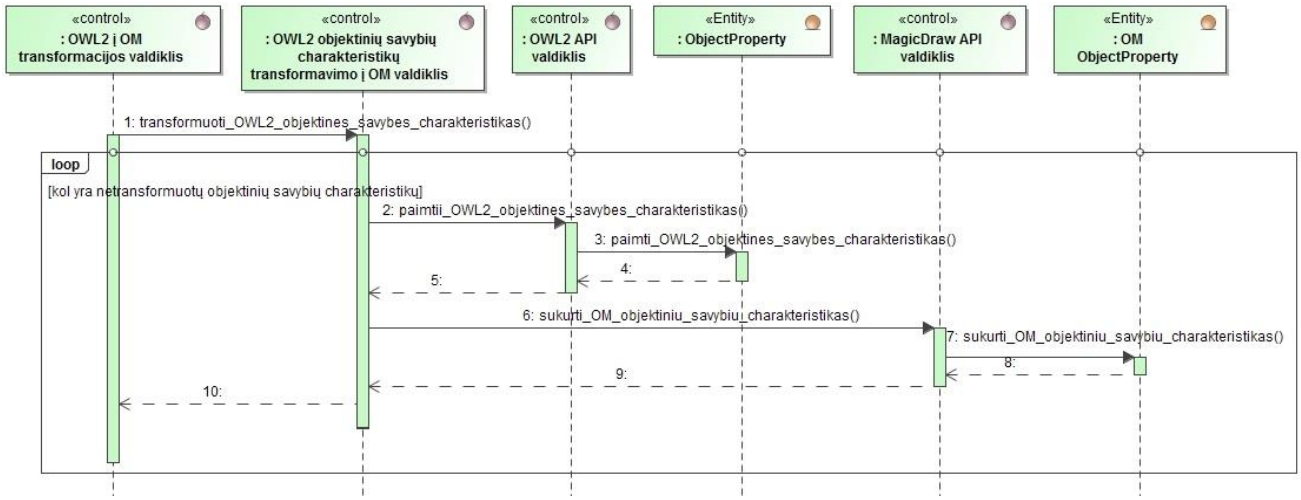
24 pav. PA Transformuoti į OM ontologiją sekų diagrama



25 pav. Transformuoti į OM klasių aksiomas sekų diagrama



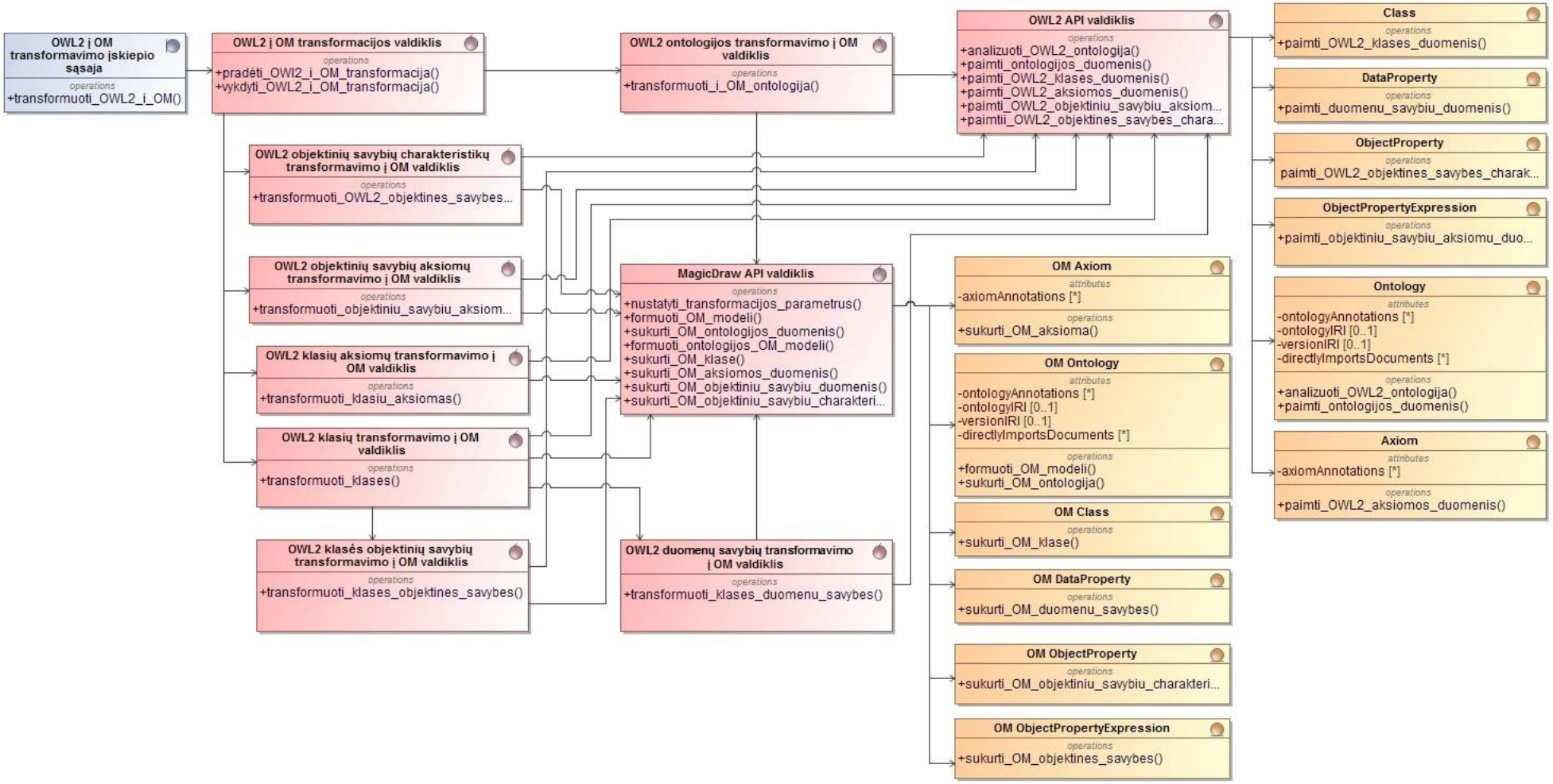
26 pav. Transformuoti į OM objektų savybių aksiomas sekų diagrama



27 pav. Transformuoti į OM objektų savybių charakteristikas sekų diagrama

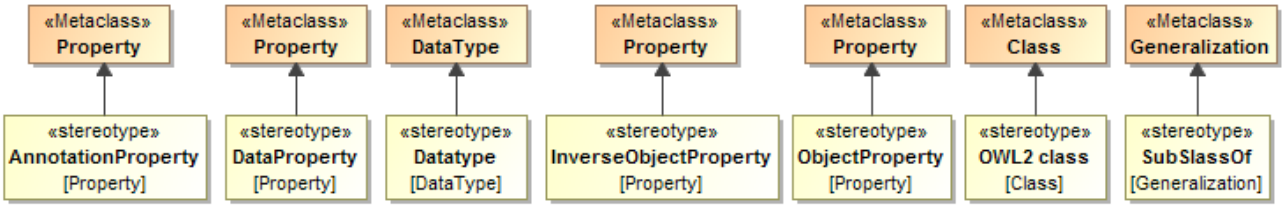
3.5. Transformacijos įskiepio detalus projektas

Trijų lygių transformavimo įskiepio klasių diagrama (28 pav.) galima išskirti vartotojo sąsajos komponentą (pažymėtas <<*boundary*>> stereotipu), veiklos komponentus (pažymėti <<*control*>> stereotipu), duomenų komponentus (<<*entity*>>).



28 pav. Transformavimo įskiepio klasių diagrama

Projektui sudarytas minimalus transformacijos *UML* profilis (14 pav.) kuriame detalizuojama kokie *OWL 2* elementai į kokius *UML* elementus bus transformuojami. Profilis padengia pagrindinius *OWL Lite* elementus, kurie yra pamatiniai *OWL 2* elementai. Šį profilį naudosime kuriant prototipą.



29 pav. *UML* profilis

Įskiepiui atliktus transformaciją turi būti sukurti *UML* elementai „MagicDraw“ aplinkoje pasirinktame projekte. Sudaryta lentelė (Lentelė 11), kurioje pavaizduotos transformavimo taisyklės, matoma *OWL 2* elementų sąsaja su *UML* elementais.

