KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Ričardas Baltulis

UML PROFILIU PAVAIZDUOTŲ ONTOLOGIJŲ TRANSFORMAVIMO Į OWL 2 KALBĄ PROTOTIPAS

Baigiamasis magistro projektas

Vadovas
doc. dr. Rita Butkienė

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

UML PROFILIU PAVAIZDUOTŲ ONTOLOGIJŲ TRANSFORMAVIMO Į OWL 2 KALBĄ PROTOTIPAS

Baigiamasis magistro projektas

Informacinių sistemų inžinerijos studijų programa (kodas 621E15001)

Vadovas

doc. dr. Rita Butkienė

2016-05-24

Konsultantas

prof. dr. L. Nemuraitė

2016-05-24

Recenzentas

lekt. mag. A. Šukys

2016-05-24

Projektą atliko

Ričardas Baltulis

2016-05-24

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

(Fakultetas)

Ričardas Baltulis

(Studento vardas, pavardė)

Informacinių sistemų inžinerijos studijų programa, 621E15001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto
„UML profiliu pavaizduotų ontologijų transformavimo į OWL 2 kalbą prototipas“
AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. Gegužės 24 d.
Kaunas

Patvirtinu, kad mano, **Ričardo Baltulio**, baigiamasis projektas tema „UML PROFILIU PAVAIZDUOTŲ ONTOLOGIJŲ TRANSFORMAVIMO Į OWL 2 KALBĄ 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)*

Ričardas, Baltulis. *UML* PROFILIU PAVAIZDUOTŲ ONTOLOGIJŲ TRANSFORMAVIMO Į *OWL 2* KALBĄ PROTOTIPAS. Magistro baigiamasis projektas / vadovas doc. dr. Rita Butkienė; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Informatikos inžinerija, technologijos mokslai

Reikšminiai žodžiai: *ontologija, transformacija, OWL, OWL profilis, UML, MagicDraw.*

Kaunas, 2016. 51 p.

SANTRAUKA

Egzistuoja daug informacinių sistemų kurios integruojamos, arba teikia paslaugas naudojant ontologijas. „MagicDraw“ yra sistemų inžinerijai skirtas įrankis, dirbantis UML pagrindu, palaikantis modeliais grįstą kūrimą. Kadangi ontologijos yra tų sistemų dalimi tai reikia turėti vieningą būdą kaip jas ten įkomponuoti.

Norint sėkmingai panaudoti ontologijas informacinių sistemų kūrime yra svarbu turėti įrankius leidžiančius projektuojant sistemą projektuoti ir pačias ontologijas tais pačiais įrankiais. Ontologijų inžinieriai daug sėkmingiau suderintų vienodą požiūri į dalykinę sritį jeigu naudotų grafines priemones ontologijoms vizualizuoti. Ontologijos paskirtis yra leisti skirtingiems atstovams į tą patį dalyką žiūrėti vienodai ir jį suprasti vienodai. Ontologijų perkėlimas į UML leistu sukurtus modelius transformuoti ir į kitas formas, taip pat susieti ontologiją su kuriamos sistemos reikalavimais ar sistemos architektūra.

Išanalizavus ontologijų kalbą OWL 2 bei ontologijų įrankius, nustatyta, kad šiam uždaviniui geriausiai tiktų „MagicDraw“ įrankis, todėl buvo sukurtas OWL profilis ir „MagicDraw“ įskiepis leidžiantis atlikti transformacijas iš UML pagrindu pavaizduotų ontologijų į OWL 2 ontologijų kalbą. Įskiepis išbandytas su keliais pavyzdžiais ir gauti rezultatai palyginti keliuose skirtinguose ontologijų vaizdavimo įrankiuose.

Realizuotas sprendimas suteikia galimybę transformuoti sumodeliuotą ontologiją į ontologijų kalbą OWL 2 ir toliau ją plėtoti, taip pat ontologiją patikrinti ontologijų kūrimo įrankiais galinčiais atlikti loginius išvedimus.

Ričardas, Baltulis. PROTOTYPE FOR TRANSFORMATION OF ONTOLOGY REPRESENTED USING UML PROFILE TO ONTOLOGY IN OWL 2 LANGUAGE: Master‘s thesis in Information Systems Engineering / supervisor assoc. doc. dr. Rita Butkienė. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Informatics Engineering, Technology Science

Key words: *ontology, transformation, OWL, OWL profile, UML, MagicDraw*.

Kaunas, 2016. 51 p.

SUMMARY

There are a lot of information systems that integrate or providing services using ontologies. “MagicDraw” is a system engineering tool intended for working with UML, model-based support for the development. Since ontologies are those parts of the system need to have a unified way as they to be incorporated there.

For a successful use of ontologies in information systems development is important to have the tools to enable design of the system design and the same ontologies of the same tools. Many of ontologies engineers successfully combined same attitude to the subject area if used graphical tools to visualize ontologies. Ontology intended to allow different agents to see the same things and understand it in the same ways. Ontologies allow the transfer to the UML models developed to transform into other forms, as well as to link the ontology to create the system requirements and system architecture.

After analyzing the ontology language OWL 2 and ontologies tools, found that this task will work best for “MagicDraw” tool, so it was created OWL profile and “MagicDraw” plugin allows to perform transformations from ontology represented in the UML to OWL 2 ontology language. Plugin tested with several examples and the results were compared in several different ontologies visualizations.

Realized solution provides the ability to transform the modeled ontology to ontologies language OWL 2, and develop it, as well as check ontology ontologies development tools capable of performing a logical output.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Web 2.0 - interaktyvus internetas.
Web 3.0 - semantinis internetas
API - aplikacijų programavimo sąsaja.
XML - duomenų struktūrų bei jų turinio aprašomoji kalba (angl. *eXtensible Markup Language*).
HTML (Hypertext Markup Language) - tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete. Kalbą standartizuoja W3C 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.
Ontologija - oficialus žodynas tinkamų sąvokų, ypatybių, kuriuos sieja tam tikros taisyklės.
OWL - Ontologijos Tinklo Kalba. Ontologijos aprašymo kalba atitinkanti W3C Semantikos Tinklo standartams.
Plug-in, įskiepis - sukompiliuotas programinis kodas, kuris padidina programos galimybes.
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.
URI - Universalus išteklių identifikatorius. Tai yra formatas, naudojamas semantiniame tinkle, kad paskirtų identifikuotus išteklius.
Žiniatinklis - pasaulinis tinklas (angl. *World Wide Web 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 objektinių programų dokumentus.

ĮVADAS

Šis darbas - „*UML* profiliu pavaizduotų ontologijų transformavimo į *OWL 2* kalbą prototipas“ - priklauso informaciniu sistemų inžinerijos studijų programai. Darbo metu realizacijos klausimais taip pat buvo konsultuotasi su „NoMagic“ įmonės atstovu Justinu Bisikirsku.

Darbo problematika ir aktualumas

Šiuo metu kuriama daug projektų, kuriuose naudojamos ontologijos, tačiau projektavimo procesuose naudojamuose *CASE* įrankiuose ontologijų redaktoriais sukurtas ontologijas reikia rankiniu būdu pavaizduoti *UML* modeliais, todėl gaištamas papildomas laikas ir išauga tikimybė, kad projekto modeliai nesutaps su tikrosiomis ontologijomis. Šią problemą galima išspręsti sukuriant *UML* profilį *OWL 2* ontologijoms vaizduoti ir *CASE* įrankio įskiepį, kuris leistų transformuoti *UML* profiliu pavaizduotas ontologijas į ontologijų kalbą *OWL 2*.

Jeigu sistemų kūrimas yra modeliais grindžiamas ir yra naudojamos *UML* diagramos kaip priemone projektuoti sistemas, tada atsiranda poreikis turėti priemones kurios leistu į informacines sistemas įtraukti ontologijas. Jų naudojimas padėtų realizuoti tam tikras funkcijas. Pavyzdžiui semantinė paieška. Šiai funkcijai būtų reikalinga ontologija kuri turėtų būti suprojektuota toje sistemoje.

Darbo tikslas ir uždaviniai

Šio darbo tikslas yra sudaryti galimybę įtraukti ontologijų kūrimą į informacinių sistemų projektavimą, sukuriant *UML* profiliu pavaizduotų *OWL 2* ontologijų transformavimo į *OWL 2* kalbą taisykles bei algoritmus ir juos realizuojančio įrankio prototipą.

Šiam tikslui pasiekti iškelti uždaviniai:

1. Išanalizuoti:
   1.1. Ontologijų kalbą *OWL 2*, 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. Sukurti *UML OWL 2* profilį, sudaryti transformavimo taisykles ir algoritmus, suprojektuoti transformavimo įrankį;
3. Realizuoti įrankio prototipą;
4. Atlikti eksperimentą, kuris leistų įvertinti sprendimo tinkamumą ir efektyvumą;
5. Apibendrinti tyrimo rezultatus.

Darbo rezultatai ir jų svarba

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

1. PROBLEMINĖS SRITIES ANALIZĖ

1.1. Analizės tikslas

Pagrindinis analizės tikslas yra išsiaiškinti, kurie sprendimai bei technologinės priemonės yra tinkamiausios ontologijoms į automatizuotų informacinių sistemų projektavimo procesus įtraukti, sukuriant taisykles, algoritmus ir juos realizuojančio įrankio prototipą.

1.2. Tyrimo objektas, sritis ir problema

Tyrimo objektas – *UML* profiliu pavaizduotų *OWL 2* ontologijų transformavimo į *OWL 2* ontologijų kalbą procesas.

Tyrimo sritis – *OWL 2* ontologijų vaizdavimas *UML* informacinių sistemų kūrimo procesuose, šių procesų automatizavimas taikant modeliais grindžiamas transformacijas ir *CASE* įrankių plėtimo technologijas.

Tyrimo problema – šiuo metu kuriama daug projektų, kuriuose naudojamos ontologijos, tačiau projektavimo procesuose naudojamuose *CASE* įrankiuose ontologijų redaktoriais sukurtas ontologijas reikia rankiniu būdu pavaizduoti *UML* modeliais, todėl gaištamas papildomas laikas ir išauga tikimybė, kad projekto modeliai nesutaps su tikrosiomis ontologijomis. Šią problemą galima išspręsti sukuriant *UML* profilį *OWL 2* ontologijoms vaizduoti ir *CASE* įrankio įskiepį, kuris leistų transformuoti *UML* profiliu pavaizduotas ontologijas į ontologijų kalbą *OWL 2*.

1.3. Ontologijos apibrėžimas

Pats terminas „Ontologija“ yra kilęs iš filosofijos srities, kur ši šaka nagrinėja egzistencijos klausimus, ryšius, tipus ir kategorijas pasireiškiančias būtyje. Graikų kalboje šis žodis reiškia - „būtis“. Egzistuoja bent keli ontologijos apibrėžimai. Vienas plačiausiai naudojamų apibrėžimų sako, kad ontologija tai tam tikros srities sąvokų visumos specifikavimas išreikštu pavidalu
(angl. „*explicit specification of a conceptualization”* T. R. Gruber, 1993 [7]).

Kitais žodžiais tariant, ontologija tai tam tikros dalykinės srities suderintas vaizdas, supratimas, aprašas, tai yra tą sritį nusakančių sąvokų visuma, tam tikros dalykinės srities modelis. Tai tarsi tam tikros bendruomenės požiūris į tam tikrą dalykinę sritį, kuri aprašyta susijusių sąvokų visuma. Ontologijos gali būti įvairios, nuo formalių iki taikomųjų. Šiame darbe naudojami taikomųjų ontologijų pavyzdžiai, kurie skirti siauresnių dalykinių sričių taikomiesiems uždaviniams spręsti.

Norint vizualizuoti ontologijas, jos turi būti aprašytos ontologijų kalba.

1.4. Ontologijų kalbų analizė

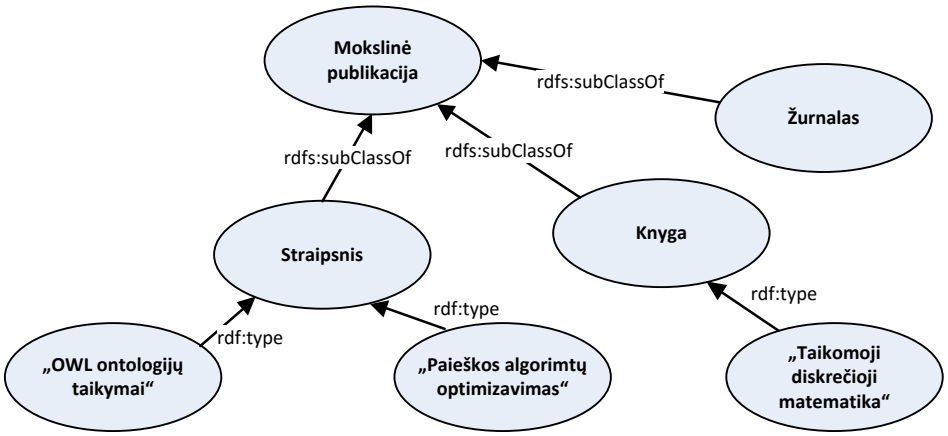
Ontologijos kalba – tai formali kalba, kurios tikslas aprašyti ontologijas. Tinkamos ontologijos kalbos pasirinkimas priklauso nuo dalykines srities. Kalba turi būti suprantama, lengvai išmokstama, suderinama su kitais įrankiais. *OWL* ontologijų aprašymo kalba teikia didelį žodyną savybėms ir klasėms aprašyti, aprašo ryšius tarp klasių. *OWL* pagerina galimybę automatiškai interpretuoti interneto turinį, lyginant su tuo ką gali *XML* ir *RDF*. Tai užtikrinama todėl, kad *OWL* teikia papildomas informacijos aprašymo galimybes. Ši ontologijų aprašymo kalba laikoma perspektyvia kuriant semantinį tinklą, todėl yra gana platus ontologijų kūrimo įrankių pasirinkimas.

1.4.1. RDF analizė

Semantinio tinklo resursams aprašyti konsorciumas W3C rekomenduoja RDF (angl. *Resource Description Framework*) kalbą. Tai yra XML ženklinimo kalbos standartu paremtas metaduomenų aprašymo formatas. RDF kalboje kiekvienas teiginys yra sudarytas iš 3 elementų: objekto, objekto savybės ir objekto savybės reikšmės. Dažnai šis trejetas suvokiami kaip veiksnys (angl. *subject*) tarinys (angl. *predicate*) ir papildinys (angl. *object*).

Kiekvienas iš *RDF* trejeto elementų identifikuojami unikaliu *URI* (angl. *Uniform Resource Identifier*) identifikatoriumi. Norint reprezentuoti faktų, galiojančių dalykinėje srityje, visumą, šiuos trejetus reikia kombinuoti (jungti į grafą). *RDF* schema (*RDFS*) praplečia *RDF* kalbos žodyną iki galimybės aprašyti specifines klases, hierarchijos ryšius ir savybes.

RDF kalbą, papildyta *RDF* schema, galima, vadinti riboto išraiškingumo žinių ontologija. Paveiksle 1.1 pavaizduota supaprastinta mokslinių publikacijų klasifikavimo ontologija panaudojant bazinius *RDF* klasifikavimo ir priskyrimo atributus*: rdfs:subClassOff, rdf:type.*



1.1 pav. Mokslinių publikacijų ontologijos klasių taksonomija RDFS kalba

1.4.2. XML analizė

XML pažodžiui reikštų – išplėstinė žymėjimo kalba (angl. *Extensible Markup Language*). Ši kalba yra *W3C* konsorciumo rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba. *XML* kalba tikslas yra palengvinti duomenų keitimąsi tarp skirtingų sistemų, dažniausiai sujungtų internetu. Šios kalbos pagrindinis struktūros vienetas yra elementas, kuris visada turi turėti vardą. Elementas taip pat gali turėti:

- neribotą kiekį atributų, atributas turi turėti savo vardą bei reikšmę;
- dukterinius elementus savo viduje;
- su elementu susijusį tekstą kiekvieno elemento viduje.

Žemiau pateikiamas *XML* kalbos pavyzdys:

```
<asmenys>
   <asmuo atributas=”identifikatorius”>
      <vardas>Jonas</vardas>
      <pavarde>Jonaitis</pavarde>
   </asmuo>
</asmenys>
```

XML kalboje gali būti tik vienas šakninis elementas kuris gali turėti begalybe dukterinių elementų. Šiuo atveju pavyzdyje šakninis elementas yra „<asmenys>“, šis elementas gali saugoti tiek dukterinių elementų, šiuo atveju asmenų, kiek tik reikia. Elementas „<asmuo>“ lygiai taip pat gali saugoti dukterinius elementus savo viduje, kurie gali saugoti dukterinius elementus savo viduje ir taip kol pasiekiamas tikslas.

XML elementai gali būti automatiškai randami pagal vardą arba pagal kelią. Tačiau *XML* tas pats kelias gali vesti į kelis elementus.

1.4.3. OWL analizė

Viena iš plačiausiai naudojamų ontologijų aprašymo kalbų yra būtent *OWL 2* (angl. *Web Ontology Language*). Ši kalba yra RDF kalbos plėtinys paremtas RDF/XML struktūra. *OWL 2* palaiko tokius sintaksės formatus kaip: *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. Šios išraiškos ir jų semantiniai tarpusavio ryšiai vadinami ontologija.

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

OWL Lite yra 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, ir ji turi mažesnį formalų sudėtingumą.

OWL DL suteikia maksimalų ekspresyvumą, išlaikant skaičiavimų baigtumą (užtikrinama, kad visos išvados yra garantuotai suskaičiuojamos) 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 poklasė, bet klasė negali būti kitos klasės atskiras atvejis). *OWL DL* pavadinimas išplaukia iš jo suderinamumo su apibūdinimo logika (angl. *description logics*) – tyrinėjimų srities, kuri studijavo logiką nuo pat formalaus *OWL* įkūrimo.

OWL Full suteikia maksimalų ekspresyvumą, bet nesuteikia garantijų dėl skaičiavimų baigtumo. Deja, panašu, kad jokia samprotaujanti programinė įranga negalėtų palaikyti pilną pagrindimą kiekvienai *OWL Full* ypatybei.

Kalbant apie ontologijos sąvokas, galima išskirti pagrindinius ontologijos elementus kurie sutelpa į OWL Lite: klasės (angl. *Class*), objektų savybės (angl. *Object property*), duomenų savybės (angl. *Data property*), individai (angl. *Individuals*) ir likusios aksiomos. Visiems šiems elementams būdingos specifinės charakteristikos.