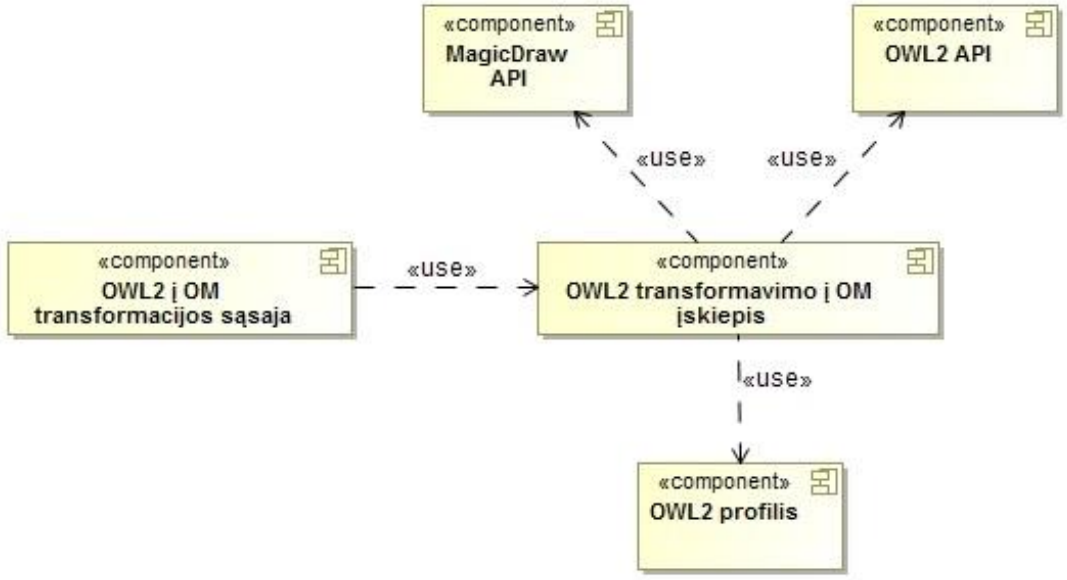
Lentelė 11. Transformavimo taisyklės

	OWL 2 esybė	**OWL 2 profilyje šiame darbe**
1.	boolean	*UML* data type Boolean
2.	Class	*UML* class
3.	Class Assertion (the type of the Individual)	*UML* Classifier
4.	Data All Values From	*UML* data type
5.	Data Exact Cardinality	*UML* attribute multiplicity
6.	Data Max Cardinality	*UML* attribute multiplicity
7.	Data Min Cardinality	*UML* attribute multiplicity
8.	Data Property	*UML* attribute (property)
9.	Data Property Domain	*UML* class in attribute’s specification
10.	Data Property Range	Type in attribute’s specification
11.	Data Union Of	Is derived union = truein attribute’s specification
12.	Data Some Values From	*UML* attribute multiplicity 0..1
13.	Different Individuals	*UML* dependency with stereotype
14.	Disjoint Classes	*UML* dependency with stereotype, GeneralizationSet with disjoint subclasses
15.	Disjoint Object Properties	*UML* dependency with stereotype
16.	Disjoint Union	Is Disjoint=true, Is Covering=true in Specification of Generalization Set
17.	Entity IRI	Tagged value
18.	Equivalent Classes	*UML* dependency with stereotype
19.	Functional Object Property	*UML* association constraint
20.	Functional Data Property	*UML* attribute constraint
21.	Has Key	*UML* stereotype
22.	Inverse Object Property	Opposite property
23.	Irreflexive Object Property	Property constraint
24.	Named Individual	Object
25.	Object Exact Cardinality	*UML* multiplicity of Association End
26.	Object Max Cardinality	*UML* multiplicity of Association End
27.	Object Min Cardinality	*UML* multiplicity of Association End
28.	Object Property	*UML* association
29.	Object Property Assertion	*UML* link
30.	Object Property Domain	*UML* class
31.	Object Property Range	*UML* class
32.	Ontology	*UML* package
33.	Reflexive Object Property	Property constraint
34.	Same Individuals	*UML* dependency
35.	string	*UML*datatype
36.	SubClass Of	*UML* class specialization
37.	SubDataPropertyOf	Subsetted property in attribute specification
38.	SubObject Property Of	Subsetted property in property specification
39.	Symmetric Object Property	Property constraint
40.	Transitive Object Property	Property constraint
41.	xsd:datetime	*UML*datatype
42.	xsd:integer	*UML*datatype
43.	xsd:nonnegative integer	*UML*datatype
44.	xsd:positive integer	*UML*datatype

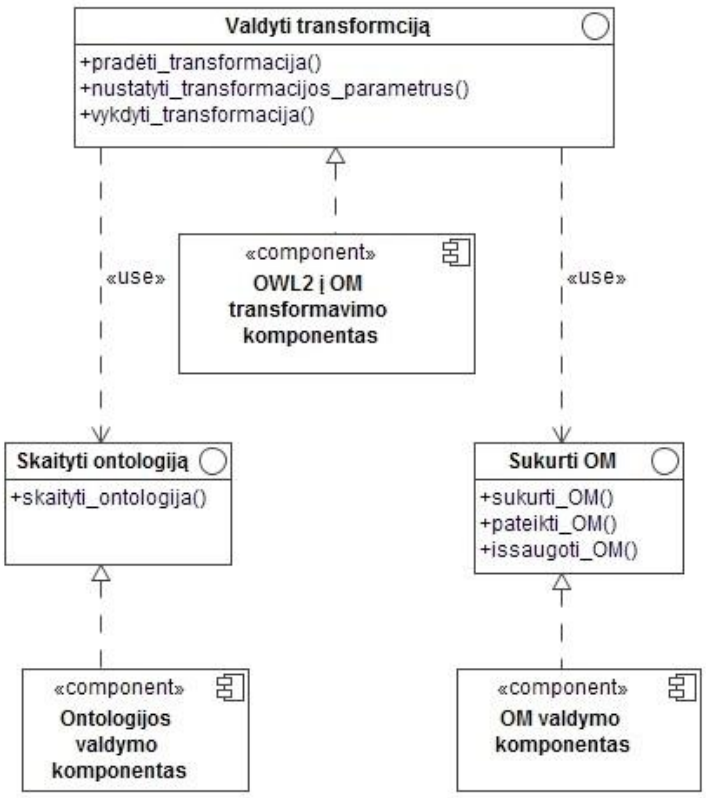
3.6. Transformacijos įskiepio realizacijos modelis

„MagicDraw“ įskiepio komponentų diagramoje (30 pav.) atsispindi ryšiai tarp transformavimo įskiepio ir reikiamu komponentų:

- „MagicDraw“ *API – UML* elementams ir diagramai sukurti;
- *OWL API* [8] - *OWL 2* ontologijos failo nuskaitymui ir apdorojimui;
- *OWL 2* ontologija;
- *OWL 2* profilis;
- transformacijos sąsaja.



30 pav. Transformacijos įskiepio komponentų diagrama



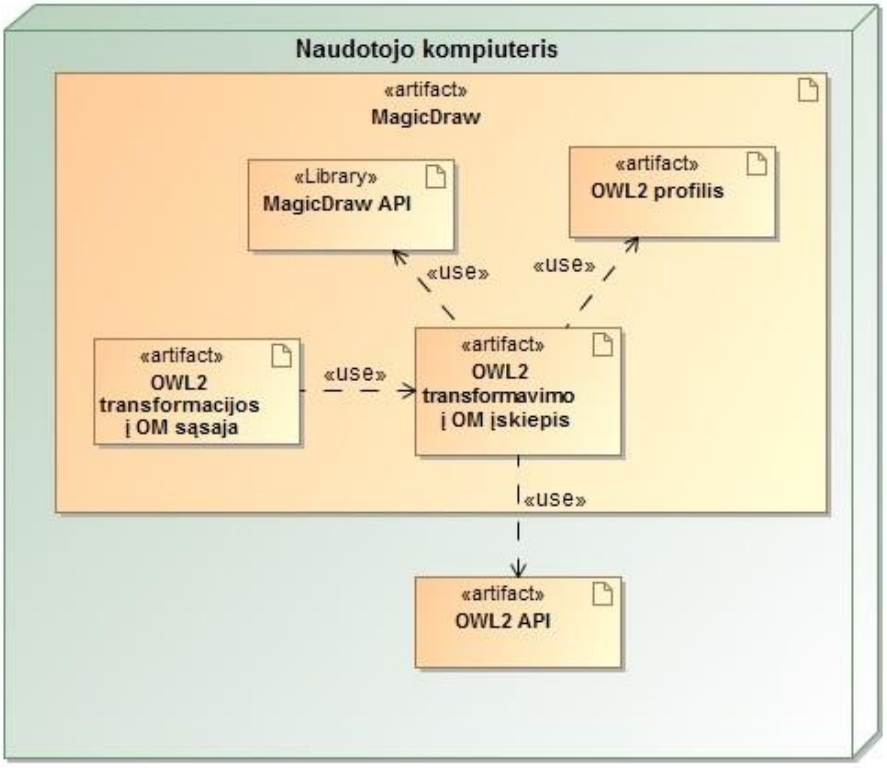
31 pav. Komponento „*OWL 2* transformavimo į OM įskiepis“ specifikacija

OWL 2 transformavimo į *UML* įskiepio komponento sandara pavaizduota komponentu specifikacijoje (31 pav.). Pagrindinis komponentas – valdyti transformaciją, kuris susideda iš metodu:

- *pradėti_transformaciją();*
- *nustatyti_transformacijos_parametrus();*
- *vykdyti transformacija().*

Pirmiausia vykdomas metodas *pradėti_transformaciją()*, patikrinamas ar pasirinktas failas yra *OWL 2* ontologijos kalba aprašytas failas toliau pagal nuoseklumą vykdomas *nustatyti_transformacijos_parametrus()* metodas kuris leidžia įvesti norimą *UML* paketo pavadinimą. Įvedus paketo pavadinimą vykdomas metodas – *vykdyti_transformacija()* kurio metu nuskaitoma informacija iš OWL 2 kalba aprašyto failo *skaityti_ontologiją()* metodo pagalba.