Klasės gali turėti poklasių (angl. *Sub Class Of*) aksiomas, kurios nusako ryšius tarp deklaruotų klasių. Taip pat klasės gali turėti nesusikertančių sąjungų (angl. *Disjoint Union*) aksiomas, kurios nusako, jog klasės negali būti susikertančios. Pavyzdžiui jei turime vieną klasę „Vyras“, kitą klasę „Moteris“ ir jos sudaro apibendrinama rinkinį (angl. *Generalization Set*) kaip nesusikertančią sąjungą, tai reiškia, kad asmuo bus arba vyras arba moteris, bet neįmanoma, kad jis būtų ir vyras ir moteris.

Objektų savybėms būdingos charakteristikos tokios kaip funkcinės objektų savybės (angl. *Functional Object Property*), kuri nurodo, kad savybė gali turėti tik vieną unikalią reikšmę kiekvienam individui. Taip pat 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 posavybe. Ir žinoma 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 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 yra klasių egzemplioriai

Semantinis tinklas paremtas naujomis *WEB* kalbomis tokiomis kaip *XML*, *RDF* ir *OWL* bei įrankiais kurie naudoja šias kalbas. Semantinio tinklo duomenims aprašyti naudojamos tekstų žymėjimo kalbos:

- *XML;*
- ontologijų kalba *OWL;*
- resursų aprašymo kalba *RDF*.

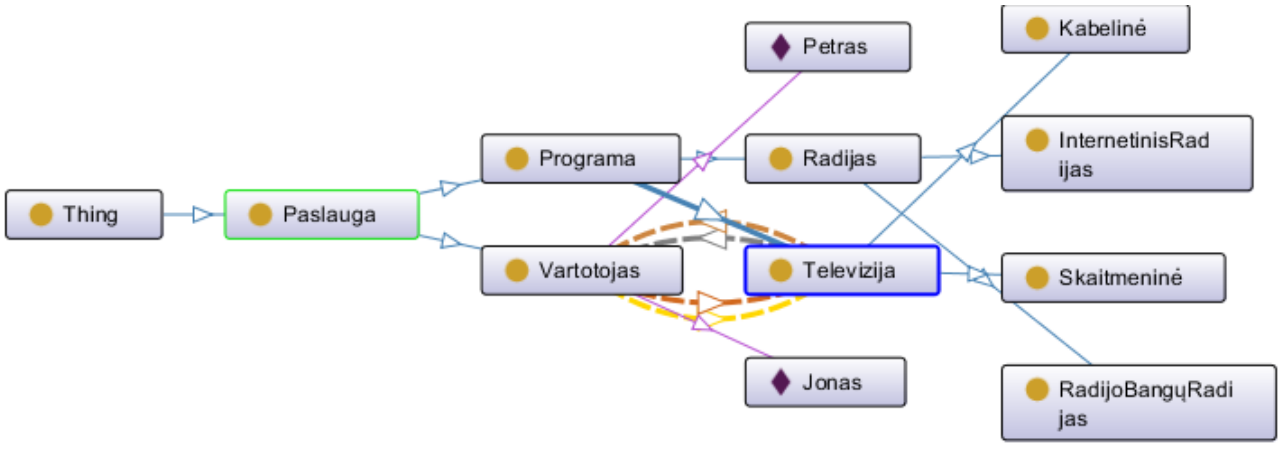
1.5. Ontologijų įrankių analizė

Ontologijų įrankių analizė atliekama norint išsiaiškinti geriausią sprendimą ontologijoms vizualizuoti. Įrankių analizės tikslas išanalizuoti kelis įrankius ir remiantis analizės rezultatais įvertinti analizuotus įrankius taip išsirenkant tinkamiausią sprendimą.

1.5.1. „Protégé“ analizė

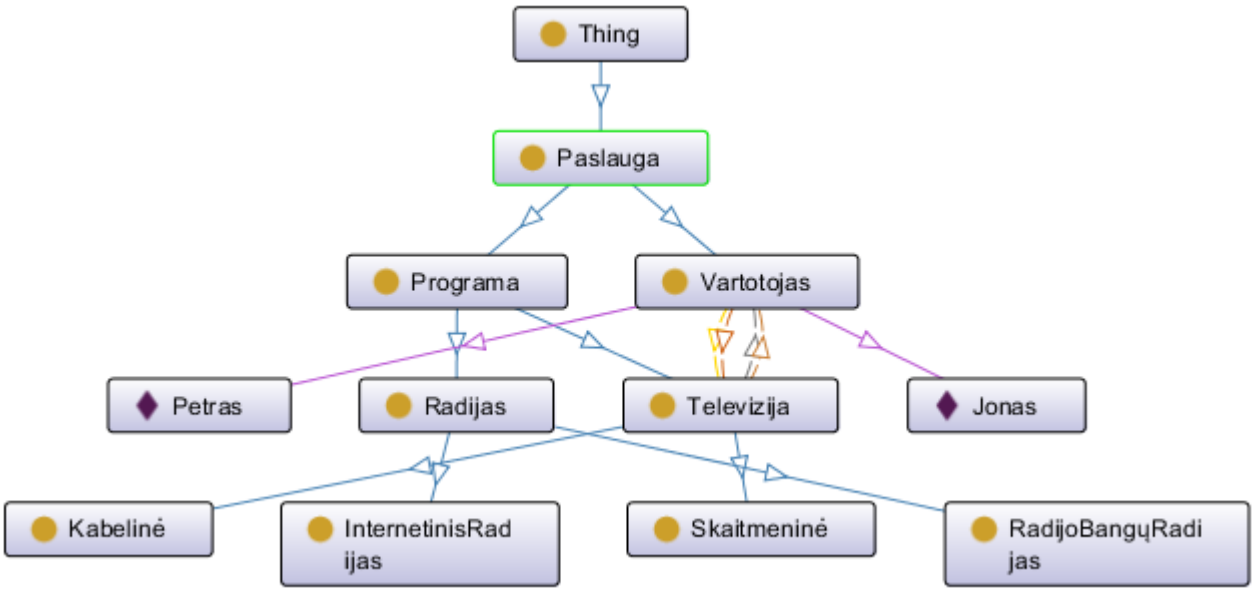
„Protégé“ įrankis leidžia peržiūrėti ontologijas grafiškai, naudojant „OntoGraf“ įskiepį, tačiau neleidžia ontologijų grafiškai kurti ar redaguoti. Įrankyje galima išsidėstyti ontologiją kaip patogiau, bet negalima jos išsaugoti. Ontologija labai gražiai išdėstoma grafiškai ryšių sąskaita. Tai reiškia, kad klasių hierarchija išdėstoma taip, kad visos klasės išsirikiuotų pagal paveldimumą, tačiau dėl to nukenčia ryšiai, kurie pradeda pintis ir lipti vieni per kitus.

Ontologija šiame įrankyje pradedama vaizduoti suskleista – rodoma bazinė klasė su galimybe ją išskleisti. Tačiau adaptuojant diagramą į patogesnį vaizdą skaitymui neišskleidus visos klasių hierarchijos ir adaptavimo metu išskleidus bent vieną klasę viską teks pradėti iš naujo, nes įrankis vėl išrikiuoja viską pagal klasių hierarchiją. Pradinis išskleistos ontologijos vaizdas parodytas 1.2 pav.



1.2 pav. „OntoGraf“ vizualizuota ontologija

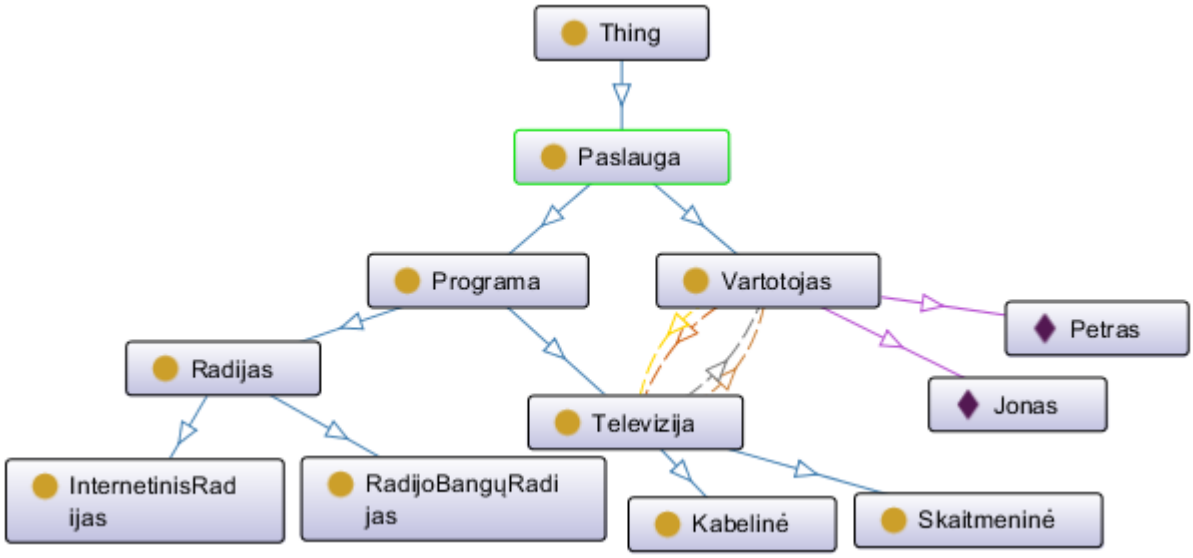
„OntoGraf“ leidžia pasirinkti kelias opcijas kaip išdėstyti diagramą, tačiau tai neišsprendžia ryšių problemos, nes bet kuriuo atveju grafikas rikiuojamas pagal klasių hierarchiją. Žemiau esančiame 1.3 pav. pateikiamas pavyzdys kaip atrodo schema pasirinkus kitą rikiavimo metodą.



1.3 pav. „OntoGraf“ vizualizuota ontologija pakeitus rikiavimo metodą

Vienas iš didesnių pastebėtų trūkumų šiame vizualizavimo sprendime yra tai, kad atvirkštinės objektų savybės nors ir yra dvipusės, jos atvaizduojamos atskirai. Dėl šios priežasties dvigubai išauga ryšių tinklas su lyg kiekviena atvirkštine objektų savybe, o tai labai gadina skaitomumą, nes grafikas gaunasi labai apkrautas pertekliniais ryšiais. Šiame darbe „MagicDraw“ įrankyje ši problema išspręsta atvirkštines objektų savybes atvaizduojant ant to paties ryšio galų.

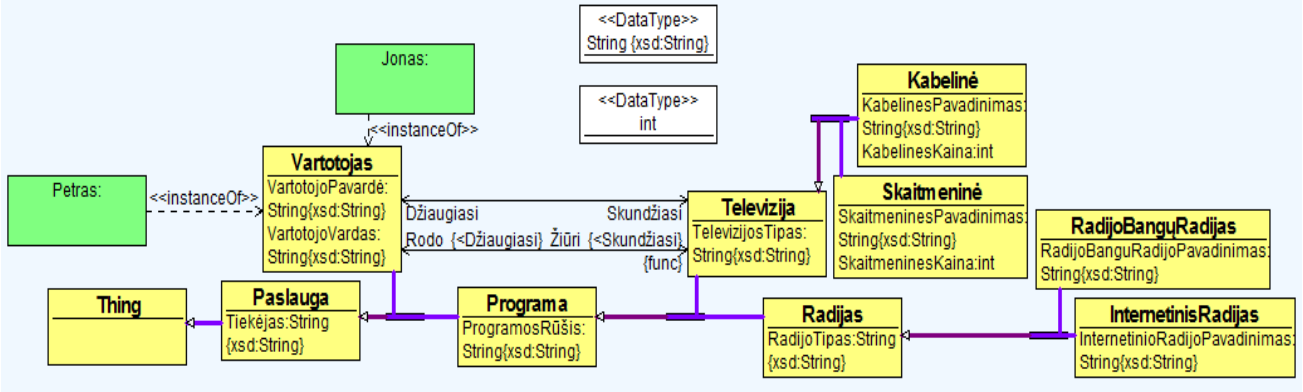
Apibendrinant šį ontologijų vizualizavimo sprendimą galima pasakyti jog tai nėra labai patogus įrankis vizualizuoti ontologijas, tačiau ignoruojant kai kurias ryšių problemas ontologiją gan nesunkiai galima adaptuoti į skaitymui patogesnį vaizdą, kaip parodyta 1.4 pav.



1.4 pav. „OntoGraf“ vizualizuota diagrama perstumdyta skaitymo patogumui

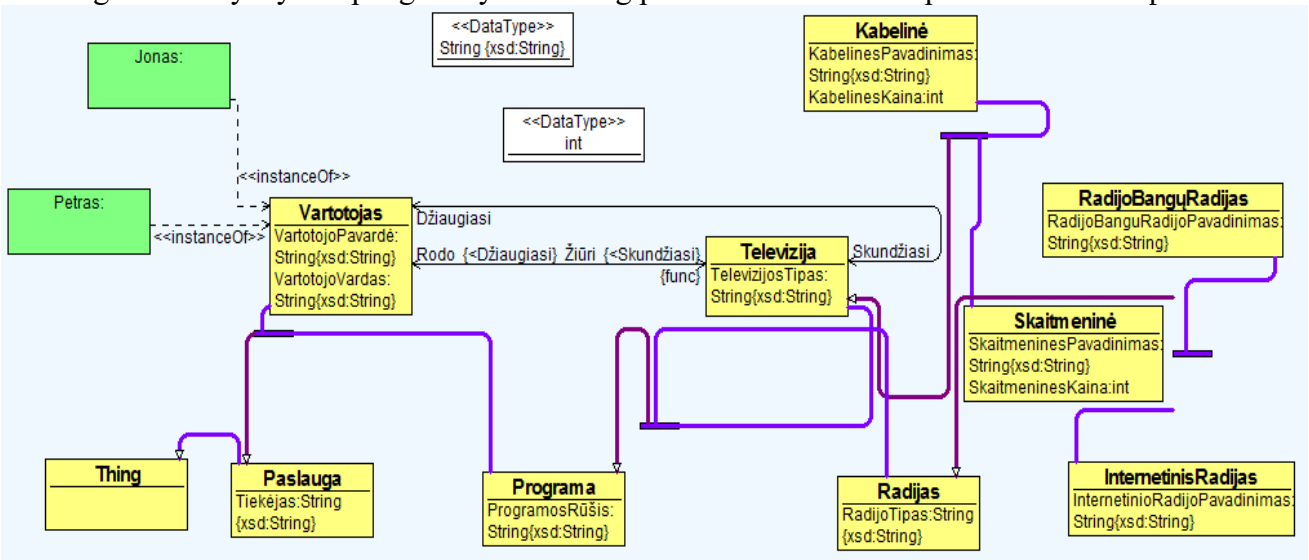
1.5.2. „OWLGrEd“ analizė

„OWLGrEd“ įrankis leidžia skaityti, redaguoti, kurti ir išsaugoti ontologijas grafiniu būdu. Tačiau eksperimento metu buvo nustatyta, jog naudojantis šiuo įrankiu norint persidėlioti schemą pagal save yra sugaištama kur kas daugiau laiko nei su „Protégé“ ar „MagicDraw“. Pradinis ontologijos vaizdas „OWLGrEd“ įrankyje sudėliojamas tvarkingai, pavyzdys matomas 1.5 pav., tačiau esant poreikiui adaptuoti ontologijos schemą į kitokį vaizdą susiduriama su nepatogumais.



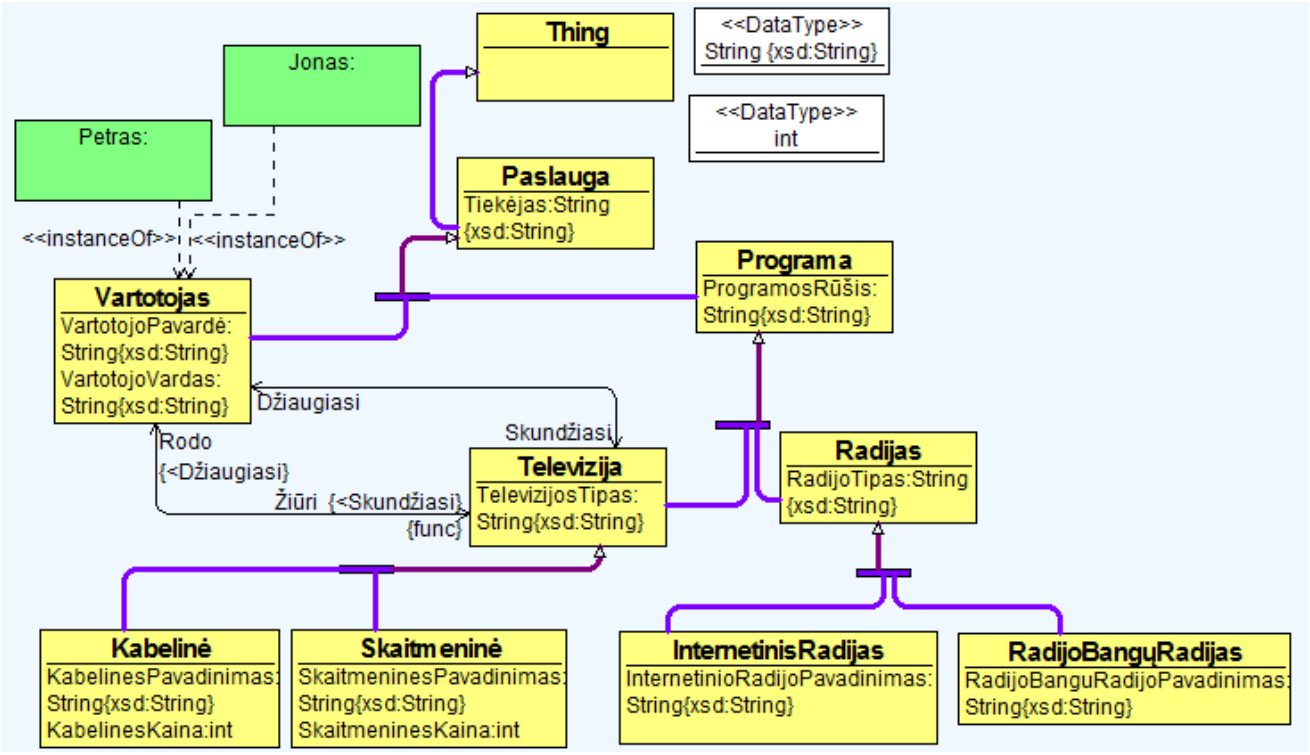
1.5 pav. „OWLGrEd“ vizualizuota ontologija

Pabandžius adaptuoti pavyzdinę ontologijos schemą labiausiai užkliuvo nelankstūs ryšiai. Tai yra, kad kilnojant klases ryšiai darosi sunkiai kontroliuojami. Juos reikia tampyti atskirai, kas užima nemažai papildomo laiko. Ryšiu tampymas taip pat labai nelankstus, todėl norint sutvarkyti vieną ryšį jį gali tekti tampyti per kelias vietas, o jei ryšys priklausomas ir nuo kitų ryšių tada situacija tampa dar sudėtingesnė. Pavyzdys kaip elgiasi ryšiai tiesiog perkėlus kelias klases pavaizduotas 1.6 pav.