Įskiepio diegimo diagramoje (32 pav.) pavaizduotas transformavimo įskiepis ir šiam įskiepiui reikalingi atributai: „MagicDraw“ modeliavimo priemonė, „MagicDraw“ *API*, *OWL 2* profilis, *OWL 2 API*. Reprezentatyvus pavyzdys (49 pav.), kurį panaudosime testavimui ir duomenų transformavimo teisingumui nustatyti.



32 pav. Transformacijos įskiepio diegimo diagrama

4. ONTOLOGIJOS TRANSFORMACIJOS ĮSKIEPIO SPRENDIMO REALIZACIJA IR TESTAVIMAS

Pritaikant sukurtą ontologijos transformavimo metodiką, galima realizuoti įskiepį, kuris automatizuotu ontologijos transformacija į *UML*. Realizuojant turėtų būti sukurtas „MagicDraw“ įskiepio prototipas, kuris leistų į pasirinkta projektą įkelti transformuojamos ontologijos byla ir atlikti transformaciją *UML CASE* įrankyje. Šiame įrankyje po atliktos transformacijos būtų pavaizduotas transformacijos gautas rezultatas diagramoje.

4.1. Sprendimo realizacijos ir veikimo aprašas

Transformacijos įskiepiui realizuoti buvo pasirinkta Java platforma. Pagrindinės pasirinkimo priežastys: Java platformos portabilumas, Java kalbos paprastumas, saugumas, patikimumas, suderinamumas su „MagicDraw“ - programinės įrangos paketas taip pat realizuotas ant Java platformos ir yra pateikiamas dokumentacijos aprašas (API). Java ir C#.NET programavimo aplinkos palyginimas pateikiamas žemiau lentelėje (Lentelė 12).

Lentelė 12. Java ir C#.NET programavimo aplinkų palyginimas

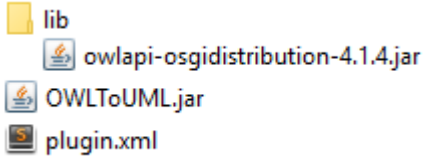
Kriterijus	Java	C#.NET
Greitaveika	+-	+
Efektyvus taikomųjų programų kūrimas	-	+
Sintaksės paprastumas	+	+
Palaikymas, ištekliai	+	-
Didelis API bibliotekų kiekis	+	-
Stabilumas, saugumas	+	+
Komfortabilūs IDE redaktoriai	+	+

UML modeliavimo įrankį „MagicDraw“ sudaro pagrindiniai moduliai, įskiepiai kurie skirti darbui su *UML*.

Integruoto kūrimo aplinkai (angl. - *Integrated Development Environment*) buvo pasirinktas Eclipse Mars [3] įrankis, kadangi šiai aplinkai jau yra aprašyta „MagicDraw“ dokumentacija (*Open API*)

4.2. Transformavimo įrankio realizacijos aprašymas

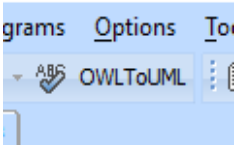
Pasirinkta realizacijos platforma leidžia kiekvienam vartotojui nesudėtingai integruoti *OWL 2* transformavimo į *UML* įskiepį į vieną iš populiariausių „MagicDraw“ modeliavimo sistemų. Įskiepio *OWLToUML* failų katalogas: turi būti įkeliami į „„MagicDraw“ instaliacijos katalogas/plugins“ aplanką. Katalogo turinys pavaizduotas 33 pav. *OWLToUML.jar* sukompiliuotas programinis kodas, *plugin.xml* aprašomi bendriniai parametrai ir nurodomos pagalbinės bibliotekos, vidiniame kataloge *lib* laikomos papildomos reikalingos bibliotekos šiuo atveju: *owlapi-osgidistribution-4.1.4.jar*



33 pav. Įskiepio failai

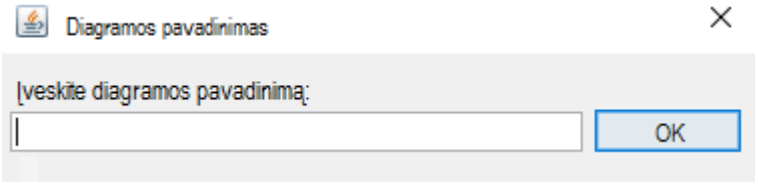
„MagicDraw“ modeliavimo aplinkoje įrankių juostoje yra mygtukas „OWLToUML“. Pasirinkus šį įrankį atidaromas bylos pasirinkimo langas (7 pav.) kuriame reikia pasirinkti transformuojamos *OWL 2* ontologijos bylą. Pasirinkus ontologijos byla, įskiepis atveria iššokantį langa (35 pav.) kuriame reikia įvesti paketo (angl. - *package*) pavadinimą. Įvedus paketo pavadinimą įskiepis siūlo (36 pav.)

įvesti diagramos pavadinimą. Neįvedus paketo pavadinimo sugeneruojamas pagal ontologijos pavadinimą, taip pat kaip ir diagramos.

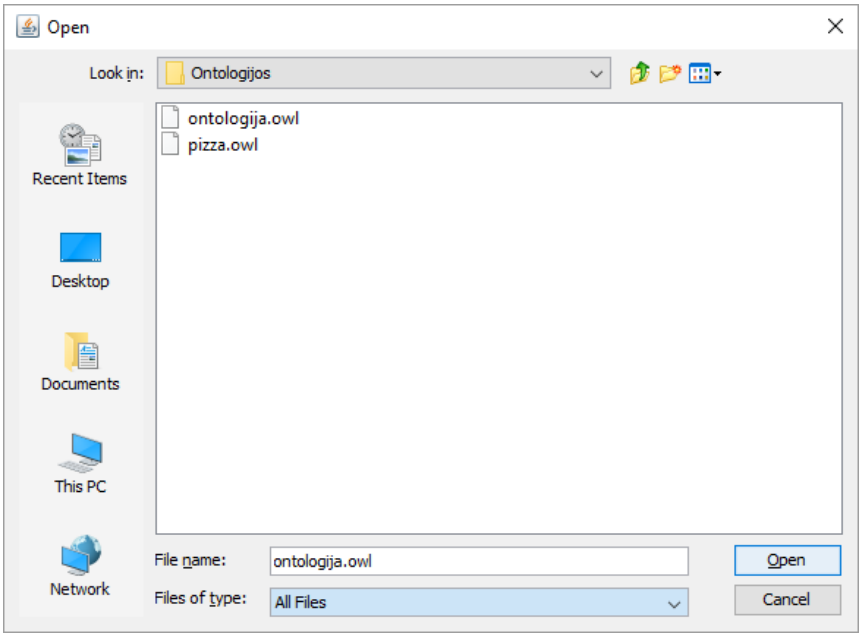


34 pav. Įskiepio mygtukas „OWLToUML“ įrankių juostoje „MagicDraw“ aplinkoje

35 pav. Paketo pavadinimo įvedimas

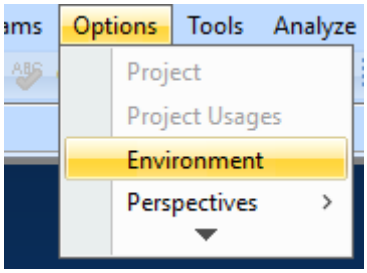


36 pav. Diagramos pavadinimo įvedimas

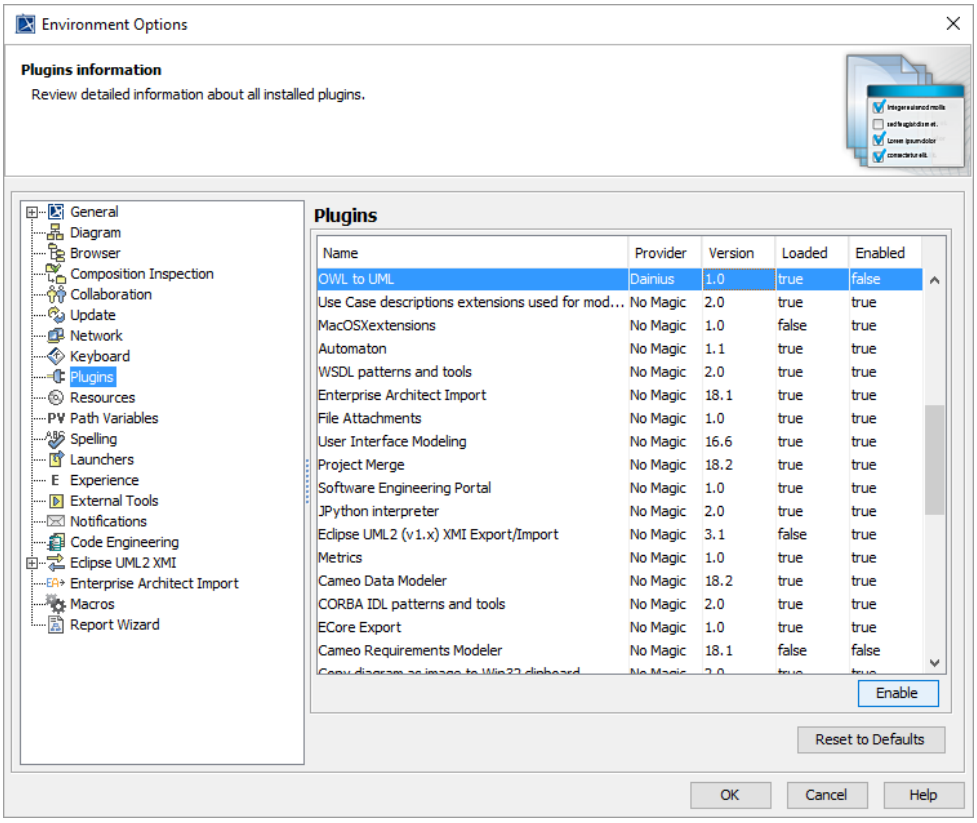


37 pav. Ontologijos bylos parinkimo sąsaja

Įkėlus įskiepį į anksčiau nurodytą aplanką ir paleidus „MagicDraw“ modeliavimo aplinka pagal numatytuosius nustatymus šis yra aktyvuojamas automatiškai ir įrankių juostoje atsiranda mygtukas „OWLToUML“. Jei įskiepio mygtukas modeliavimo aplinkos įrankių juostoje neatsiranda, tuomet galima įskiepį įjungti rankiniu būdu: spaudžiame menių juostoje pasirinkimą *Options->Environment* (38 pav.) atsivėrusioje kortelėje kairėje pusėje pasirenkame *Plugins* ir iš atsiradusio sąrašo pasirenkame *OWL to UML* įskiepį, paspaudus žemiau esantį mygtuką *Enable* įskiepio mygtukas „OWLToUML“ yra pridedamas į įrankių juostą (34 pav.)

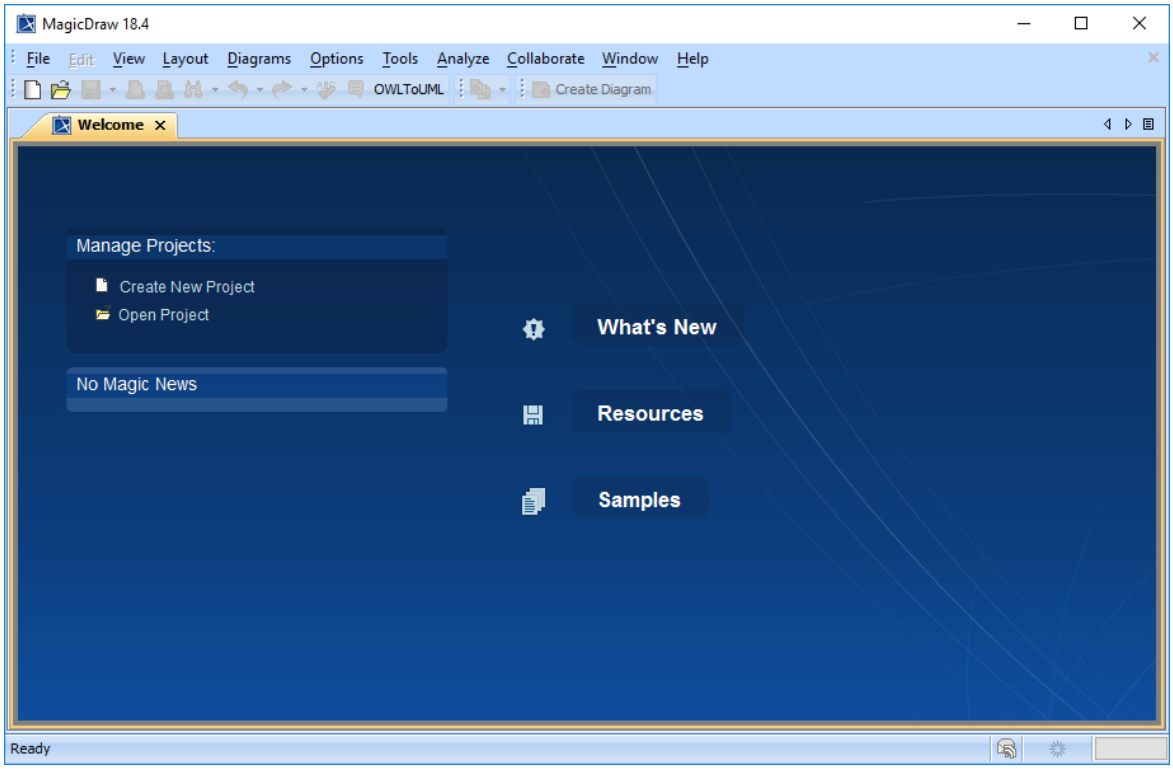


38 pav. „MagicDraw“ modeliavimo aplinkos nustatymų pasirinkimas

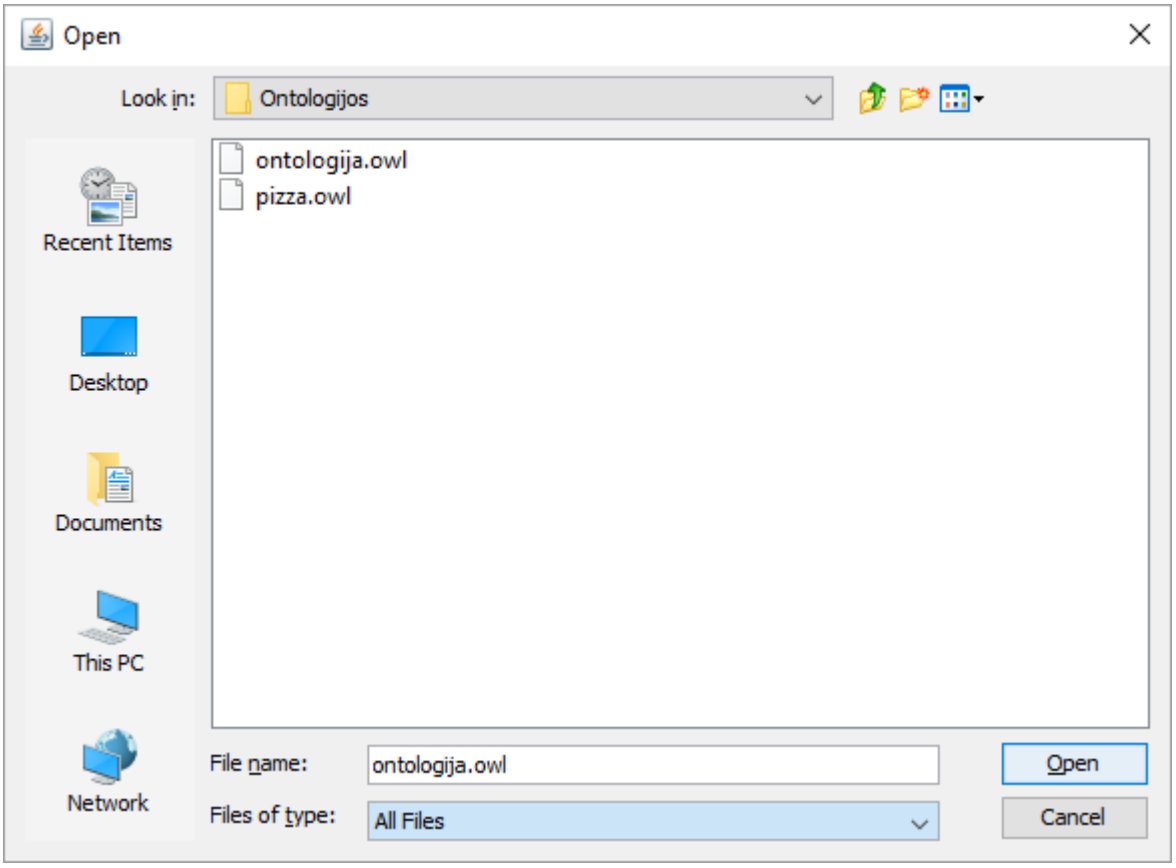


39 pav. Rankinis įskiepio aktyvavimas

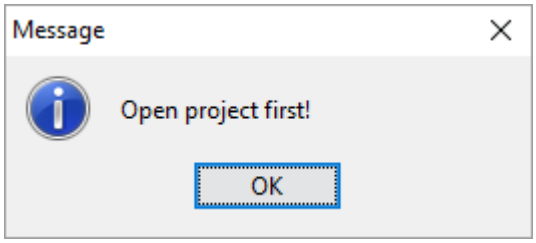
Pagrindiniame „MagicDraw“ modeliavimo aplinkos lange (40 pav.) įrankių juostoje pasirinkus mygtuką *OWLToUML* atidaromas ontologijos aprašo failo pasirinkimo langas. Jei nėra atidaryto projekto įskiepis praneša (42 pav.), kad pirmiau turi būti atidarytas aktyvus projektas į kuria bus kuriamas paketas. Pasirinkus ontologijos aprašo failą sistema informuoja (43 pav.), kad galima įvesti paketo pavadinimą, sekančiu atveju pavadinimas bus nustatomas pagal ontologijos failo pavadinimą. Sekančiame žingsnyje įskiepis siūlo įvesti diagramos pavadinimą (44 pav.), jei pavadinimas neįvedamas ši diagrama pavadinama – *Untitled*.