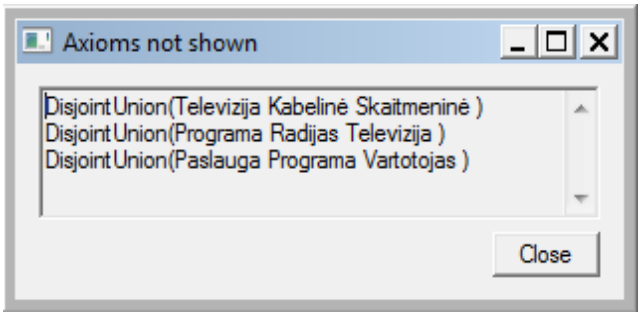
1.6 pav. „OWLGrEd“ vizualizuojamos ontologijos ryšiai

„OWLGrEd“ taip pat turi galimybę pasirinkti, norime schemą matyti vertikaliai ar horizontaliai, tačiau išbandžius šias dvi opcijas jau nebegrįžtama prie vaizdo matomo 1.5 pav. Schemoje atsiranda chaosas, nutrūkę ryšiai, bei vėl pakitęs išdėstymas. Adaptavus ontologiją gautas vaizdas pateikiamas 1.7 pav.



1.7 pav. „OWLGrEd“ vizualizuojama adaptuota ontologija

„OWLGrEd“ įrankis vizualizuojant šį ontologijos pavyzdį perspėja naudotoją jog ne visos aksiomos yra rodomos. Šiuo atveju kaip pavaizduota 1.8 pav. įrankis perspėja apie nerodomas „DisjointUnion“ aksiomas.



1.8 pav. Neatvaizduojamos aksiomos

1.6. *UML* kalbos analizė

UML yra kalba paremta grafiniu modeliavimu, kuri apima keturiolika diagramų rūšių bei turi vieningą terminologiją. UML ypatingai geras įrankis vizualizavimui, specifikavimui bei dokumentavimui.

UML kalbos privalumai:

- supaprastėja komunikacija, visi kalba ta pačia kalba, iššvaistoma mažiau laiko;
- reikalavimai lengviau apibrėžiami ir dokumentuojami, mažiau pamirštų vietų;
- vartotojai įtraukiami į programos kūrimą nuo pat pradžių, mažiau perdarymų pabaigoje;
- priemonė išsaugoti sukauptas žinias firmoje, net jei žmonės ją palieka;
- sutaupo laiko susipažįstant su jau sukurtomis sistemomis.

Šiuo metu *UML* yra labiausiai paplitęs programinės įrangos specifikavimo standartas, palaikomas įvairių gamintojų: „Borland“, „IBM“, „Telelogic“, „NoMagic“. Lietuvoje yra kuriama visame pasaulyje žinoma „MagicDraw“ - *UML CASE* priemonė.

Vieninga modeliavimo kalba *UML* (angl. *Unified Modeling Language*) - skirta programinei įrangai, verslo logikai ir kitoms sistemoms aprašyti, vizualizuoti ir dokumentuoti. *UML* - svarbi objektinio programavimo proceso dalis. Naudojant *UML* galima vizualiai atvaizduoti tokius sistemos architektūros elementus kaip:

- aktorius (angl. *actors*);
- verslo procesus (angl. *business processes*);
- komponentus (angl. *components*);
- veiksmus (angl. *activities*);
- programavimo kalbos elementus (angl. *programming language statements*);
- duomenų bazių elementus (angl. *database schemas*).

Pirmiausia svarbu pastebėti skirtumą tarp *UML* modelio ir sistemos diagramų rinkinio. Diagramos yra tik dalinis grafinis sistemos modelio atvaizdavimas. Į modelį įeina ir modelio semantika – dokumentacija, apibūdinanti modelio elementus ir diagramas.

UML turi 14 diagramų tipų, suskirstytų į dvi kategorijas. Septynios diagramos apibūdina modelio objektų struktūras, kitos septynios – būsenas bei sąveikas. *UML* naudojami diagramų tipai:

1. Veiklos diagrama - modeliuoja dinaminę sistemos elgseną (vaizduojami veiksmai);
2. Panaudos atveju diagrama - apibūdina funkcinį sistemos veikimą vartotojo požiūriu;
3. Sekos diagrama - apibūdina dinaminę veikėjų (aktorių), sistemos objektų ir sistemos sąveiką;
4. Bendradarbiavimo diagrama - apibūdina pranešimus, siunčiamus tarp komponentų;
5. Klasių diagrama - apibūdina statinę sistemos struktūrą: objektus, atributus, asociacijas;
6. Būsenų diagrama - apibūdina vieno sistemos objekto dinaminį elgesį kaip būsenų kaitą;
7. Komponentų diagrama - aprašo sistemoje naudojamus komponentus;
8. Išdėstymo diagrama - aprašo fizinį sistemos diegimą;
9. Realizacijos diagrama – nusako sistemos komponentus ir parodo fizinę sistemos struktūrą;
10. Objektų diagrama - visiškai ar iš dalies atvaizduoja modelio struktūrą tam tikru laiko momentu;
11. Paketų diagrama - parodo sistemos vidinę organizaciją. Sistemos struktūrines dalis atitinka paketai, į juos dedami tų dalių modeliai, priklausomybės tarp paketų modeliuoja dalių sąryšius;
12. Profilių diagrama - naudojama meta modelio lygmenyje ir parodo klasių stereotipus ir profilius;
13. Sudedamųjų struktūrų diagrama - apibūdina vidinę klasės struktūrą ir galimas šios struktūros sąveikas;
14. Laiko diagrama - 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.

Taigi tam puikiai tinka „MagicDraw“ įrankis, kuris palaiko modeliais grindžiamą projektavimą. Šiuo atveju iš visų *UML* siūlomų diagramų aktualiausia yra klasių diagrama, nes ji gali pasiūlyti daug atitikmenų su ontologijų elementais jų vizualizavimui.

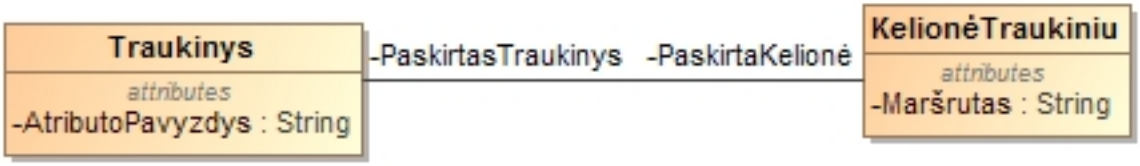
Pati *UML* klasių diagrama susideda iš stačiakampių – klasių, kurios apima atributus ir operacijas. Klasės pavyzdys pateiktas 1.9 pav.

1.9 pav. Klasės pavyzdys

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

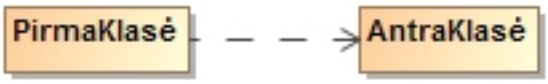
- asociacija;
- generalizacija;
- priklausomybė.

Asociacija skirta specifikuoti dvipusį ryšį tarp klasių. Asociacijos pabaiga yra vadinama vaidmeniu (*angl. role*) kuriam galima suteikti pavadinimą. Asociacijos vaidmenų pavyzdys pateiktas 1.10 pav. Traukinys kelionėje užima paskirto traukinio vaidmenį, o kelionė traukiniu vaidina paskirtos kelionės vaidmenį.



1.10 pav. Asociacijos pavyzdys

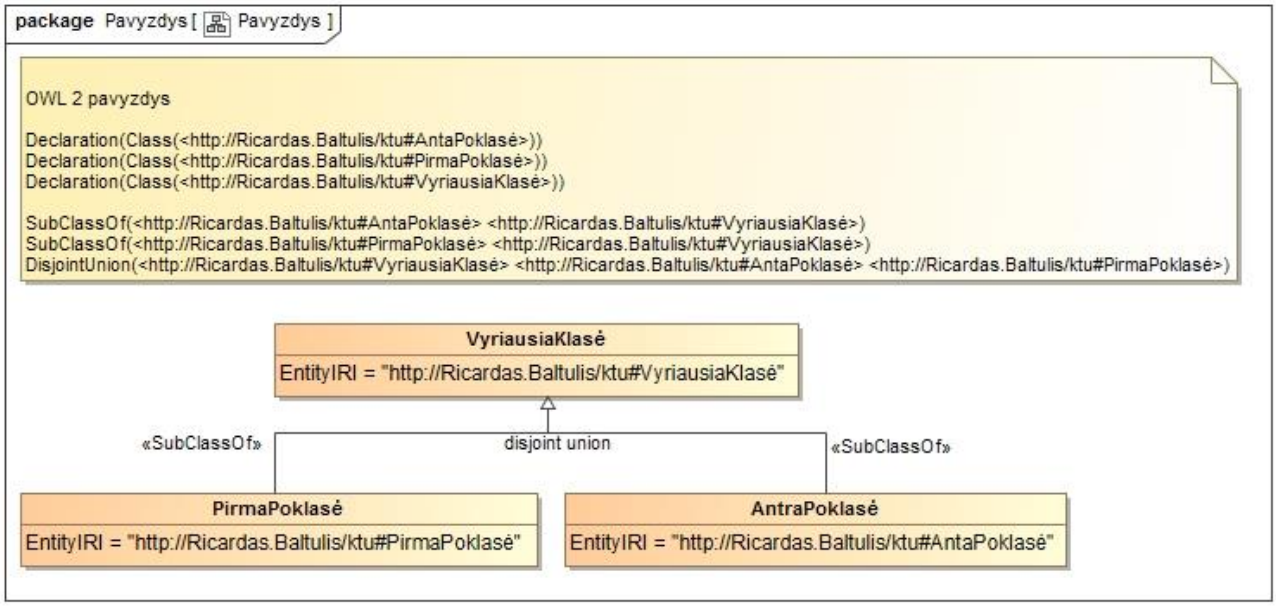
Generalizacija tai ryšys, kuriame viena klasė yra kitos klasės konkretizacija. Priklausomybė tai vienkryptis ryšys, nurodantis, kad vienas elementas naudoja kitą elementą. Priklausomybės pavyzdys pateikiamas 1.11 pav.



1.11 pav. Priklausomybės pavyzdys

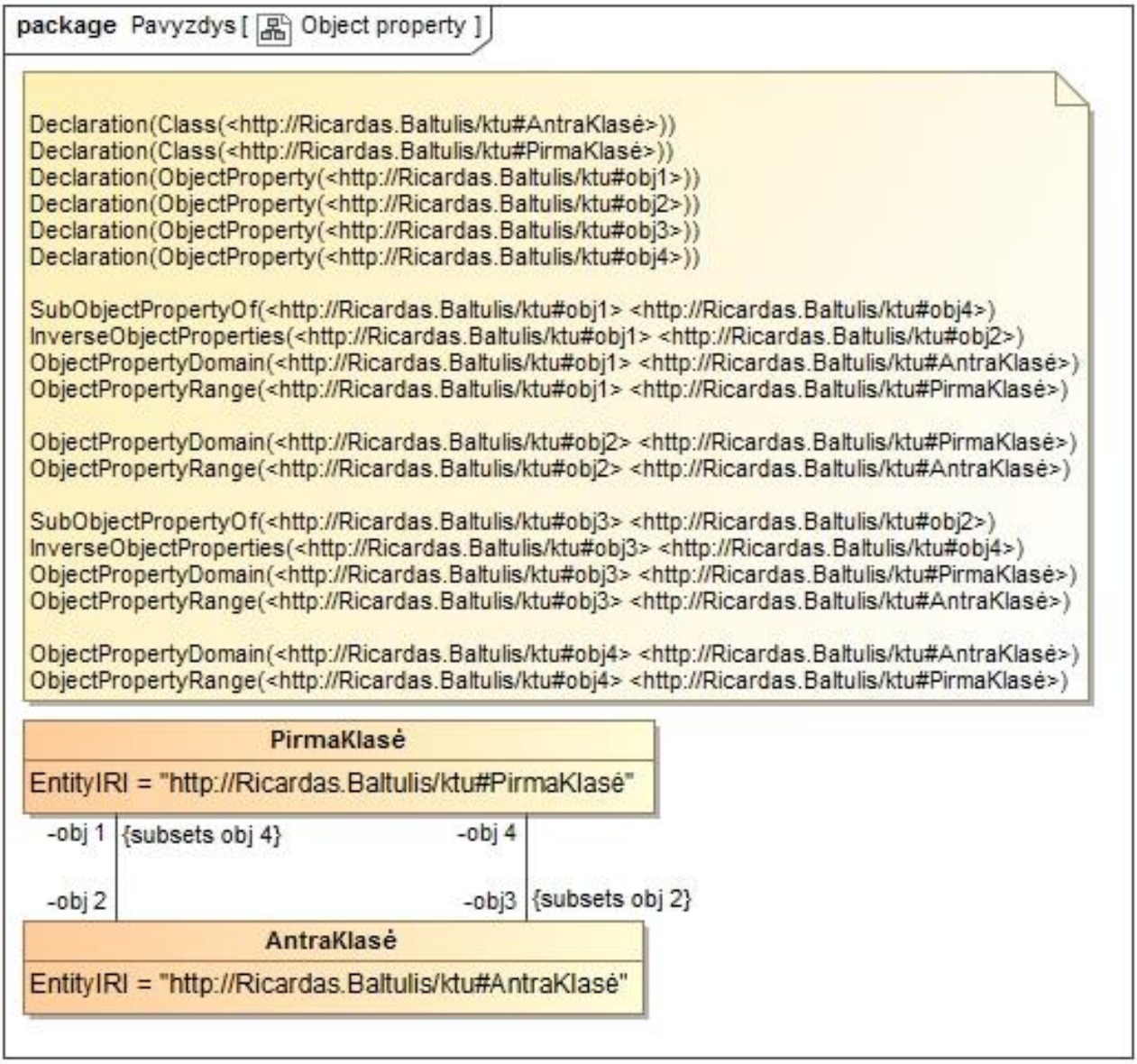
Taigi *UML* klasių diagrama turi pakankamai elementų atvaizduoti ontologijų elementus. Šiuo atveju mums aktualūs prieš tai analizėje jau minėti OWL Lite pagrindiniai komponentai: klasės, objektų savybės, duomenų savybės, individai ir visų jų aksiomos.

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 „<<OWL2 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 1.12 pav.



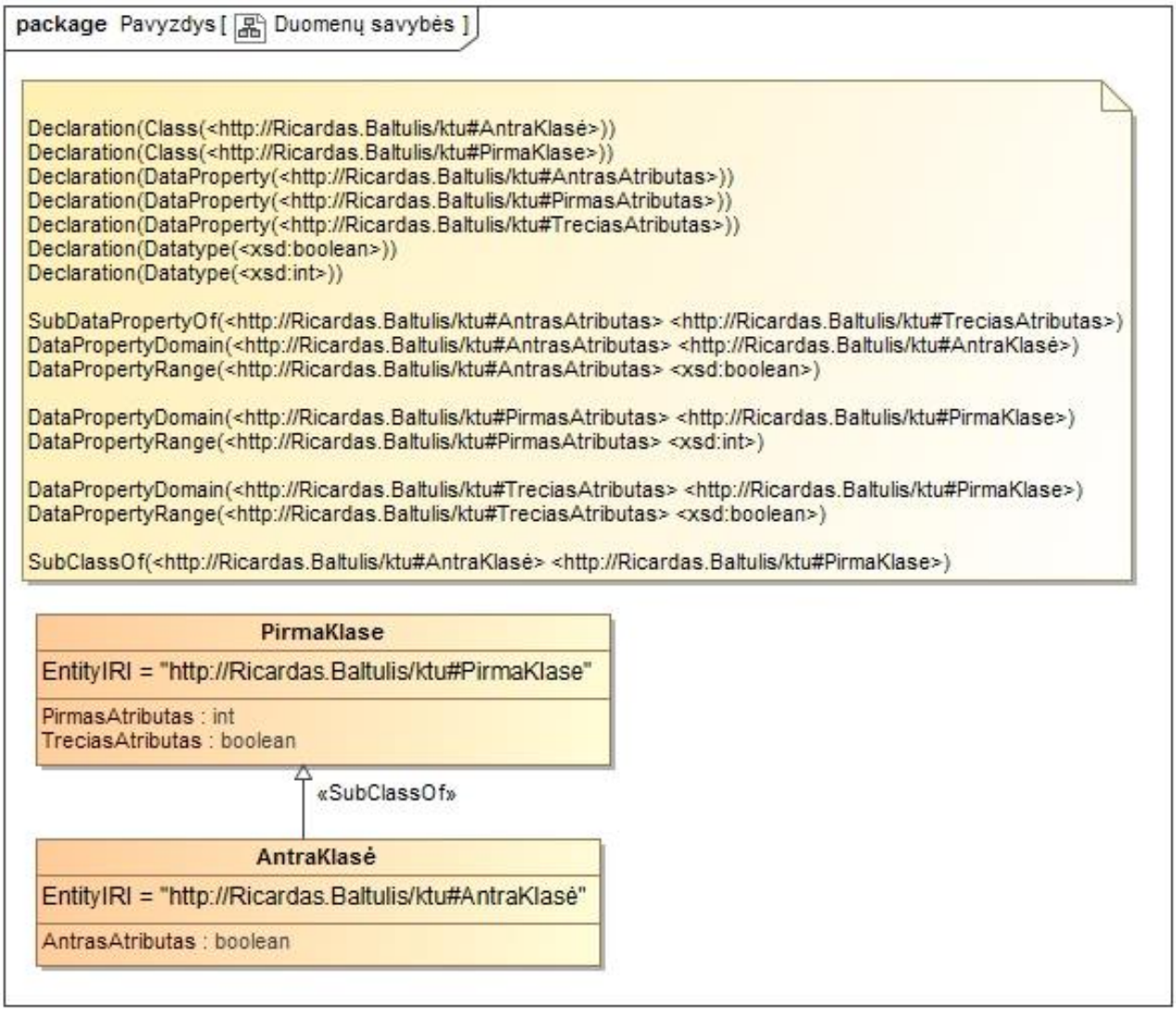
1.12 pav. Klasių ir klasės aksiomų pavyzdys *UML* ir *OWL 2*

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. Taigi transformacijos metu sukurtas „MagicDraw“ įrankio įskiepis atpažintai objektų savybei priskirs ir domeną bei sritį. Įskiepis taip pat geba nustatyti ar objektų savybė yra atvirkštinė. Taip pat kiekvienai objektų savybei galima nustatyti poaibį (angl. *Subset*). Tai transformacijos metu būtų traktuojama kaip specializuojanti objektų savybė (angl. *Sub Object Property Of*). Objektų savybių ir jų aksiomų pavyzdys *UML* ir *OWL 2* parodytas 1.13 pav.