40 pav. „MagicDraw“ aplinka su įskiepio funkcionalumu

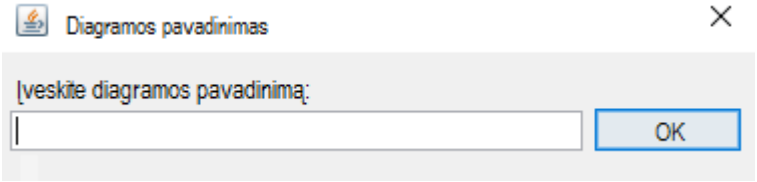


41 pav. Ontologijos pasirinkimo langas



42 pav. Informacinis pranešimas

43 pav. Paketo pavadinimo įvedimo langas



44 pav. Diagramos pavadinimo įvedimo langas

4.3. Testavimo modelis, duomenys, rezultatai

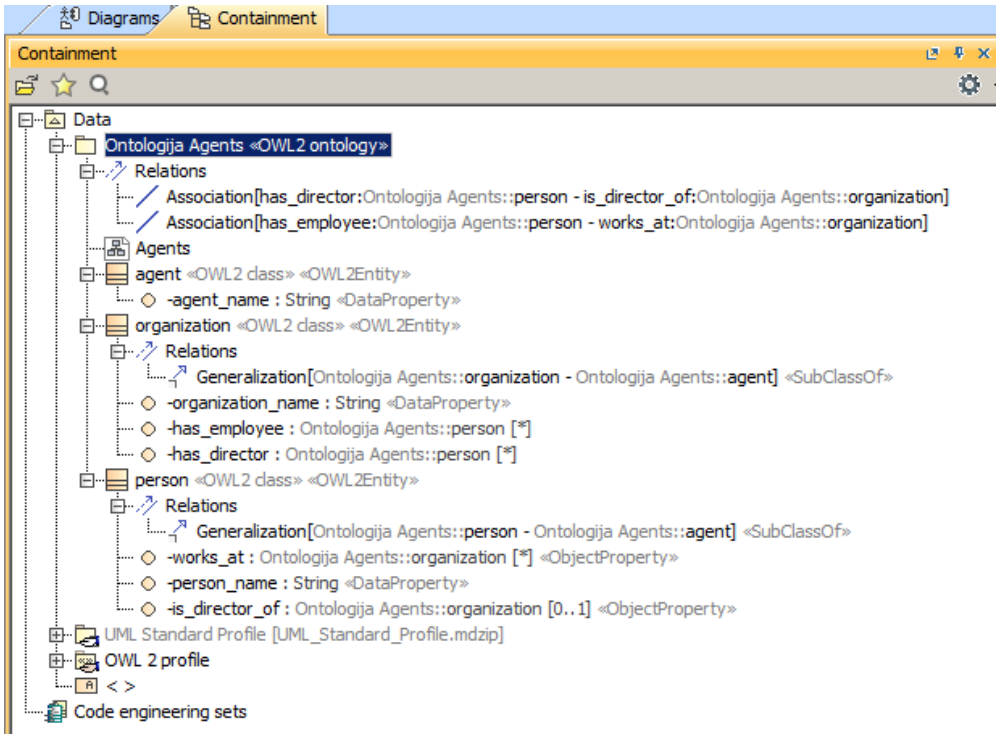
Testavimui buvo sudaryta *Agentai* ontologija (45 pav.). Ontologiją sudaro pagrindiniai minimalaus *OWL 2* profilio elementai.

Po įvykdytos transformacijos „MagicDraw“ aplinkoje gauta objektų hierarchija pavaizduota 46 pav. Transformacijos metu sudarytas įskiepio diagrama pavaizduota 47 pav.

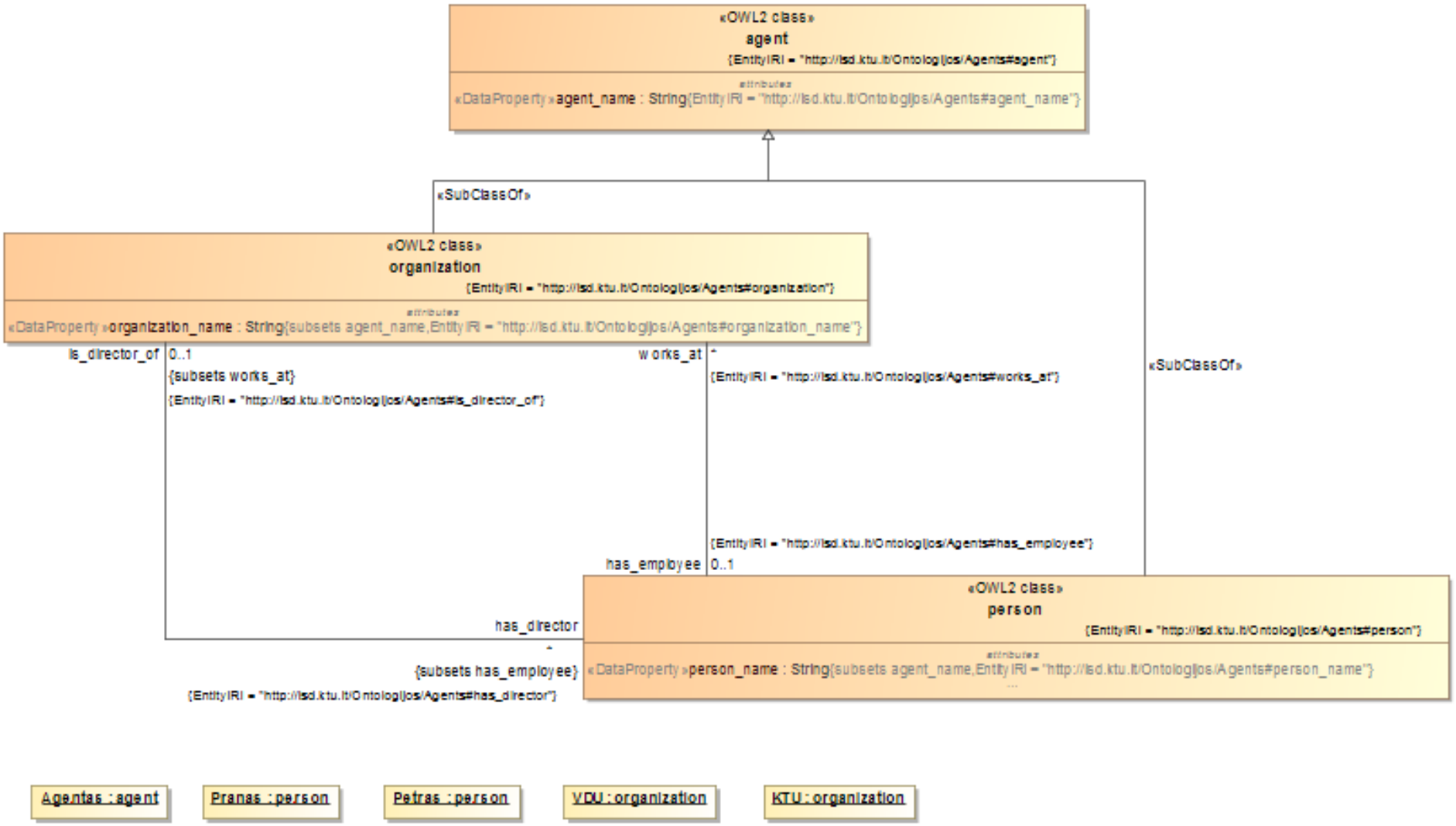
Pagal gautus rezultatus darome išvada, kad transformacija įvykdyta teisingai ir be duomenų praradimo.

```
Prefix(:=<http://isd.ktu.lt/Ontologijos/Agents#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)

Ontology(<http://isd.ktu.lt/Ontologijos/Agents>
Declaration(Class(:agent))
Declaration(Class(:organization))
Declaration(Class(:person))
Declaration(ObjectProperty(:has_director))
Declaration(ObjectProperty(:has_employee))
Declaration(ObjectProperty(:is_director_of))
Declaration(ObjectProperty(:works_at))
Declaration(DataProperty(:agent_name))
Declaration(DataProperty(:organization_name))
Declaration(DataProperty(:person_name))
Declaration(NamedIndividual(:Agentas))
Declaration(NamedIndividual(:KTU))
Declaration(NamedIndividual(:Petras))
Declaration(NamedIndividual(:Pranas))
Declaration(NamedIndividual(:VDU))
Declaration(Datatype(xsd:String))

# Object Property: :has_director (:has_director)
SubObjectPropertyOf(:has_director :has_employee)
InverseObjectProperties(:has_director :is_director_of)
FunctionalObjectProperty(:has_director)
ObjectPropertyDomain(:has_director :organization)
ObjectPropertyRange(:has_director :person)

# Object Property: :has_employee (:has_employee)
InverseObjectProperties(:has_employee :works_at)
FunctionalObjectProperty(:has_employee)
ObjectPropertyDomain(:has_employee :organization)
ObjectPropertyRange(:has_employee :person)

# Object Property: :is_director_of (:is_director_of)
SubObjectPropertyOf(:is_director_of :works_at)
FunctionalObjectProperty(:is_director_of)
ObjectPropertyDomain(:is_director_of :person)
ObjectPropertyRange(:is_director_of :organization)

# Object Property: :works_at (:works_at)
FunctionalObjectProperty(:works_at)
ObjectPropertyDomain(:works_at :person)
ObjectPropertyRange(:works_at :organization)

# Data Property: :agent_name (:agent_name)
DataPropertyDomain(:agent_name :agent)
DataPropertyRange(:agent_name xsd:String)

# Data Property: :organization_name (:organization_name)
SubDataPropertyOf(:organization_name :agent_name)
DataPropertyDomain(:organization_name :organization)
DataPropertyRange(:organization_name xsd:String)

# Data Property: :person_name (:person_name)
SubDataPropertyOf(:person_name :agent_name)
DataPropertyDomain(:person_name :person)
DataPropertyRange(:person_name xsd:String)

# Class: :agent (:agent)
DisjointUnion(:agent :organization :person)
# Class: :organization (:organization)
SubClassOf(:organization :agent)
# Class: :person (:person)
SubClassOf(:person :agent)

# Individual: :Agentas (:Agentas)
ClassAssertion(:agent :Agentas)
# Individual: :KTU (:KTU)
ClassAssertion(:organization :KTU)
# Individual: :Petras (:Petras)
ClassAssertion(:person :Petras)
# Individual: :Pranas (:Pranas)

ClassAssertion(:person :Pranas)
# Individual: :VDU (:VDU)
ClassAssertion(:organization :VDU)
)
```

45 pav. Agentai ontologija



46 pav. „MagicDraw“ transformuotos ontologijos komponentų hierarchija



47 pav. Gauti rezultatai *UML* diagramoje

5. EKSPERIMENTINIS ONTOLOGIJOS TRANSFORMACIJOS ĮSKIEPIO TYRIMAS

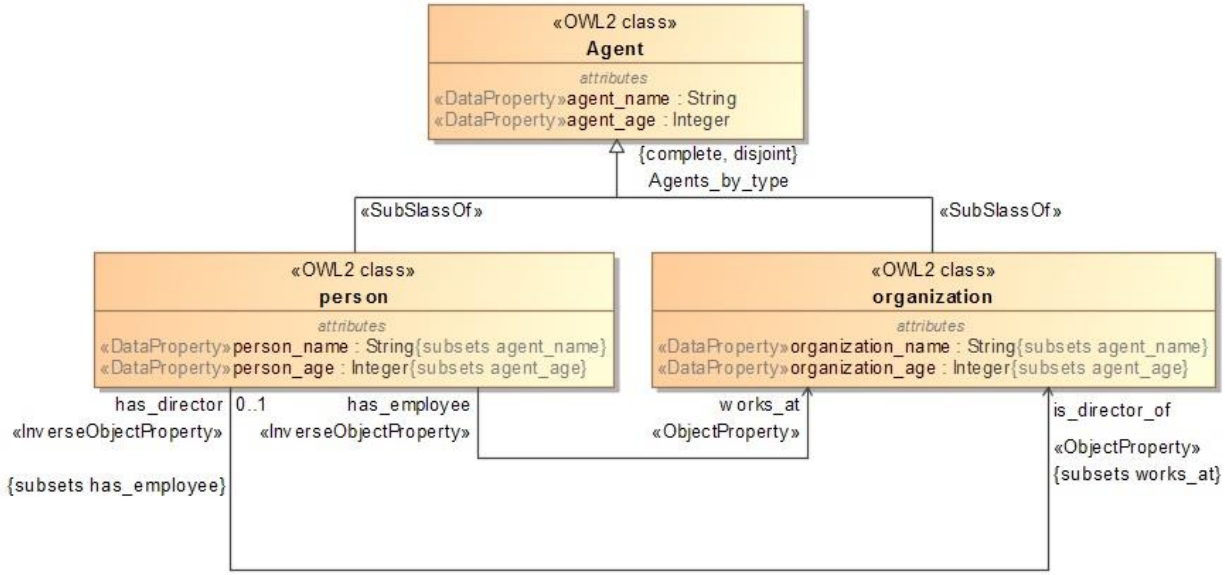
5.1. Eksperimento planas

Eksperimento tikslas yra nustatyti ar „MagicDraw“ transformavimo *OWL* į *UML* algoritmą realizuojantis įskiepis geba teisingai ir greitai atlikti transformaciją su sukurta testuojama *Agentai OWL 2* ontologija (48 pav.) kuri padengia pagrindinius *OWL 2 lite* elementus kurie yra pamatas *OWL 2*. Šiam tikslui nustatyti sudarytas planas:

1. Palyginti gaunamus rezultatus analizuotuose „OWLGrEd“ ir „Protégé“ įrankiuose įskaitant taip pat ir sukurtą prototipą.
2. Palyginti diagramos generavimo laiką.
3. Palyginti funkcionalumo galimybes.

```
Prefix(:=<http://isd.ktu.lt/Ontologijos/Agents#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)

Ontology(<http://isd.ktu.lt/Ontologijos/Agents>
Declaration(Class(:agent))
Declaration(Class(:organization))
Declaration(Class(:person))
Declaration(ObjectProperty(:has_director))
Declaration(ObjectProperty(:has_employee))
Declaration(ObjectProperty(:is_director_of))
Declaration(ObjectProperty(:works_at))
Declaration(DataProperty(:agent_name))
Declaration(DataProperty(:organization_name))
Declaration(DataProperty(:person_name))
Declaration(NamedIndividual(:Agentas))
Declaration(NamedIndividual(:KTU))
Declaration(NamedIndividual(:Petras))
Declaration(NamedIndividual(:Pranas))
Declaration(NamedIndividual(:VDU))
Declaration(Datatype(xsd:String))

# Object Property: :has_director (:has_director)
SubObjectPropertyOf(:has_director :has_employee)
InverseObjectProperties(:has_director :is_director_of)
FunctionalObjectProperty(:has_director)
ObjectPropertyDomain(:has_director :organization)
ObjectPropertyRange(:has_director :person)

# Object Property: :has_employee (:has_employee)
InverseObjectProperties(:has_employee :works_at)
FunctionalObjectProperty(:has_employee)
ObjectPropertyDomain(:has_employee :organization)
ObjectPropertyRange(:has_employee :person)

# Object Property: :is_director_of (:is_director_of)
SubObjectPropertyOf(:is_director_of :works_at)
FunctionalObjectProperty(:is_director_of)
ObjectPropertyDomain(:is_director_of :person)
ObjectPropertyRange(:is_director_of :organization)

# Object Property: :works_at (:works_at)
FunctionalObjectProperty(:works_at)
ObjectPropertyDomain(:works_at :person)
ObjectPropertyRange(:works_at :organization)

# Data Property: :agent_name (:agent_name)
DataPropertyDomain(:agent_name :agent)
DataPropertyRange(:agent_name xsd:String)

# Data Property: :organization_name (:organization_name)
SubDataPropertyOf(:organization_name :agent_name)
DataPropertyDomain(:organization_name :organization)
DataPropertyRange(:organization_name xsd:String)

# Data Property: :person_name (:person_name)
SubDataPropertyOf(:person_name :agent_name)
DataPropertyDomain(:person_name :person)
DataPropertyRange(:person_name xsd:String)

# Class: :agent (:agent)
DisjointUnion(:agent :organization :person)
# Class: :organization (:organization)
SubClassOf(:organization :agent)
# Class: :person (:person)
SubClassOf(:person :agent)

# Individual: :Agentas (:Agentas)
ClassAssertion(:agent :Agentas)
# Individual: :KTU (:KTU)
ClassAssertion(:organization :KTU)
# Individual: :Petras (:Petras)
ClassAssertion(:person :Petras)
# Individual: :Pranas (:Pranas)

ClassAssertion(:person :Pranas)
# Individual: :VDU (:VDU)
ClassAssertion(:organization :VDU)
)
```

48 pav. Agentai ontologijos aprašas



49 pav. *UML* reprezentatyvus pavyzdys

Kiekvienos skirtingos aplinkos eksperimento metu naudojantis chronometru fiksuota apytikslė laiko trukmė kuria užtruko įrankis atlikdamas užduoti nuo ontologijos įkėlimo iki ontologijos atvaizdavimo vartotojo sąsajoje. Žemiau esančiame skyriuje aprašoma apie kiekvieną atvejį atskirai.