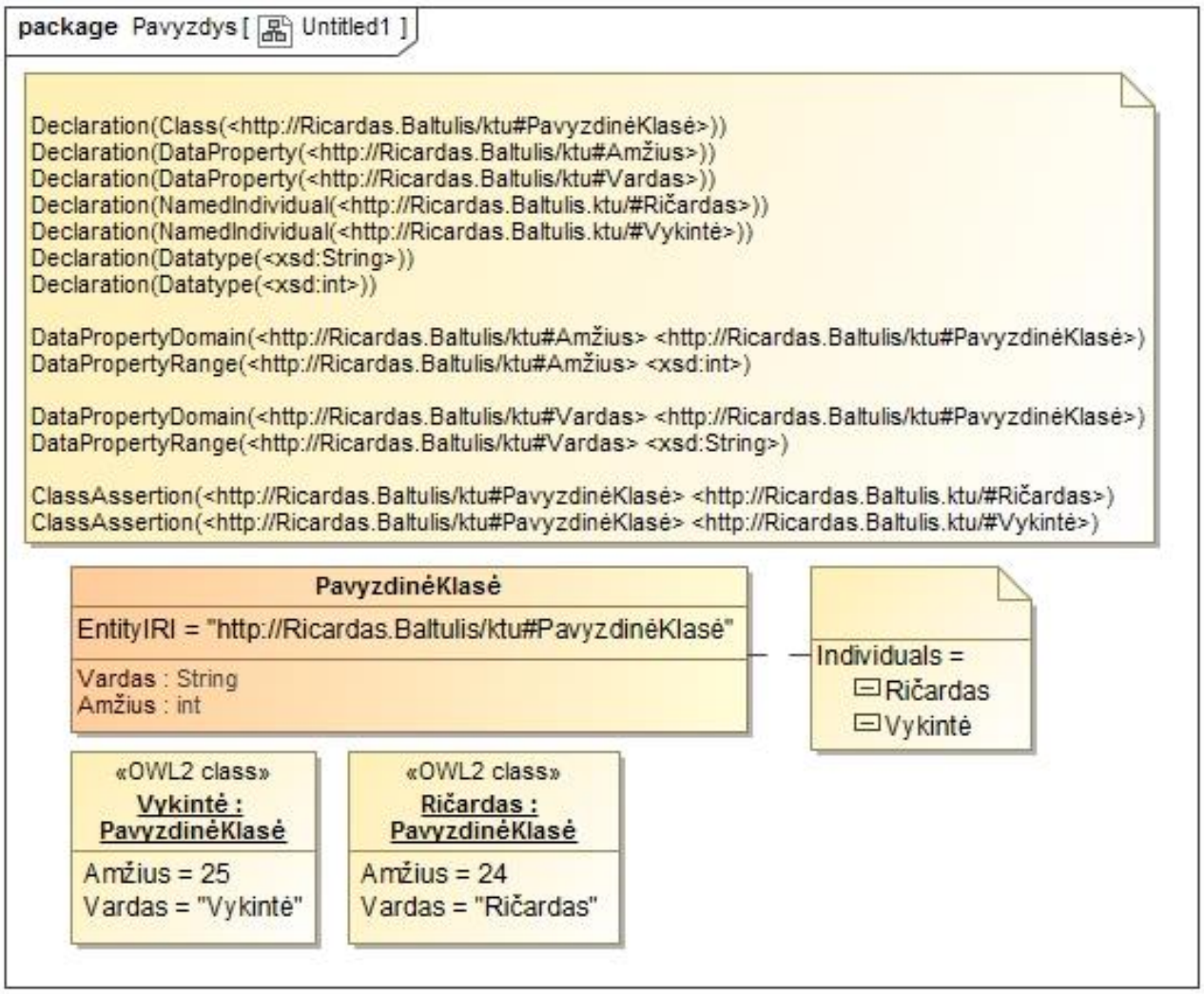
1.13 pav. Objektų savybių ir jų aksiomų *UML* ir *OWL 2* pavyzdys

Duomenų savybės *UML* klasių diagramoje prilygsta atributams. Norint sukurti duomenų savybes užtenka pridėti klasei atributą su stereotipu „<<DataProperty>> “ iš OWL profilio ir nurodyti atributo tipą. Tipas nusako ar atributas yra „String“ tipo, ar „Integer“, ar „Boolean“ ir t.t. Transformacijos metu tipai bus priskiriami atitinkamai duomenų savybių sričiai (angl. *Data Property Range*), o esama atributo klasė bus kaip domenas. Analogiškai objektų savybių atvejui, duomenų savybėms galima nustatyti poaibį, kas transformacijoje prilygs specializuojančiai duomenų savybei (angl. *Sub Data Property Of*). Duomenų savybių ir jų aksiomų pavyzdys *UML* ir *OWL 2* parodytas 1.14 pav.



1.14 pav. Duomenų savybių ir jų aksiomų *UML* ir *OWL 2* pavyzdys

Individu vizualizavimas realizuotas per egzempliorius (angl. *Instances*), nurodant individo pavadinimą bei užpildant egzemplioriaus reikšmes. Kiekvienos klasės individų peržiūrai naudojamas prisegamas komentaras, kuriame reikia pasirinkti ką norime matyti. Šiuo atveju komentaras turi būti prisegtas prie klasės kurios individus norime matyti, tada prisegamo komentaro specifikacijoje pasirenkame jog norime matyti individus. Tokiu būdu galime matyti kiekvienos klasės individus. Individų ir jų aksiomų pavyzdys *UML* ir *OWL 2* parodytas 1.15 pav.



1.15 pav. Individų ir jų aksiomų *UML* ir *OWL 2* pavyzdys

Klasės, objektų savybės ir duomenų savybės turi turėti savo unikalų IRI. Jis ontologijoje naudojamas kaip unikalus identifikatorius.

1.8. Esamų problemos sprendimo metodų analizė

1.1 lentelė. Esamų sprendimų palyginimas

Palyginimo kriterijus	**Protégé**	**OWLGrEd**	**MagicDraw**
Grafinis redagavimas/kūrimas	-	+	+
Grafiniai elementai objektų savybėms	Du ryšiai	Vienas ryšys	Vienas ryšys
Palaiko UML	-	-	+

Įvertinus gautus palyginimo rezultatus 1.1 lentelėje 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

Darbo tikslas - sudaryti galimybę įtraukti ontologijų kūrimą į informacinių sistemų projektavimą, sukuriant *UML* profiliu pavaizduotų *OWL 2* ontologijų transformavimo į *OWL 2* kalbą taisykles bei algoritmus ir juos realizuojančio įrankio prototipą.

Šiam tikslui pasiekti iškelti uždaviniai:

1. Išanalizuoti:
   1.1. Ontologijų kalbą *OWL 2*, 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. Sukurti *UML OWL 2* profilį, sudaryti transformavimo taisykles ir algoritmus, suprojektuoti transformavimo įrankį;
3. Realizuoti įrankio prototipą;
4. Atlikti eksperimentą, kuris leistų įvertinti sprendimo tinkamumą ir efektyvumą;
5. Apibendrinti tyrimo rezultatus.

1.10. Siekiamo sprendimo apibrėžimas

Norint sėkmingai panaudoti ontologijas informacinių sistemų kūrime yra svarbu turėti įrankius leidžiančius projektuojant sistemą projektuoti ir pačias ontologijas tais pačiais įrankiais. Ontologijų inžinieriai daug sėkmingiau suderintų vienodą požiūri į dalykinę sritį jeigu naudotų grafines priemones ontologijoms vizualizuoti. Ontologijos paskirtis yra leisti skirtingiems atstovams į tą patį dalyką žiūrėti vienodai ir jį suprasti vienodai. Ontologijų perkėlimas į *UML* leistu sukurtus modelius transformuoti ir į kitas formas, taip pat susieti ontologiją su kuriamos sistemos reikalavimais ar sistemos architektūra.

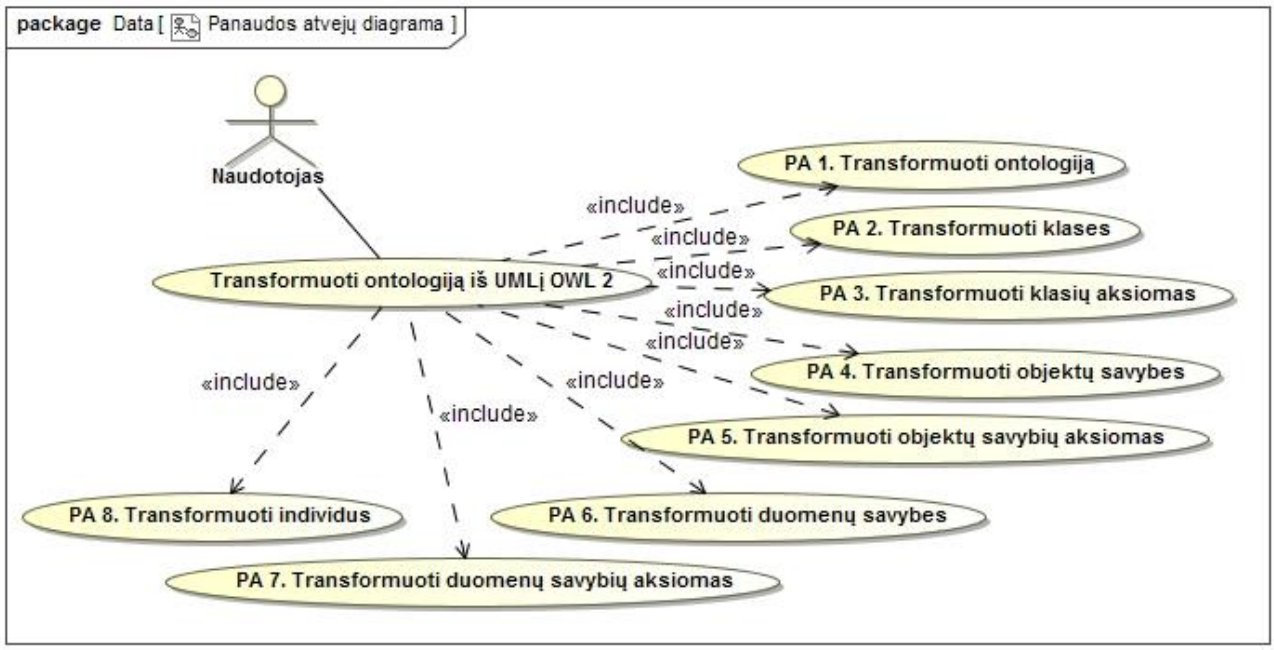
1.11. Analizės išvados

1. Atlikus analizę buvo išanalizuoti pagrindiniai ontologijos kalbos elementai, kuriuos reikėtų transformuoti ir prieita prie išvados, kad reikia transformuoti klases, objektu savybes, duomenų savybes, individus ir jų visų aksiomas tokias kaip duomenų tipai, domenai, sritys, atvirkštinės savybės, nepersikertančių elementų sąjunga, poklasės, specializuojančios objektų ir duomenų savybės. Šie elementai yra baziniai visuose kalbos lygiuose. Tai kaip pamatas, ant kurio gali būti statoma visa kita.
2. Atlikus įrankių analizę ir palyginus juos pagal išsikeltus kriterijus, problemos sprendimui buvo pasirinktas „MagicDraw“ įrankis. Vienas pagrindinių argumentų lėmusių sprendimą buvo modeliais grindžiamo sistemų projektavimo galimybė. Tai suteikia galimybę įtraukti ontologijų kūrimą į informacinių sistemų projektavimą.

2. OWL2 TRANSFORMACIJOS ĮSKIEPIO SPRENDIMO REIKALAVIMŲ SPECIFIKACIJA IR PROJEKTAS

2.1. Reikalavimų specifikacija

Ontologija modeliuojama naudojant „No Magic“ kompanijos kuriamą įrankį „MagicDraw“. Atliekant magistrinį darbą buvo naudota 18.2 versija. Ontologijos modeliavimui naudojamas magistrinio darbo metu sukurtas *OWL* profilis. Žemiau pateiktoje panaudojimo atvejų diagramoje (2.1 pav.) atsispindi sukurto įskiepio apimtis.



2.1 pav. *OWL2* transformacijos įskiepio panaudojimo atvejų diagrama

Žemiau esančiose 2.1 - 2.8 lentelėse pateikiami kiekvieno atvejo aprašymai lentelėmis.

2.1 lentelė PA 1 aprašymo lentelė

ID	PA 1
Pavadinimas	Transformuoti ontologiją
Aprašymas	Įskiepis transformuoja ontologijos diagramos paketą į ontologijos aksiomą. Ontologijos paketas turi turėti „<<OWL2 ontology>> “stereotipą ir unikalų *IRI*.
Aktoriai	Naudotojas
UML atitikmuo	Paketas su ontologijos stereotipu ir *IRI*.
OWL atitikmuo	Ontology(<*IRI*>

2.2 lentelė PA 2 aprašymo lentelė

ID	PA 2
Pavadinimas	Transformuoti klases
Aprašymas	Įskiepis transformuoja klases pavaizduotas *UML* klasių diagrama, kurios turi stereotipą „<<OWL2 class>> “ ir unikalų *IRI*.
Aktoriai	Naudotojas
UML atitikmuo	*UML* klasių diagramos klasė su stereotipu ir *IRI*.
OWL atitikmuo	Declaration(Class(*IRI*))

2.3 lentelė PA 3 aprašymo lentelė

ID	PA 3
Pavadinimas	Transformuoti klasių aksiomas
Aprašymas	Įskiepis transformuoja poklasių ryšius kurie turi stereotipą „<<SubClassOf>> “ į poklasių aksiomas. Taip pat transformuoja generalizacijos ryšius kurie pažymėti kaip nepersikertantys į nepersikertančių elementų sąjungą (angl. *disjoint union*).
Aktoriai	Naudotojas
UML atitikmuo	Generalizacijos rinkinys nustatytas kaip nepersikertančių elementų sąjunga. Generalizacija su poklasės stereotipu.
OWL atitikmuo	DisjointUnion(*IRI1 IRI2 IRI3*) SubClassOf(*IRI1 IRI2*)

2.4 lentelė PA 4 aprašymo lentelė

ID	PA 4
Pavadinimas	Transformuoti objektų savybes
Aprašymas	Įskiepis transformuoja asociacijos ryšius su jų rolėmis į objektų savybes ontologijų kalboje. Rolės turi turėti „<<ObjectProperty>>“ stereotipą.
Aktoriai	Naudotojas
UML atitikmuo	Asociacija su rolėmis ir objektų savybių stereotipu.
OWL atitikmuo	Declaration(ObjectProperty(*IRI*))

2.5 lentelė PA 5 aprašymo lentelė

ID	PA 5
Pavadinimas	Transformuoti objektų savybių aksiomas
Aprašymas	Įskiepis transformuoja specializuojančiu asociacijų ryšius į specializuojančias objektų savybes. Atvirkštines asociacijų ryšių roles į atvirkštines objektų savybes. Kardinalumus į funkcines objektų savybes. Asociacijų ryšių roles į objektų savybių domenus (angl. *Domain*) ir sritis (angl. *Range*).
Aktoriai	Naudotojas
UML atitikmuo	Asociacijos ryšio rolės poaibis. Atvirkštinė asociacijos ryšio rolė. Asociacijos ryšio rolė su „<<Functional>> “ stereotipu.
OWL atitikmuo	SubObjectPropertyOf(*IRI1 IRI2*) InverseObjectProperties(*IRI1 IRI2*) FunctionalObjectProperty(*IRI*) ObjectPropertyDomain(*IRI1 IRI2*) ObjectPropertyRange(*IRI1 IRI2*)

2.6 lentelė PA 6 aprašymo lentelė

ID	PA 6
Pavadinimas	Transformuoti duomenų savybes
Aprašymas	Įskiepis transformuoja *UML* klasių diagrama klasėse aprašytus atributus, kurie turi stereotipą „<<DataProperty>> “ į duomenų savybes *OWL 2* ontologijų kalboje.
Aktoriai	Naudotojas
UML atitikmuo	Klasės atributas su stereotipu ir IRI.
OWL atitikmuo	Declaration(DataProperty(*IRI*))

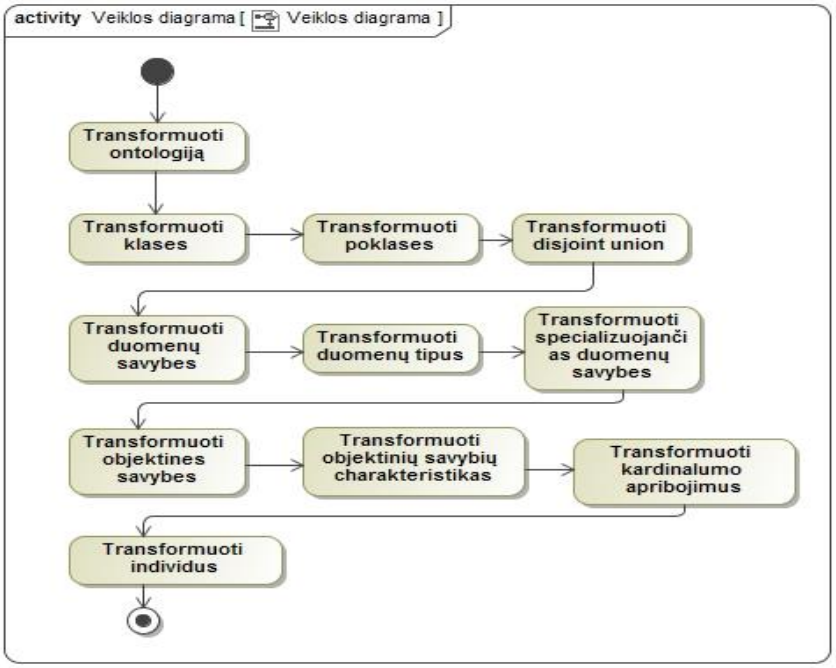
2.7 lentelė PA 7 aprašymo lentelė

ID	PA 7
Pavadinimas	Transformuoti duomenų savybių aksiomas
Aprašymas	Įskiepis transformuoja UML klasių diagrama atvaizduotus duomenų tipus į duomenų savybių sritis (angl. *Range*) ontologijų kalboje. Atributų klases į duomenų savybių domenus (angl. *Domain*). Paveldinčius atributus i paveldinčias duomenų savybes ontologijų kalboje.
Aktoriai	Naudotojas
UML atitikmuo	Atributo klasė. Klasės atributo tipas. Klasės atributo ryšys su stereotipu. Duomenų tipas.
OWL atitikmuo	DataPropertyDomain(*IRI1 IRI2*) DataPropertyRange(*IRI* xsd:DataType) SubDataPropertyOf(*IRI1 IRI2*) Declaration(Datatype(xsd:DataType))

2.8 lentelė PA 8 aprašymo lentelė

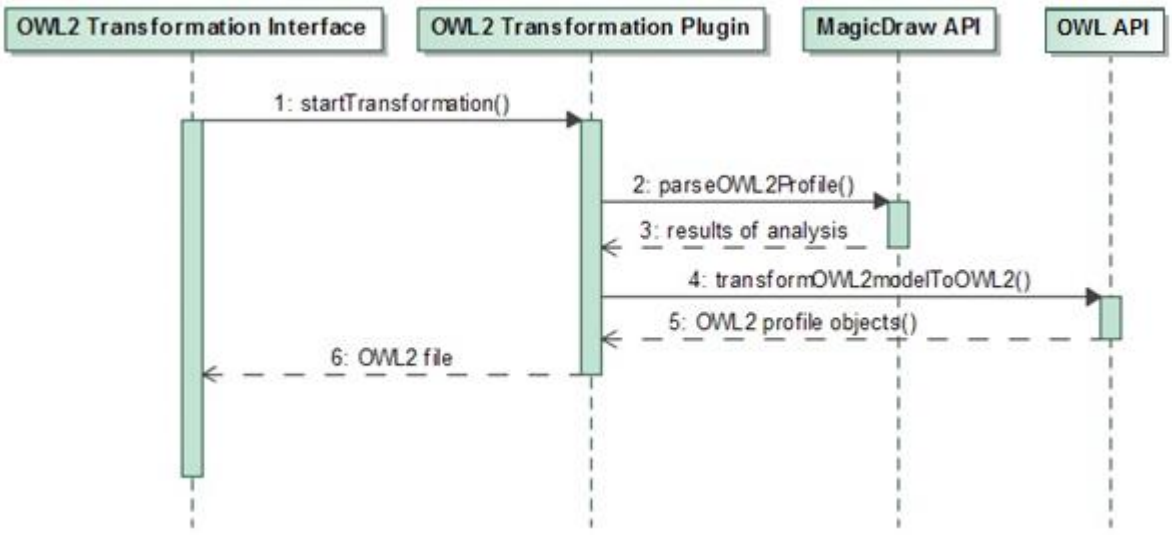
ID	PA 8
Pavadinimas	Transformuoti individus
Aprašymas	Įskiepis transformuoja egzempliorius į individus *OWL* ontologijų kalboje.
Aktoriai	Naudotojas
UML atitikmuo	Egzempliorius su stereotipu ir aprašytomis individo reikšmėmis.
OWL atitikmuo	Declaration(NamedIndividual(IRI))

Realizuoti sprendimui buvo išsikelti reikalavimai naudojant sukurtą profilį transformuoti pagrindinius ontologijos elementus, stengiantis kiek įmanoma neprarasti duomenų. Ontologijos pavaizduotos *UML* naudojant sukurtą *OWL* profilį transformacijos procesas į OWL 2 kalbą pavaizduotas 2.2 pav.



2.2 pav. *OWL2* profilio transformavimo į *OWL2* procesas pavaizduotas *UML* veiklos diagrama

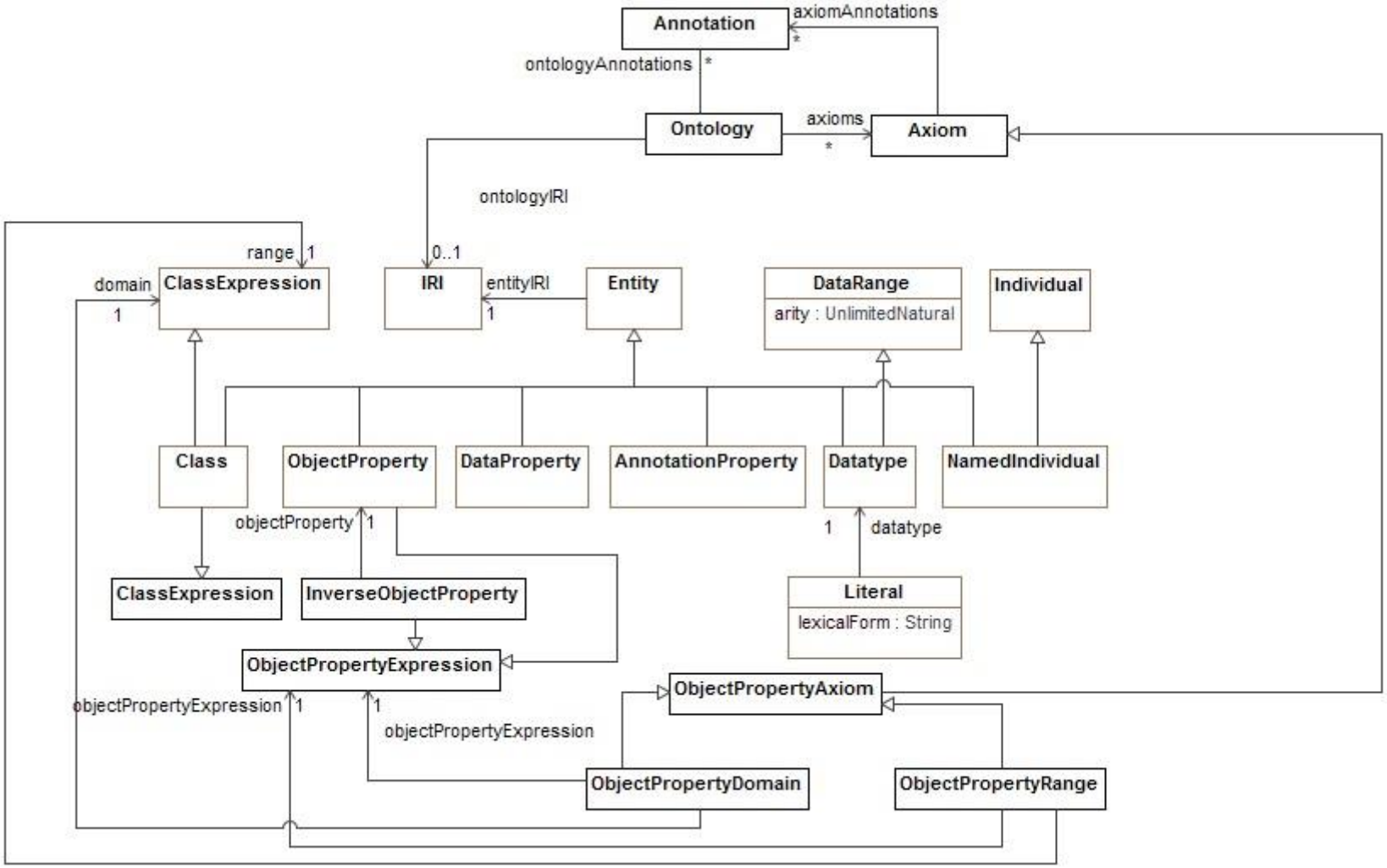
Apibendrintas įskiepio veikimo principas pavaizduotas sekų diagrama 2.3 paveikslėlyje. Aktyvavus įskiepį „MagicDraw open API“ pagalba analizuojama ontologija pavaizduota *UML* naudojant *OWL2* profilį. Surinkti duomenys palaipsniui perduodami į *OWL API* ir galiausiai sugeneruojamas failas, kuriame išsaugoma transformuota ontologija į *OWL 2* ontologijų kalbą.



2.3 pav. *OWL 2* transformacijos įskiepio sekų diagrama

2.2. Dalykinės srities modelis

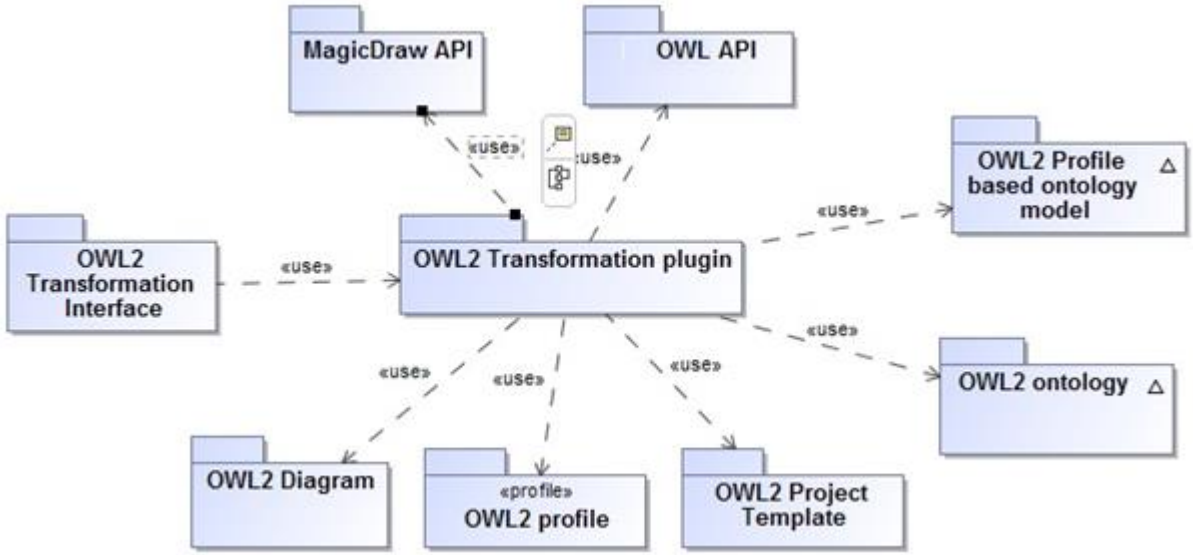
Žemiau esančiame 2.4 pav. pateikiami pagrindiniai *OWL 2* meta modelio elementai. Kurtas įskiepis padengia beveik visus elementus, tai yra pagrindiniai elementai, kuriuos turint galima plėsti ontologiją toliau.



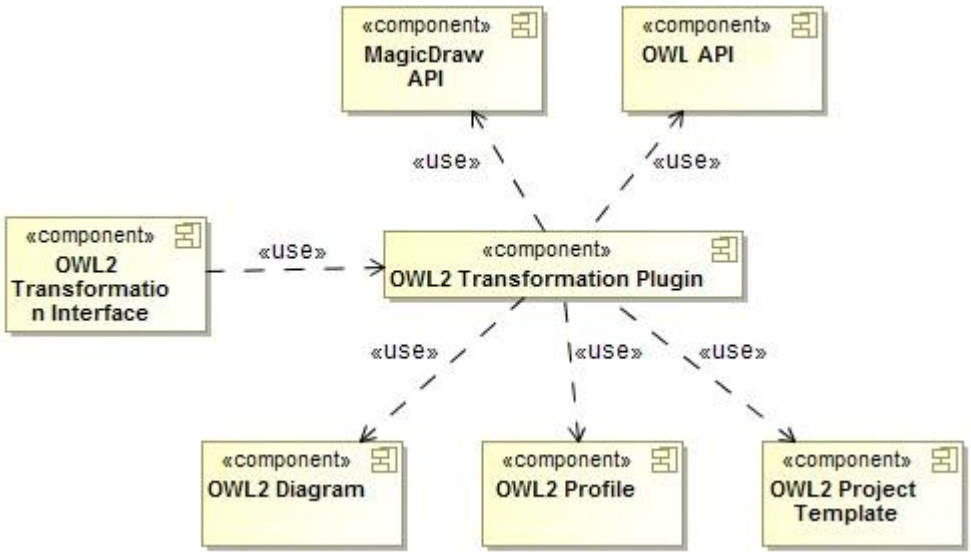
2.24 pav. Pagrindiniai *OWL2* meta modelio elementai

2.3. Ontologijų transformavimo iš UML į OWL 2 projektas

Žemiau esančiuose paveikslėliuose 2.5 pav. 2.6 pav. matoma kokie komponentai reikalingi įskiepio naudojimui. Ontologijos pavaizduotos *UML* nuskaitymui naudojamas „MagicDraw API“. Ontologijos Modeliavimui *UML* naudojamas *OWL 2* profilis. Transformacijai į *OWL 2* ontologijų kalba naudojamas „OWL API“.

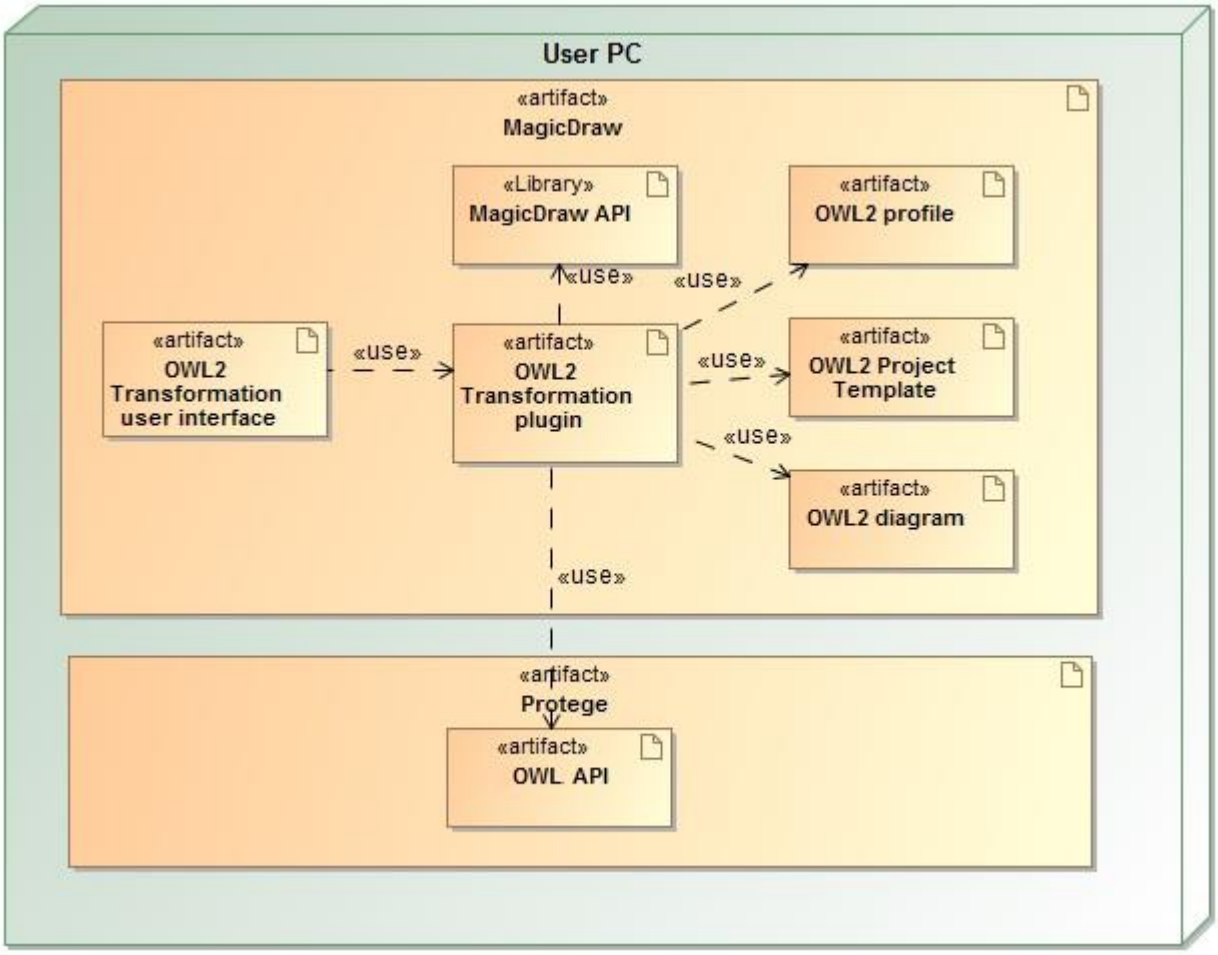


2.35 pav. *OWL2* transformacijos įskiepio loginė architektūra



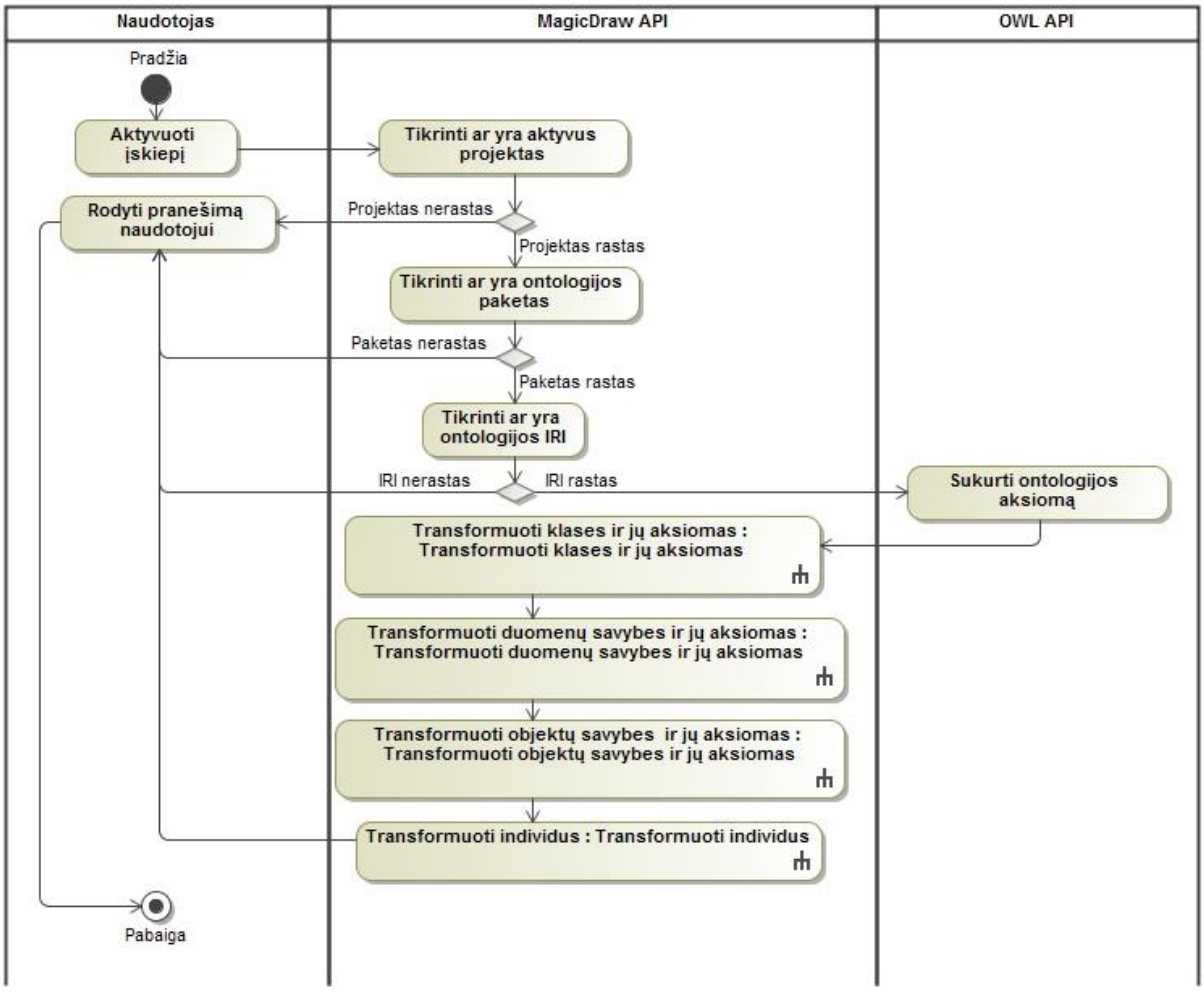
2.36 pav. *OWL2* transformacijos įskiepio komponentų diagrama

Transformacijos įskiepio diegimo diagrama pateikta 2.7 pav. Diagramoje atsispindi ryšiai tarp visų naudotų komponentų. Kad funkcionuotų įskiepio paleidimo mygtukas, įrankyje turi būti įdiegtas darbo metu sukurtas įskiepis. Įskiepis diagramos elementų surinkimui ir analizavimui naudoja „MagicDraw API“. *OWL 2* profilis naudojamas saugoti stereotipams, kurie yra reikalingi norint transformuoti ontologiją pavaizduotą *UML* klasių diagrama į *OWL 2* ontologijų kalbą. Projekto šablonas naudojamas tam, kad būtų patogiau kurti ontologijas *UML* pagrindu. Šablonas turi paruoštus elementus, kurie iškarto turi nustatytas tam tikras savybes, kurios yra būdingos atitinkamiems elementams. Galiausiai visą surinktą informaciją įskiepis perduoda *OWL API*, kuris sukuria ontologiją ir ją išsaugo tekstiniame faile „.owl“ formatu.