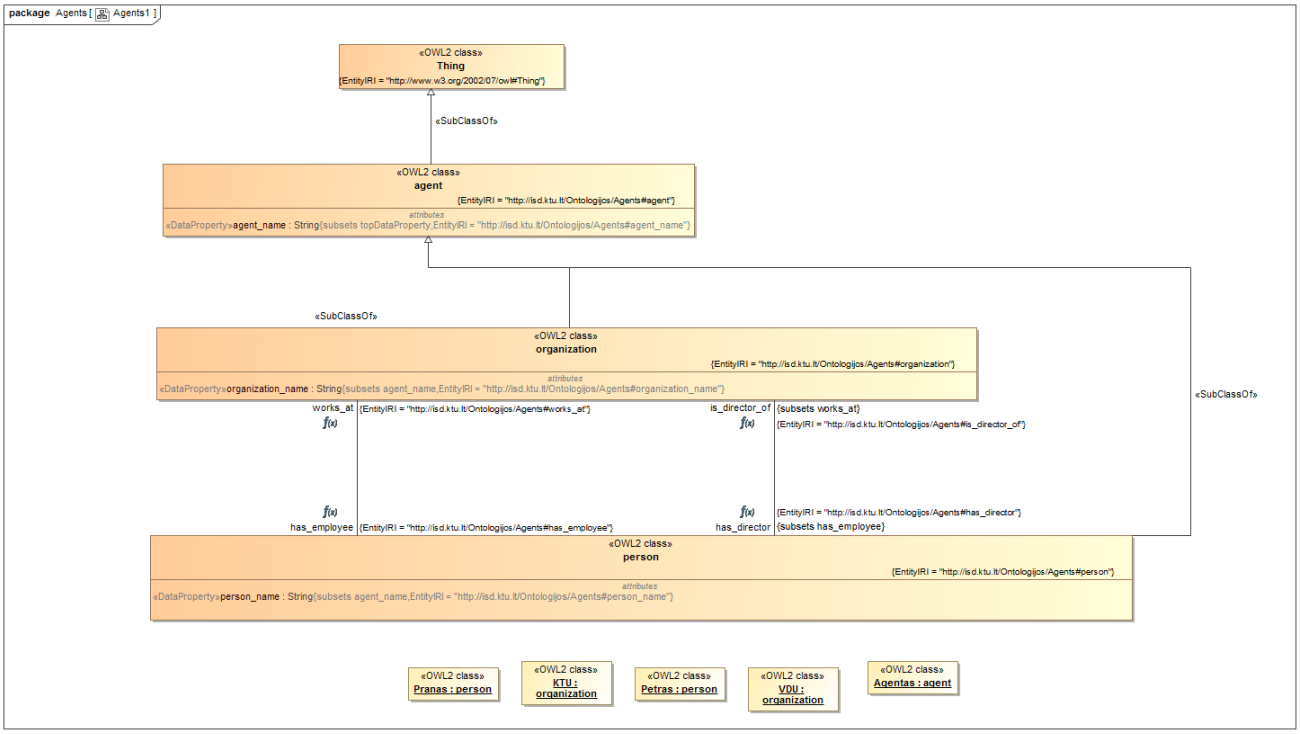
5.2. Eksperimentas su „MagicDraw“

Eksperimento metu su „MagicDraw“ modeliavimo aplinkoje darbo metu sukurto įskiepio pagalba buvo transformuota *OWL 2 lite* elementus padengianti ontologija į *UML* diagramą (50 pav.). Transformacijos metu buvo perkelti ontologijos objektai į *UML* (51 pav.)

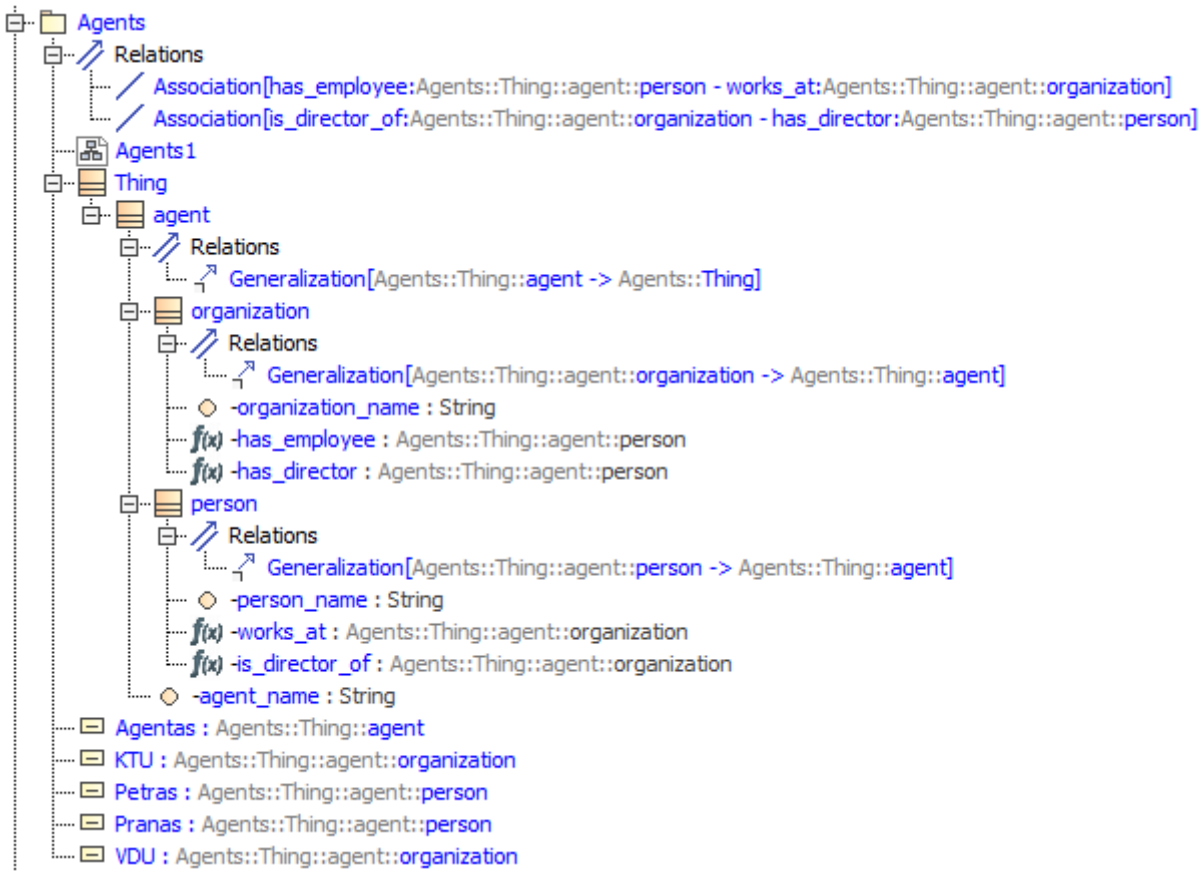
Kaip matoma iš žemiau pateiktos lentelės (Lentelė 13) ontologijos elementai perkelti į *UML* sklandžiai – transformacija suveikė kaip tikėtasi. Ontologijos vidutine transformavimo trukmė nuo ontologijos pasirinkimo iki transformuotų ir įrankio automatiškai atvaizduotų *UML* elementų diagramoje „MagicDraw“ aplinkoje apytiksliai 3 sekundės. Visi *OWL 2* lite elementai buvo perkelti tvarkingai į *UML*.

Lentelė 13. Transformacijos rezultatai

	Prieš transformaciją	Po transformacijos
Klasės	3	3
Poklasės	2	2
Nepersikertančių elementų sąjungos	1	1
Objektų savybės	4	4
Objektų savybių domenai	4	4
Objektų savybių sritys	4	4
Atvirkštinės objektų savybės	4	4
Specializuojančios objektų savybės	2	2
Funkcinės objektų savybės	1	1
Duomenų savybės	3	3
Duomenų savybių domenai	3	3
Duomenų savybių sritys	3	3
Specializuojančios duomenų savybės	1	1
Individai	3	3
Duomenų tipai	2	2



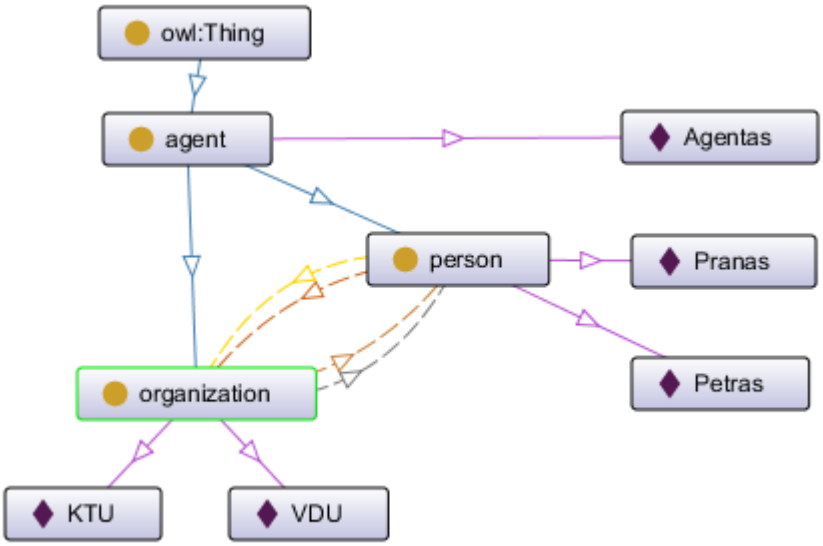
50 pav. „MagicDraw“ modeliavimo aplinkoje sugeneruota ontologijos diagrama



51 pav. „MagicDraw“ modeliavimo aplinkoje sugeneruotos ontologijos diagramos elementai