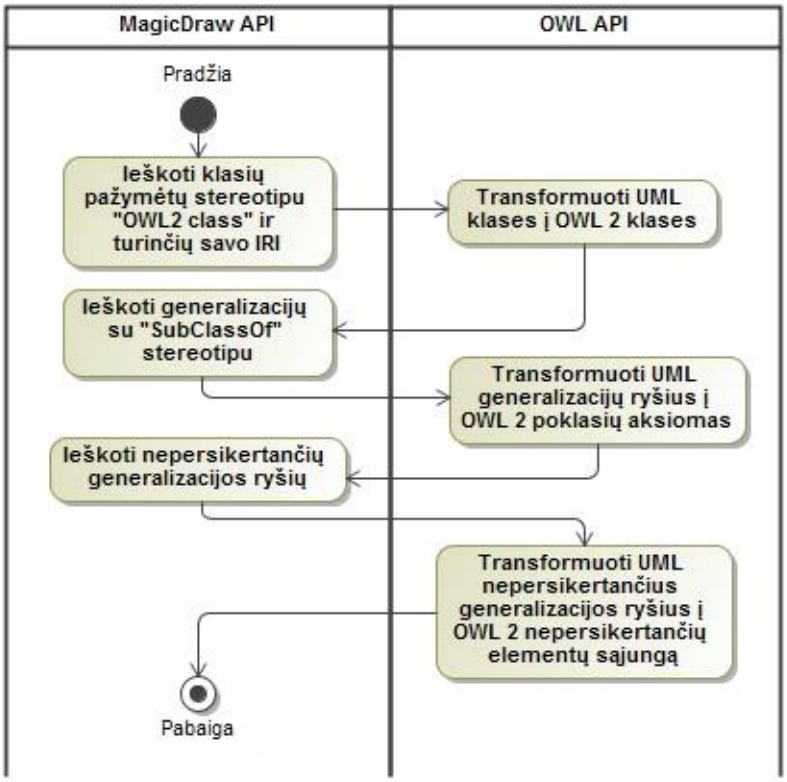
2.37 pav. *OWL2* transformacijos įskiepio diegimo diagrama

Paveikslėliuose 2.8 - 2.12 pateikiamos detalizuotos veiklos diagramos. Pirmasis paveikslėlis parodo bendrą viso įskiepio veikimo principą nedetalizuojant klasių, objektų savybių, duomenų savybių ir individų veikimo principų. Kiekvienas paminėtas konceptas pavaizduotas atskiroje veiklos diagramoje. Taigi, kaip galima matyti 2.8 pav., įskiepis yra aktyvuojamas naudotojo, aktyvavus įskiepį pirma patikrinama ar įrankyje yra atidarytas projektas. Jei nėra vartotojas gauna pranešimą nurodantį, kad norint naudotis įskiepiu reikalingas projektas. Egzistuojant projektui įskiepis pirmiausiai transformuoja ontologiją. Šiuo atveju ontologija laikomas paketas, kuris privalo turėti stereotipą iš darbo metu kurto *OWL* profilio, kuris nurodo, kad paketas transformuosis į *OWL 2* ontologijos aksiomą. Paketas taip pat privalo turėti savo unikalų *IRI*.



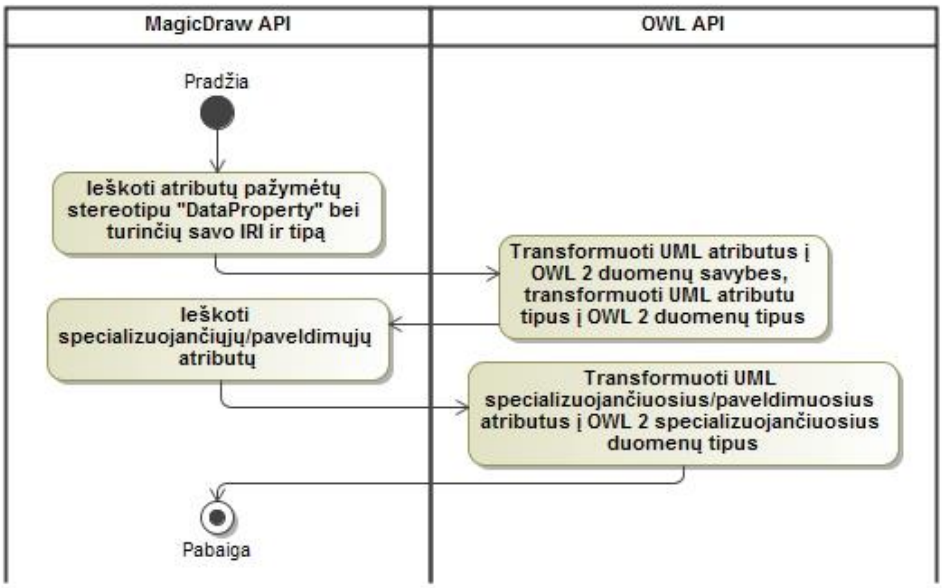
2.8 pav. Detalizuota bendra veiklos diagrama

Žemiau esančiame 2.9 pav. pateikiama detalizuota klasių ir jos aksiomų transformacijos veiklos diagrama. Ji atspindi šios dalies transformacijos procesą, kuriame pirma yra transformuojamos klasės, tada poklasės ir galiausiai nepersikertančių elementų sąjunga.



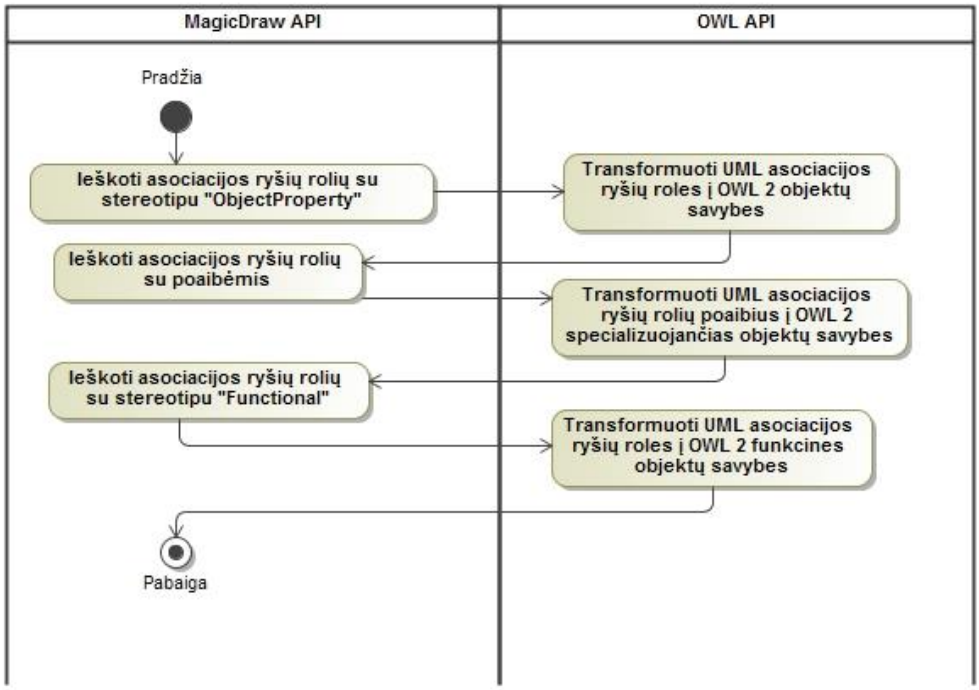
2.9 pav. Detalizuota klasių veiklos diagrama

Žemiau esančiame 2.10 pav. pateikiama detalizuota duomenų savybių ir jų aksiomų transformacijos veiklos diagrama. Ji atspindi šios dalies transformacijos procesą, kuriame pirma yra transformuojamos duomenų savybės ir duomenų tipai, tada transformuojamos specializuojančios duomenų savybės (angl. *Sub Data Property Of*).



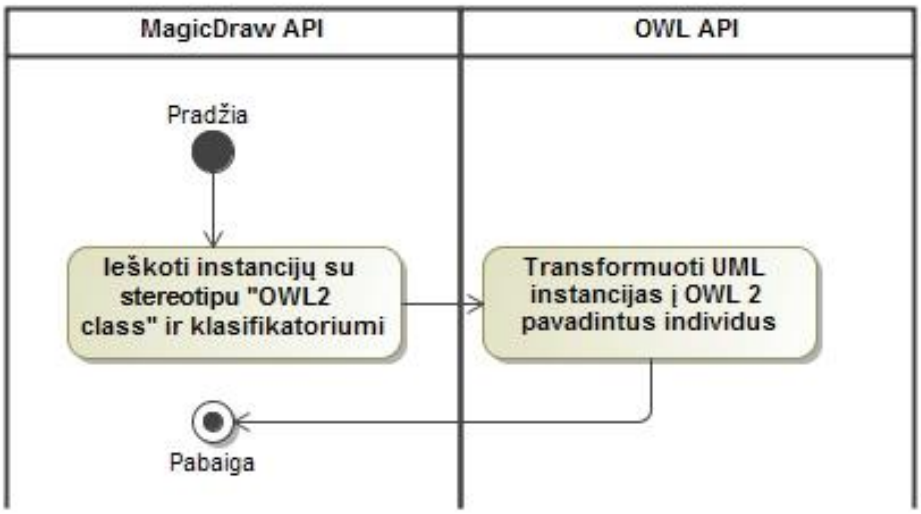
2.10 pav. Detalizuota duomenų savybių veiklos diagrama

Žemiau esančiame 2.11 pav. pateikiama detalizuota objektų savybių ir jų aksiomų transformacijos veiklos diagrama. Ji atspindi šios dalies transformacijos procesą, kuriame pirma yra transformuojamos objektų savybės, tada specializuojančios objektų savybės (angl. *Sub Object Property Of*) ir užbaigiama transformuojant funkcines objektų savybes.



2.11 pav. Detalizuota objektų savybių veiklos diagrama

Žemiau esančiame 2.12 pav. pateikiama detalizuota individų transformacijos veiklos diagrama. Ji atspindi šios dalies transformacijos procesą, kuriame yra transformuojami individai (angl. *Named Individuals*).



2.12 pav. Detalizuota individų veiklos diagrama

2.4. Formalus sprendimo aprašas

Šiame poskyryje aprašomos taisyklės, kuriomis buvo vadovautasi kuriant įskiepį. Jos atspindi kas į ką transformuojasi. Šiuo atveju kalbama apie transformacijas atliekamas iš *UML* klasių diagrama pavaizduotos ontologijos į *OWL 2* ontologijų kalbą. Taisyklių sąrašas:

- kiekvieną *UML* klasę su stereotipu „OWL2 class“ reikia transformuoti į *OWL 2* klasę;
- kiekvieną *UML* generalizacijos ryšių rinkinį nustatytą kaip nepersikertantį reikia transformuoti į *OWL 2* nepersikertančių elementų sąjungos (angl. *Disjoint Union*) aksiomą;
- kiekvieną *UML* klasę kuri turi generalizacijos ryšį su stereotipu „SubClassOf“ į kitą klasę transformuoti į *OWL 2* poklasės aksiomą;
- kiekvieną *UML* asociacijos ryšio rolę su stereotipu „ObjectProperty“ transformuoti į *OWL 2* objektų savybę;
- kiekvieną *UML* asociacijos ryšio rolės poaibį transformuoti į *OWL 2* specializuojančią objektų savybės aksiomą (angl. *Sub Object Property Of*);
- kiekvieną *UML* asociacijos ryšį su rolėmis abiejuose galuose, kurios turi stereotipus „ObjectProperty“ transformuoti į *OWL 2* atvirkštinės objektų savybės aksiomą;
- kiekvieną *UML* asociacijos ryšio rolę su stereotipu „ObjectProperty“ ir stereotipu „Functional“ transformuoti į *OWL 2* funkcinės objektų savybės aksiomą;
- kiekvienos *UML* asociacijos ryšio rolės su stereotipu „ObjectProperty“ charakteristikas transformuoti į *OWL 2* objekto savybės domeno (angl. *Object Property Domain*) aksiomą ir objekto savybės srities (angl. *Object Property Range*) aksiomą;
- kiekvieną *UML* klasės atributą su stereotipu „DataProperty“ transformuoti į *OWL 2* duomenų savybę;
- kiekvieno *UML* klasės atributo su stereotipu „DataProperty“ charakteristikas transformuoti į *OWL 2* duomenų savybės domeno (angl. *Data Property Domain*) aksiomą ir duomenų savybės srities (angl. *Data Property Range*) aksiomą;
- kiekvieną *UML* klasės atributo su stereotipu „DataProperty“ poaibį transformuoti į *OWL 2* specializuojančią duomenų savybės (angl. *Sub Data Property Of*) aksiomą;
- kiekvieną *UML* egzempliorių (angl. *Instance*) transformuoti į *OWL 2* įvardintą individą (angl. *Named Individual*).

2.5. Reikalavimų apibendrinimas

Projektui keliami reikalavimai yra sukurti įskiepį „MagicDraw“ įrankiui, kuris suteiktų galimybę transformuoti *UML* pavaizduotą ontologiją į *OWL 2* ontologijų kalbą. Transformacija turi padengti pagrindinius *OWL Lite* elementus:

- klases;
- objektų savybes;
- duomenų savybes;
- individus;
- šių konceptų aksiomas.

Įskiepis atlikęs transformaciją turi išsaugoti ontologiją i failą „.owl“ formatu. Žemiau pateikiama apibendrinanti lentelė, kurioje pateikiamos taisyklės, kaip asocijuojasi *OWL 2* elementai ir *UML* elementai.

2.9 lentelė *OWL 2* profilio vaizdavimo ir transformavimo į *OWL 2* taisyklės

	OWL 2 Entity	**OWL2 profilyje šiame darbe**
1.	Annotation Property	
2.	Annotation Property Assertion	
3.	Asymmetric Object Property	
4.	boolean	UML data type Boolean
5.	Class	UML class
6.	Class Assertion (the type of the Individual)	UML Classifier
7.	Data All Values From	UML data type
8.	Data Complement Of	
9.	Data Exact Cardinality	UML attribute multiplicity
10.	Data Intersection Of	
11.	Data Max Cardinality	UML attribute multiplicity
12.	Data Min Cardinality	UML attribute multiplicity
13.	Data One Of (enumeration of Individuals)	
14.	Data Property	UML attribute (property)
15.	Data Property Assertion	
16.	Data Property Domain	UML class in attribute’s specification
17.	Data Property Range	Type in attribute’s specification
18.	Data Type Restriction	
19.	Data Union Of	Is derived union =truein attribute’s specification
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, GeneralizationSet with disjoint subclasses
24.	Disjoint Data Properties	
25.	Disjoint Object Properties	UML dependency with stereotype
26.	Disjoint Union	Is Disjoint=true, Is Covering=true in Specification of Generalization Set
27.	Entity IRI	Tagged value
28.	Equivalent Classes	UML dependency with stereotype

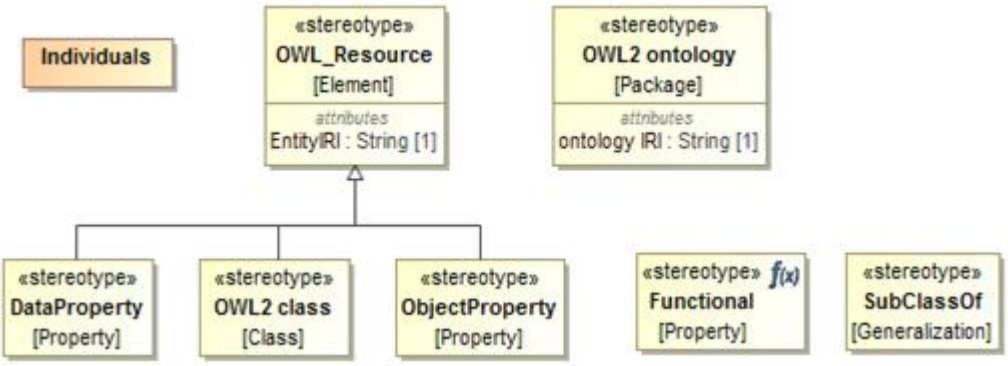
	OWL 2 Entity	OWL2 profilyje šiame darbe
29.	Equivalent Data Properties	
30.	Equivalent Object Properties	
31.	Functional Object Property	UML association constraint
32.	Functional Data Property	UML attribute constraint
33.	Has Key	UML stereotype
34.	Imported ontology	
35.	Inverse Functional Object Property	
36.	Inverse Object Property	Opposite property
37.	Irreflexive Object Property	Property constraint
38.	Language for Annotations	
39.	Named Individual	Object
40.	Negative Data Property Assertion	
41.	Negative Object Property Assertion	
42.	Object All Values From	
43.	Object Complement Of	
44.	Object Exact Cardinality	UML multiplicity of Association End
45.	Object Has Self	
46.	Object Intersection Of	
47.	Object Max Cardinality	UML multiplicity of Association End
48.	Object Min Cardinality	UML multiplicity of Association End
49.	Object One Of	
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	
55.	Object Union Of	
56.	Ontology	UML package
57.	Reflexive Object Property	Property constraint
58.	Same Individuals	UML dependency
59.	string	UMLdatatype
60.	Sub Class Of	UML class specialization
61.	Sub Data Property Of	Subsetted property in attribute specification
62.	Sub Object Property Of	Subsetted property in property specification
63.	Symmetric Object Property	Property constraint
64.	Transitive Object Property	Property constraint
65.	xsd:datetime	UMLdatatype
66.	xsd:integer	UMLdatatype
67.	xsd:nonnegative integer	UMLdatatype
68.	xsd:positive integer	UMLdatatype