5.3. Eksperimentas su „Protégé“

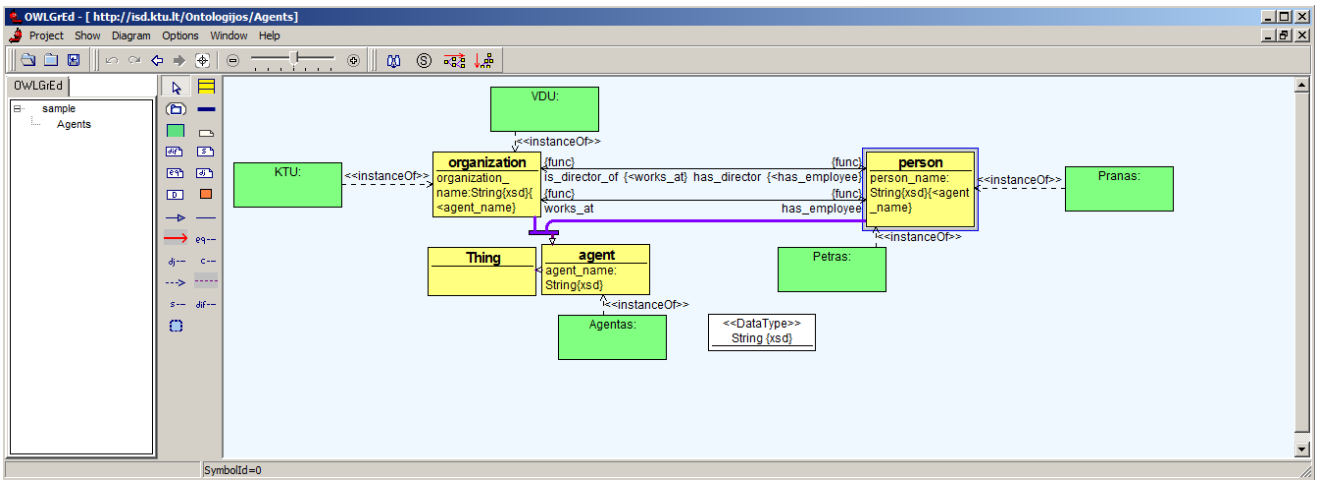
Eksperimentui su „Protégé“ panaudota ta pati ontologija *Agents*. Eksperimentas pakartotas pagal tą patį scenarijų kaip ir su „MagicDraw“ – paleidžiamas įrankis, įrankiui užsikrovus pasirenkama ontologija ir pradedamas skaičiuoti laikas per kurį įrankis ją apdoros. Ontologijos įkėlimo į įrankį trukmė apytiksliai 7s. Automatiškai sugeneruotas vaizdas pateikiamas 52 pav.



52 pav. „Protégé“ *OntoGraf* generuojama diagrama

5.4. Eksperimentas su „OWLGrEd“

Eksperimentui su „OWLGrEd“ panaudota ta pati ontologija *Agents.* Eksperimentas pakartotas pagal tą patį scenarijų kaip ir su prieš tai dviem įrankiais „Protégé“ ir „MagicDraw“ – paleidžiamas įrankis, įrankiui užsikrovus pasirenkama ontologija ir pradedamas skaičiuoti laikas per kurį įrankis ją apdoros. Ontologijos įkėlimo į įrankį trukmė nuo ontologijos pasirinkimo iki ontologijos atvaizdavimo apytiksliai yra 14s . Automatiškai „OWLGrEd“ įrankio sugeneruotas vaizdas pateikiamas 53 pav.



53 pav. „OWLGrEd“ sugeneruotos ontologijos diagrama

5.5. Eksperimento rezultatai

Gautų rezultatų analizė parodė, kad transformacijos įskiepio prototipo *OWL* į *UML* diagramą rezultatai yra teisingi. Eksperimento metu nustatytas diagramos atvaizdavimo laikas skirtingose aplinkose:

1. „MagicDraw“ – ~3s.
2. „Protégé“- ~7s.
3. „OWLGrEd“ - ~14s.

Lentelė 14. Įrankių palyginimo rezultatai

Palyginimo kriterijus	**„Protégé“**	**„OWLGrEd“**	**Darbo metu sukurtas įskiepis „MagicDraw“ aplinkoje**
Grafinis redagavimas	-	+	+
Grafinis kūrimas	-	+	+
Grafiniai elementai objektų savybėms	Du ryšiai	Vienas ryšys	Vienas ryšys
UML klasių diagramų palaikymas	-	+-	+
Grafinio projekto išsaugojimas	-	+	+
Modeliais grindžiamas sistemų projektavimas	-	-	+
OWL 2 ontologijos vizualizavimas *UML*	-	+	+
Transformavimo trukmė, s	7	14	3

Pagal gautus rezultatus sukurtas prototipas ontologiją transformuoja greičiau už „OWLGrEd“ ir „Protégé“ įrankius, taip pat turi platesnes galimybes toliau plėtojant projektą *UML*.

Taip pat galimybė transformuoti į *UML* yra tik sukurtam įskiepio prototipe, kituose įrankiuose yra tik galimybė atvaizduoti vizualiai.

Įvertinus prototipo ir kitų įrankių galimybes, galima daryti išvadą, kad sukurtas prototipas pateisina informacinių sistemų kūrėjų lūkesčius norint perkelti ontologijos kalba aprašytą failą į *UML*. Taip sutaupomas laikas, perkeliant ontologiją į *UML*, išvengiama žmogiškųjų klaidų tikimybė. Įvertinant diagramos sudarymo laiką esant tokiems pat kriterijams, „MagicDraw“ šiame darbe sukurto prototipo diagramos laikas buvo mažiausias.

6. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Atlikta esamų transformacijos įrankių analizė parodė, kad esami transformacijos įrankiai turi trūkumų ir kad nėra patogaus metodo *OWL 2* į *ULM* diagramai transformuoti.
2. Analizės metu buvo išnagrinėti jau egzistuojantys ontologijos atvaizdavimo įrankiai: „Protégé“, „OWLGrEd“. Analizė parodė, jog nėra ontologijos transformavimo galimybės į *UML* diagramą.
3. Sudaryta įskiepio projektinė dalis leido realizuoti siekiamo „MagicDraw“ įskiepio prototipą, padedantį lanksčiau transformuoti ontologiją į *UML* diagramą.
4. Atliktas eksperimentas, kurio metu pagal sudarytą metodika buvo palygintos transformacijos. Eksperimentas parodė, kad įskiepis suteikia galimybę įtraukti ontologijų kūrimą į informacinių sistemų projektavimą.

7. LITERATŪRA

[1] *OWL* Web Ontology Language [Tikle]. Available: https://www.w3.org/TR/2004/REC-owl-features-20040210/ [Kreiptasi 18 09 2015].
[2] *UML* business process, architecture, software and system modeling tool [Tinkle]. Available: www.nomagic.com/products/magicdraw.html [Kreiptasi 01 06 2015].
[3] https://www.eclipse.org/mars/
[4] Protégé tool [Tinkle]. Available: http://protege.stanford.edu/products.php [Kreiptasi 10 11 2015].
[5] OWLGrEd [Tinkle]. Available: http://owlgred.lumii.lv/ [Kreiptasi 09 09 2015]
[6] *MagicDraw* Open API [Tinkle]. Available: https://www.nomagic.com/files/manuals/MagicDraw%20OpenAPI%20UserGuide.pdf [Kreiptasi 05 01 2015].
[7] The *OWL* API [Tinkle]. Available: http://owlapi.sourceforge.net/ [Kreiptasi 10 11 2015].
[8] *OWLAPI* Distribution 4.2.3. API [Tinkle]. Available: http://owlcs.github.io/owlapi/apidocs_4_2_3/index.html [Kreiptasi 10 11 2015].
[9] Unified Modeling Language [Tinkle]. Available: https://lt.wikipedia.org/wiki/Unified_Modeling_Language [Kreiptasi 18 09 2015].
[10] Unified Modeling Language [Tinkle]. Available: http://www.uml.org/ [Kreiptasi 18 09 2015].
[11] Comparison of Java and C++ [Tinkle]. Available: https://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B [Kreiptasi 10 02 2016]
[12] ODM Implementation [Tinkle]. Available: http://www.eclipse.org/atl/usecases/ODMImplementation/
[13] Ontology Definition Metamodel [Tinkle]. Available: http://www.omg.org/spec/ODM/
[14] UML Concrete Syntax [Tinkle]. Available: https://www.w3.org/2007/OWL/wiki/UML_Concrete_Syntax
[15] PowerDesigner [Tinkle]. Available: http://go.sap.com/product/data-mgmt/powerdesigner-data-modeling-tools.html
[16] Ontologija(informatika) [Tinkle]. Available: https://lt.wikipedia.org/wiki/Ontologija_(informatika)
[17] Semantinių technologijų stekas [Tinkle]. Available: http://gerasblogas.lt/programu-inzinerija/ontologijos-ir-ju-naudojimas-duomenu-inzinerijoje/
[18] RDF [Tinkle]. Available: https://www.w3.org/RDF/
[19] Semantinio tinklo technologijos [Tinkle]. Available https://lt.wikipedia.org/wiki/Semantinis_tinklas