3. OWL2 TRANSFORMACIJOS ĮSKIEPIO SPRENDIMO REALIZACIJA IR TESTAVIMAS

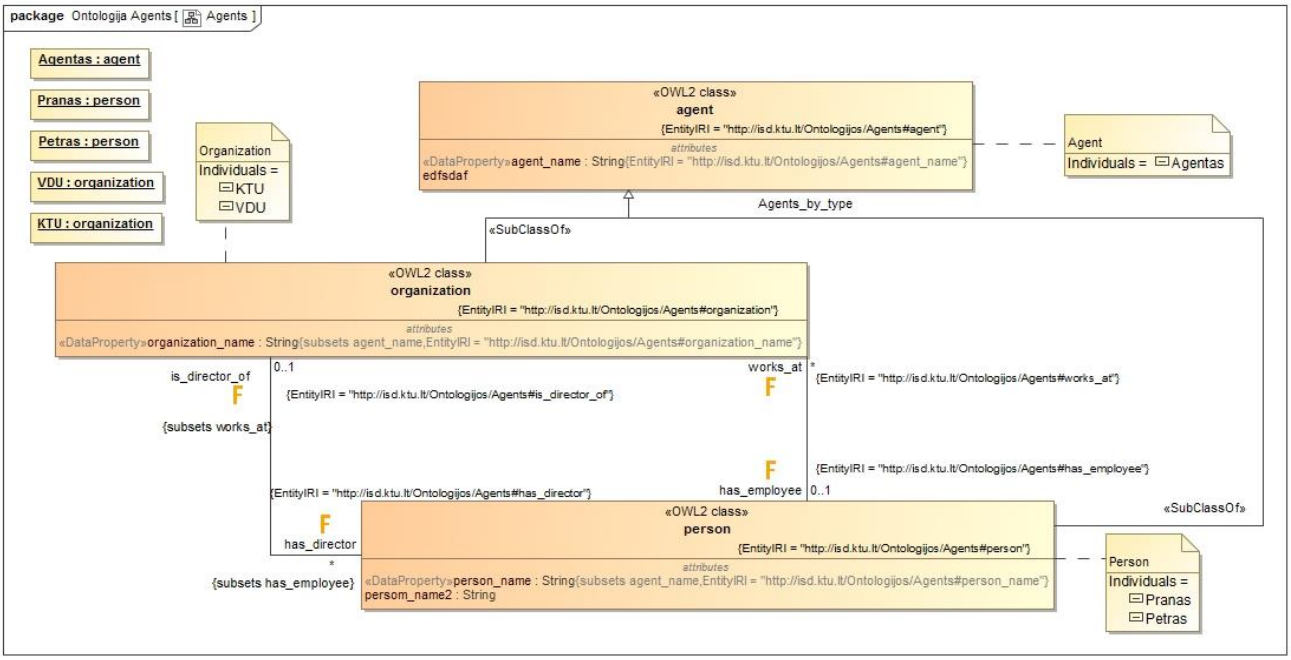
3.1. Sprendimo realizacijos ir veikimo aprašas

Realizacijos etapo metu buvo sukurtas „MagicDraw“ įrankio įskiepis, kuris iš *UML* pagrindu, *OWL* profiliu pavaizduotos ontologijos atlieka transformaciją į *OWL 2* kalbą. Transformuota ontologija išsaugoma „*.owl*“ formato faile, kurį galima atsidaryti naudojant ontologijų kūrimo ar skaitymo įrankį, tokį kaip „Protégé“.

Žemiau esančiuose paveikslėliuose pateikiami darbo metu sukurti OWL 2 profilis (3.1 pav.) ir ontologijos pavyzdys (3.2 pav.) atvaizduojamas UML pagrindu „MagicDraw“ įrankyje naudojant darbo metu sukurtą *OWL* profilį.

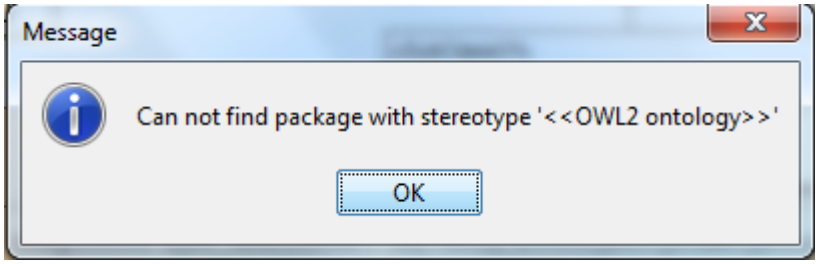


3.1 pav. Darbo metu sukurtas *OWL 2* profilis.



3.2 pav. UML atvaizduotos ontologijos pavyzdys naudojant *OWL2* profilį

Sukurto įskiepio aktyvavimui „MagicDraw“ įrankyje, „file“ meniu juostoje integruotas paleidimo mygtukas. Baigus darbus su ontologiją ir paspaudus minėtą mygtuką „*UmlToOwl*“ pradedama vykdyti transformacija. Įskiepis transformacijai aktyviame projekte ieško paketų kurie turi stereotipą „<<OWL2 ontology>>“, visus kitus paketus tiesiog ignoruoja. Neradęs paketo su minėtu stereotipu įskiepis perspėja vartotoją (3.3 pav.), kad nepavyko rasti ontologijos paketo.

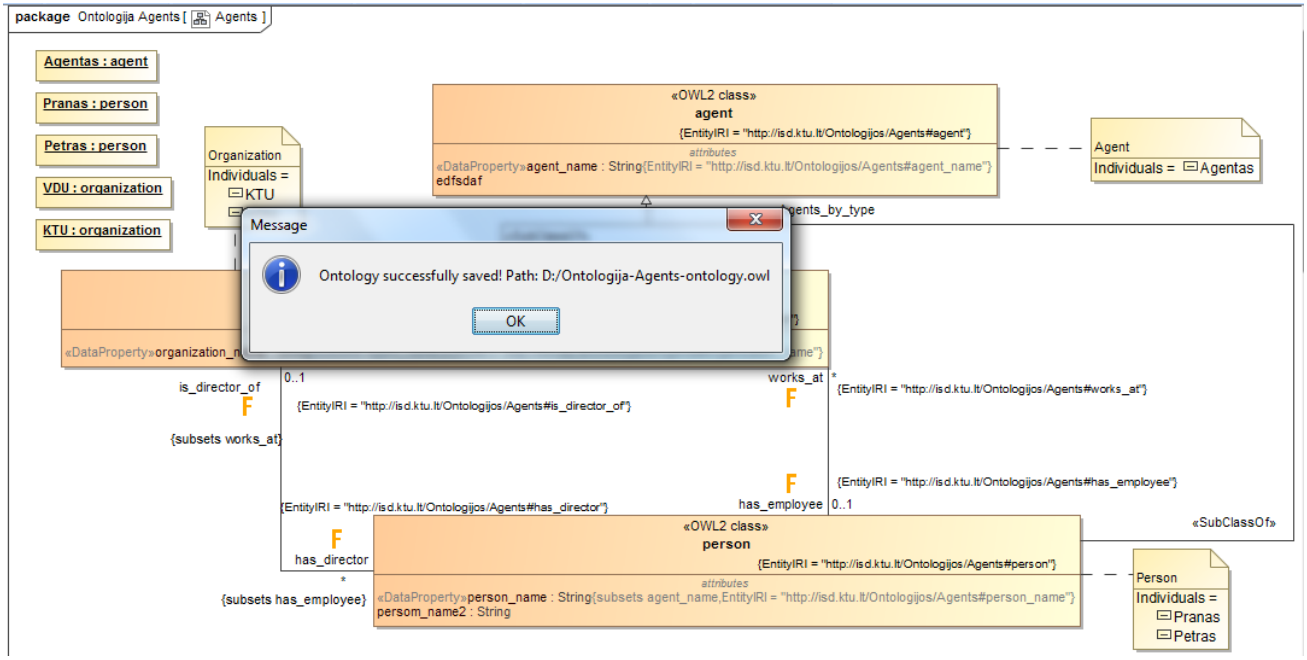


3.3 pav. Įskiepio pranešimas vartotojui

Radus atitinkamą paketą įskiepis analizuoja visus elementus pakete, remdamasis darbo metu sukurto *OWL 2* profilio stereotipais ir pagal aprašo 3 lentelėje esančias taisykles atlieka ontologijos transformaciją į *OWL 2* kalbą.

Įskiepis UML pagrindu atvaizduotą ontologiją analizuoja naudodamas „*MagicDraw API*“, atitinkamai surinktus duomenis perduoda *OWL API*, kurio pagalba užbaigiama transformacija ir ontologija išsaugoma *OWL 2* kalboje „*owl*“ formatu.

Kuriant įskiepį buvo orientuotasi į Windows operacinę sistemą, todėl sėkmės atveju įskiepis išsaugo ontologiją „D:/“ kataloge suformuodamas failo pavadinimą sekančiai: ontologijos-paketo-pavadinimas-ontology.owl. Ontologijos paketo pavadinime esantys tarpai pakeičiami „-“ simboliais. Po atliktos transformacijos vartotojui parodomas pranešimas (3.4 pav.) apie sėkmingai atliktą operaciją ir kelias iki išsaugoto failo.



3.4 pav. *OWL2* profilis pavaizduotas *UML* ir plugino aktyvavimas

Įskiepyje numatyta (nerealizuota) funkcija leidžianti pasirinkti formatą („FunctionalSyntax“, „ManchesterSyntax“, „OWLXML“), kuriuo turėtų būti išsaugota transformuota ontologija. Žemiau pateikiamas transformuotos ontologijos pavyzdys *OWL 2* kalboje „FunctionalSyntax“ formatu.

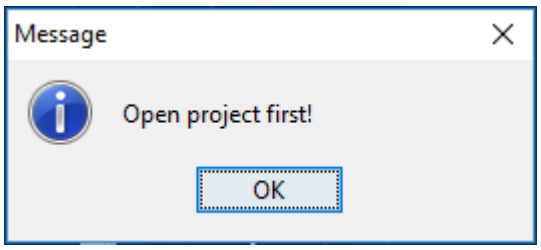
```
Prefix(:=<http://isd.ktu.lt/Ontologijos/Agents#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)

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 Properties
############################

# 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 Properties
############################

# 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)



############################
#   Classes
############################

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

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

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


############################
#   Named Individuals
############################

# 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)
)
```

3.2. Įskiepio testavimas

Iš 3.1 skyriuje aprašyto pavyzdžio galima daryti išvadą, jog tinkamai paruošus aplinką įskiepis veikia kaip numatyta. Testavimo metu buvo išbandyta kaip elgiasi įskiepis esant kokiems nors nesklandumams. Žemiau aprašomi visi bandymo scenarijai.

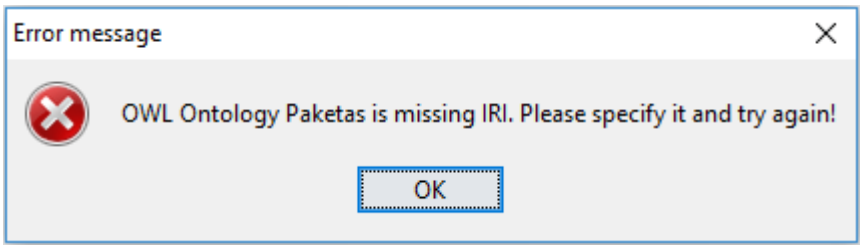
Pirmas bandymas buvo aktyvuoti įskiepi kai nėra atidarytas joks projektas. Įskiepis ima ontologijas transformacijoms tik iš aktyvaus projekto, taigi šiuo atveju neradęs atidaryto projekto įskiepis parodė informacinį pranešimą, kad pirma reikia sukurti naują arba atidaryti jau egzistuojantį projektą. Pranešimo pavyzdys pateikiamas 3.5 pav.



3.5 pav. Įskiepio pranešimas vartotojui

Atsižvelgus į įskiepio pranešimą buvo sukurtas naujas projektas pavadinimu „Testavimas“ ir pabandyta aktyvuoti įskiepį dar kartą. Įskiepis aktyviame projekte pirmiausia ieško paketo kuris turi stereotipą „<<OWL2 ontology>> “, taigi šiuo atveju paketas nebuvo rastas, todėl įrankis parodė dar vieną informacinį pranešimą su tekstu - „Can’t find package with stereotype `<<OWL2 ontology>>`“.

Pranešimas nurodo, kad nepavyko rasti paketo su minėtu stereotipu. Atsižvelgus į tai buvo sukurtas naujas paketas pavadinimu „Paketas“. Jam uždėtas ontologijos stereotipas iš darbo metu sukurto *OWL* profilio. Įskiepis buvo aktyvuotas dar kartą ir buvo gautas kitas pranešimas, šį kartą jau ne informacinis, o klaidos (angl. *Error*) tipo pranešimas kuris parodytas 3.6 pav.



3.6 pav. Įskiepio pranešimas vartotojui

OWL kalboje ontologija turi turėti savo identifikatorių - *IRI*. Šiuo atveju ontologijos paketas neturėjo įvesto *IRI*, todėl įskiepis apie tai pranešė vartotojui ir sustabdė tolimesnį savo darbą. Papildžius ontologijos paketą IRI adresu, šiuo atveju „http://TestIRI“ įskiepis sėkmingai sugeneravo ontologijos failą „.owl“ formatu, kuriame buvo deklaruota tuščia ontologija. Sugeneruotos ontologijos pavyzdys pateikiamas 3.7 paveikslėlyje.

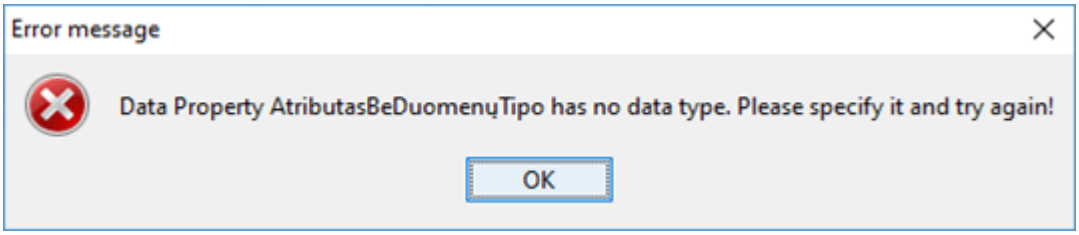
```
Prefix(:=<http://TestIRI#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)


Ontology(<http://TestIRI>

)
```

3.7 pav. OWL kalba deklaruotos ontologijos pavyzdys

Analogiškai kaip ir tuo atveju kai paketas neturėjo *IRI*, įskiepis pasielgtų ir tuo atveju, jei nurodyto *IRI* neturėtu ir klasė, objektų savybė ar duomenų savybė. Tik kiekviename pranešime būtų nurodyta konkrečiai, kuriam elementui dar reikia pridėti *IRI*.

Taip pat buvo ištestuotos ir objektų savybių charakteristikos. Kaip jau buvo minėta analizės dalyje, šis sprendimas išsiskiria nuo kai kurių kitų ontologijų vizualizavimo sprendimų, tokių kaip „Protégé“ tuo, kad atvirkštinėms objektų savybėms nekuria atskirų asociacijų, bet atvaizduoja objektų savybes kaip roles ant abiejų asociacijos galų. Testavimo metu buvo ištestuoti atvejai ne tik kai nurodomos abi rolės, kas leidžia sudaryti atvirkštinę objektų savybę, bet ir atvejis kai nurodoma tik viena rolė. Tokiu atveju įskiepis sukuria tik vieną nurodyta objekto savybę.

Testuojant duomenų savybių transformaciją buvo pastebėta, kad nenurodžius atributo tipo, „MagicDraw“ įrankis pranešą apie įskiepio kūrimo metu paliktą spragą. Dėl šios priežasties tokiu momentu sustodavo ir įskiepio darbas. Šis „MagicDraw“ pranešimas paprastam vartotojui būtų sunkiai suprantamas, nes jis nurodo įskiepio kodo spragas, bet ne problemą su kuria susiduria vartotojas. Tačiau spraga buvo ištaisyta ir to pasėkoje tokiu atveju kai būna nenurodytas atributo tipas vartotojui yra rodomas konkrečios klaidos pranešimas. Sutvarkytos spragos klaidos pranešimo pavyzdys pateikiamas 3.8 pav.



3.8 pav. Įskiepio pranešimas vartotojui

3.3. Testavimo rezultatai

Iš pavyzdžių pateiktų 3.1 skyriuje galima matyti jog atlikus transformaciją yra realizuojami visi apsibrėžti konceptai:

- ontologijos paketas;
- klasių deklaracijos;
  - nesusikertanti sąjunga (angl. *Disjoint Union*);
  - poklasės (angl. *Sub Class Of*);
- objektų savybių deklaracijos;
  - sub objektų savybės (angl. *Sub Object Property Of*);
  - atvirkštinės objektų savybės (angl. *Inverse Object Property Of*);
  - funkcinės objektų savybės (angl. *Functional Object Property*);
  - objektų savybių domenai (angl. *Object Property Domain*);
  - objektų savybių sritys (angl. *Object Property Range*);
- duomenų savybių deklaracijos;
  - duomenų savybių domenai (angl. *Data Property Domain*);
  - duomenų savybių sritys (angl. *Data Property Range*);
  - sub duomenų savybės (angl. *Sub Data Property Of*);
- individualų (angl. *Named Individual*) deklaracijos;
  - klasių teiginiai (angl. *Class Assertion*);
- duomenų tipų deklaracijos.

Taip pat testavimo metu buvo ieškoma ar nėra paliktų spragų kurios trikdytų įskiepio darbą. Visuose testavimo atvejuose išskyrus viena įskiepis sėkmingai pranešdavo vartotojui apie kylančias problemas kurias reikia išspręsti kuriant ontologiją *UML* pagrindu. Po testavimo buvo sutvarkyta spraga įskiepio kode, dėl kuriuos įskiepis nenumatytai nustodavo veikti.

4. EKSPERIMENTINIS OWL 2 TRANSFORMACIJOS ĮSKIEPIO TYRIMAS

4.1. Eksperimento tikslas

Šio eksperimento tikslas yra susikurti ontologiją „MagicDraw“ įrankyje, kuri turėtų padengti visus pagrindinius elementus aprašytus analizės dalyje: klases, objektų savybes, duomenų savybes, individus bei jų aksiomas. Sukurtą ontologiją naudojant darbo metu sukurtu įskiepiu transformuoti į *OWL 2* kalbą ir patikrinti ar buvo gauti tokie rezultatai kokių buvo tikimasi. Transformuotą ontologiją patikrinti su kitais įrankiais.

4.2. Eksperimento planas

Eksperimentiniam tyrimui atlikti *UML* pagrindu „MagicDraw“ įrankyje buvo sukurtas dar vienas, kiek didesnės ontologijos pavyzdys.

Kaip matoma iš žemiau pateiktos 4.1 lentelės, atlikus šios ontologijos transformaciją į ontologijų kalbą *OWL 2* nebuvo prarasta jokių konceptų – transformacija suveikė kaip tikėtasi. Taigi eksperimentui buvo pasirinkta išbandyti sugeneruota ontologiją su skirtingais įrankiais. Ontologija buvo sukurta *UML* pagrindu „MagicDraw“ įrankyje ir darbo metu sukurto įskiepio pagalba transformuota į *OWL 2* kalbą. Transformuota ontologija buvo išsaugota „Pirma-ontologija-ontology.owl“ faile, kuris ir buvo naudojamas išbandyti vizualizavimą su kitais įrankiais.

4.1 lentelė transformacijos rezultatai

	Prieš transformaciją	Po transformacijos
Klasės	9	9
Poklasės	8	8
DisjointUnion	3	3
Objektų savybės	4	4
Objektų savybių domenai	4	4
Objektų savybių sritys	4	4
Atvirkštinės objektų savybės	2	2
Specializuojančios objektų savybės	2	2
Funkcinės objektų savybės	1	1
Duomenų savybės	12	12
Duomenų savybių domenai	12	12
Duomenų savybių sritys	12	12
Sub duomenų savybės	1	1
Individualai	2	2
Duomenų tipai	2	2

Eksperimentui sudarytas planas:

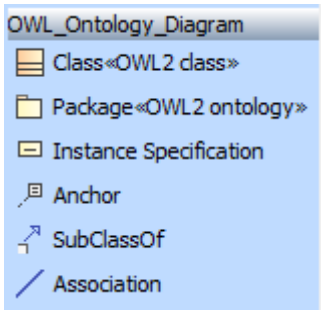
1. Išsikelti vertinimo kriterijus;
2. Išbandyti kelis įrankius su ta pačia ontologija;
3. Pagal išsikeltus vertinimo kriterijus įvertinti panašius sprendimus.

4.3. Eksperimentas

Eksperimento metu sukurta ontologija be „MagicDraw“ buvo išbandyta dar su dviem įrankiais: „Protégé“ ir „OWLGrEd“. Žemiau esančiame skyriuje aprašoma apie kiekvieną atvejį atskirai.

4.3.1. Eksperimentas su „MagicDraw“

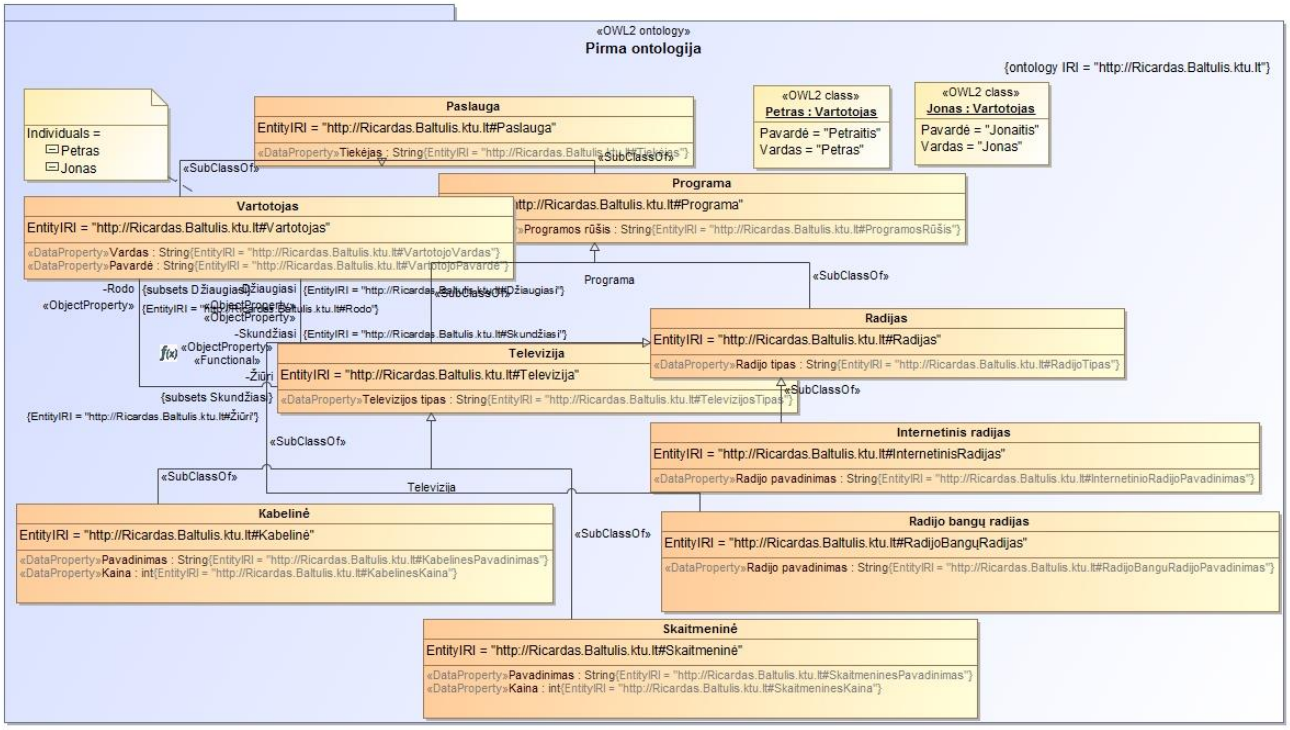
„MagicDraw“ įrankis turi labai platų grafinį konfigūravimą. Tai leidžia vartotojui susikonfigūruoti ontologijų atvaizdavimą labai įvairiai. Viena iš galimybių sukurti atskirą diagramą, kuri turėtų jau paruoštą šabloną ir jo pagalba būtų labai supaprastintas ontologijų kūrimas UML pagrindu. Žemiau esančiame 4.1 pav. matomas pavyzdys kaip atrodo šablonas kurti ontologijoms.



4.1 pav. Įrankiai pritaikyti ontologijai kurti „MagicDraw“ aplinkoje

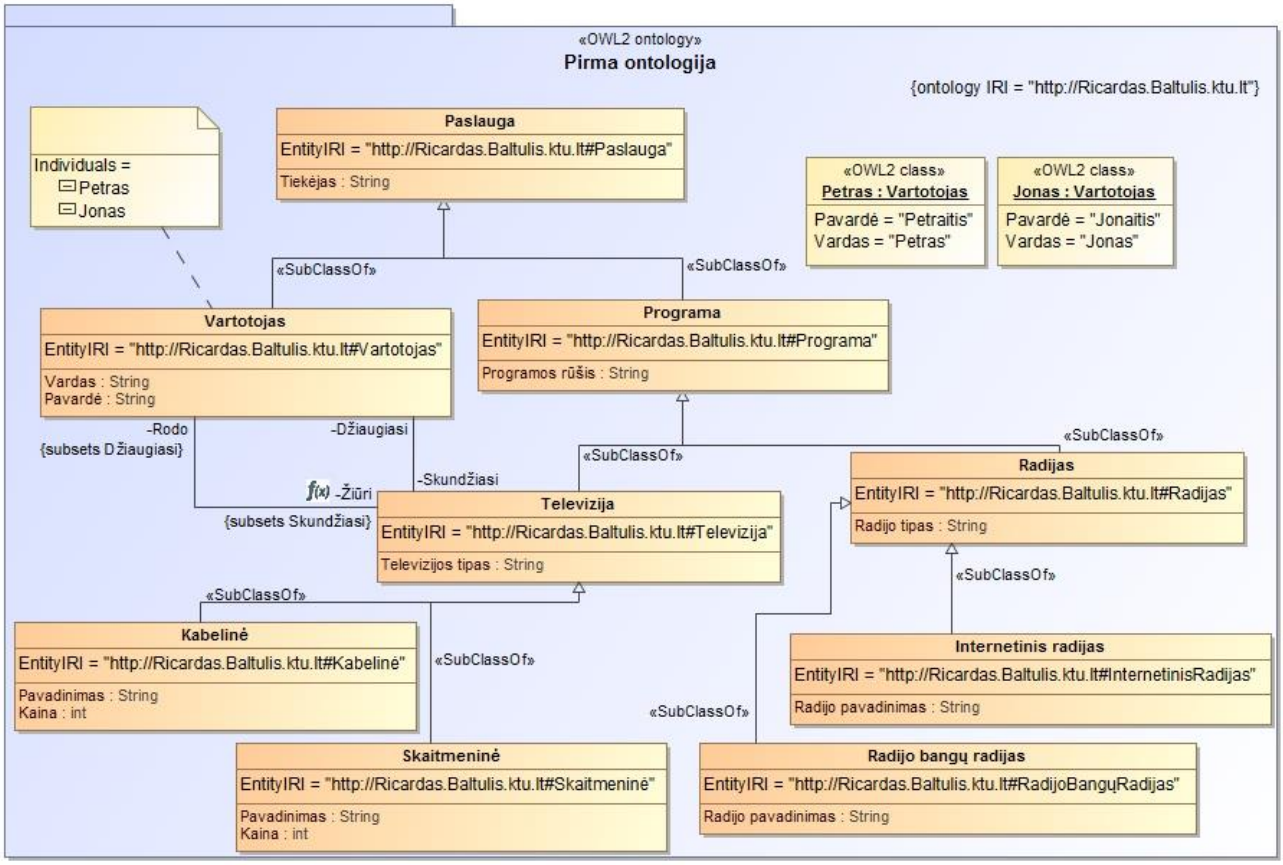
Šio šablono pagalba nereikia rankiniu būdu dėlioti stereotipų, kurie susieja *UML* diagrama su ontologijų kalba *OWL 2* transformacijos metu.

Eksperimentui naudojamam pavyzdžiui sukurti buvo naudojamasi 4.8 pav. pateiktu šablonu. Pirmas sukurtos ontologijos vaizdas neatrodė patogus skaitymui, nes buvo rodomi beveik visi duomenys (4.2 pav.). Šioje ontologijoje atvaizduojama mažiau ryšių, tačiau daugiau teksto. Bet „MagicDraw“ įrankis leidžia pasirinkti ką mes norime atvaizduoti, o ko norime nerodyti.



4.2 pav. „MagicDraw“ vizualizuojama ontologija *UML* pagrindu

Taigi atlikus kelis vizualizacijos pakeitimus vaizdas tapo daug švaresnis. Pavyzdys po pakeitimų matomas 4.3 pav. Matoma vaizdą dar galima tobulinti pagal poreikius išnaudojant plačias „MagicDraw“ teikiamas galimybes.



4.3 pav. „MagicDraw“ vizualizuojama ontologija *UML* pagrindu po pakeitimų

4.4. Eksperimento rezultatai

Eksperimento metu buvo išsikelti kriterijai pagal kuriuos buvo palyginti eksperimento metu naudoti įrankiai. Vertinant buvo atsižvelgta:

- kiek laiko trunka perdėlioti ontologijos elementus, kad pasiekti skaitomesnį vaizdą, kuriame būtų mažiau persikertančių linijų, pasislėpusių elementų, aiškiai matomos hierarchijos;
- ar turi galimybę išsaugoti grafinį ontologijos vaizdą;
- kiek grafinių elementų naudojama objektų savybei vizualizuoti;
- ar leidžia kurti/koreguoti ontologiją grafiškai;
- UML modelių palaikymas.

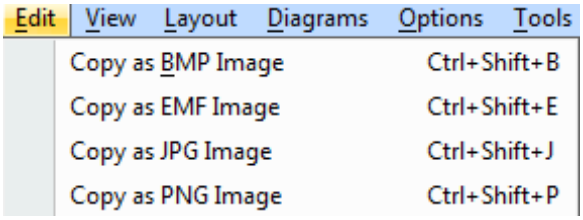
Gauti rezultatai pateikti 4.2 lentelėje.

4.2 lentelė. Palyginimo rezultatai

	MagicDraw	OWLGrEd	Protégé (OntoGraf)
Užtrukta adaptuojant (min)	4	10	3
Leidžia išsaugoti	+	+	-
Grafiniai elementai objektų savybėms	Viena linija	Viena linija	Dvi linijos
Grafinis redagavimas/kūrimas	+	+	-
UML palaikymas	+	-	-

Vertinant kiek laiko užtrunkama adaptuojant ontologiją reikėtų atkreipti dėmesį į tai, kad šis laikas priklauso ir nuo naudotojo įgūdžių su kiekvienu įrankiu. Šiuo atveju rezultatai parodo jog norint adaptuoti ontologiją į kitą vaizdą su didžiausiomis problemomis susiduriama „OWLGrEd“ įrankyje. Nelankstus elementų stumdymas atima labai daug laiko, tad norint pasiekti galutinį rezultatą sugaištama daug laiko. „MagicDraw“ ir „Protégé“ įrankiuose laiko atžvilgiu didelio skirtumo neįžvelgta.

Pasiekus norimą vizualizacijos rezultatą ne visi įrankiai leidžia jį išsisaugoti. Protégé įrankyje norint vėl pamatyti tą pati rezultatą iš naujo įjungus programą tektų vėl perdėlioti visą schemą. Tuo tarpu „MagicDraw“ ir „OWLGrEd“ leidžia išsaugoti atliktus pakeitimus ir sekantį kartą įjungus programą galima tęsti darbą nuo ten kur buvo sustota. „MagicDraw“ įrankis taip pat leidžia išsaugoti visą diagramą arba atskirą jos komponentą kaip paveiksliuką keliais formatais. Tai galima padaryti per meniu juostoje esanti punktą „Edit“, kaip parodyta 4.4 pav.



4.4 pav. Galimybė išsaugoti keliais formatais

„Protégé“ įrankis atvirkštines objektų savybes atvaizduoja atskirais ryšiais. Tai labai apkrauna diagramą ir blogina skaitomumą. Kai tuo tarpu „OWLGrEd“ atvirkštines objektų savybes atvaizduoja ant vieno ryšio skirtingų galų. Ši problema taip pat buvo išspręsta ir „MagicDraw“ įrankyje. Taip dvigubai sumažėja ryšių kiekvienos objektų savybės atžvilgiu.

„MagicDraw“ ir „OWLGrEd“ leidžia ne tik atvaizduoti ontologijas vizualiai, bet ir jas kurti ar redaguoti grafiniu būdu. „Protégé OntoGraf“ to pasiūlyti negali. Ten yra galimybė tik peržiūrėti ontologiją ir dėliotis ją patogiau.

4.5. Sprendimo įvertinimas

Šis sprendimas suteikia galimybė perkelti ontologijas į sistemų kūrimo ir projektavimo pasaulį. Eksperimentas parodė, kad ontologijų perkėlimas į *UML* modelius ir iš jų suteikia lankstumo, leidžia ontologijas panaudoti ir su kitais įrankiais.

Sukurtas sprendimas naudojant pavyzdinę ontologiją veikė kaip numatyta, nebuvo prarasta jokių duomenų ir dėka sukurto įskiepio ontologiją buvo galima išbandyti su kitais ontologijų vaizdavimo įrankiais.

5. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Darbo metu buvo atlikta analizė, po kurios transformacijoms iš *UML*, *OWL* 2 profiliu pavaizduotų ontologijų į *OWL 2* ontologijų kalbą buvo pasirinkta naudoti „MagicDraw API“ ir „OWL API“.
2. Įskiepis apima pagrindinius ontologijos elementus, kurie sudaro pamatą, ant kurio galima viską dėlioti toliau. Šio prototipo užtenka veikimo principui parodyti ir ateityje jį galima tobulinti apimant vis daugiau elementų bei pritaikyti praktikoje, taip palengvinant darbą su ontologijomis.
3. Sukurtas įskiepis atlieka transformacija iš *UML* pavaizduotų ontologijų naudojant darbo metu sukurtą *OWL* profilį į ontologijų kalbą *OWL 2*. Sugeneruotą ontologijos failą galima peržiūrėti su kitais ontologijų redaktoriais.
4. Eksperimentas parodė, kad įskiepis suteikia galimybė įtraukti ontologijų kūrimą į informacinių sistemų projektavimą.

6. LITERATŪRA

1. Natalya F. Noy and Deborah L. McGuinness. 94305 Ontology Development 101: A Guide to Creating Your First Ontology. Stanford University, Stanford, CA. [Kreiptasi 13 04 2016].
2. No Magic, Inc. MagicDraw Architecture Made Simple, Open API 18.1, 2015. Lietuva [Kreiptasi 21 05 2015]
3. OWL Working Group, https://www.w3.org/2001/sw/wiki/OWL [Kreiptasi 23 05 2015]
4. Kirill Fakhroutdinov, UML Profile Diagrams [Tinkle]. Pasiekiama: http://www.uml-diagrams.org/profile-diagrams.html#profile [Kreiptasi 07 03 2015].
5. Andrew D. Spear. Ontology for the Twenty First Century: An Introduction with Recommendations. Germany University at Buffalo, Buffalo, New York, U.S.A. [Kreiptasi 05 05 2016]
6. Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. Stanford Knowledge Systems Laboratory [Kreiptasi 13 04 2016]
7. T. R. Gruber, "A translation approach to portable ontologies. Knowledge Acquisition", 1993
8. T. Berners-Lee, J. Hendler ir O. Lassila, „The Semantic Web," Scientific American, May 2001.
9. „Semantic Web," W3C, 2013. [Tinkle]. Available: http://www.w3.org/standards/semanticweb/. [Kreiptasi 19 05 2016].
10. F. Lindörfer, Semantic Web Frameworks, 2010.
11. F. Ghaleb, S. Daoud, A. Hasna, J. M. ALJa'am, S. A. El-Seoud ir H. El-Sofany, E-Learning Model Based On Semantic Web Technology, 2006.
12. J. Dmitrieva ir F. J. Verbeek, Node-Link and Containment Methods in Ontology Visualization.
13. Librelotto, G.R. Ramalho J. C., Henriques P. R., "TMBuilder: An Ontology Builder based on XML Topic Maps". Clei electronic journal, vol. 7, no. 2, paper 4.