ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Mantas Reminas
Informatikos specialybės II kurso magistrantūros ištęstinio skyriaus studentas

Ontologijų tobulinimo metodų tyrimas

Research on Ontology Evolution Methods

MAGISTRO DARBAS

Darbo vadovė:
Lekt. L. Tankelevičienė

Recenzentas:
Lekt. V. Dumskis

Šiauliai, 2011

„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta naudojant literatūros sąraše pateiktus informacinius šaltinius bei savo tyrimų duomenis“

Darbo autoriaus Mantas Reminas_____

(vardas, pavardė, parašas)

Darbo tikslai ir uždaviniai

Tikslas
Išsiaiškinti ontologijų kūrimo, tobulinimo ir apjungimo būdus bei panaudoti įgytas žinias savarankiškai apjungiant dvi ontologijas į vieną.

Uždaviniai
- Išsiaiškinti kaip yra skirstomos ir vaizduojamos ontologijos;
- Išanalizuoti egzistuojančias ontologijas;
- Išnagrinėti ontologijų kūrimo, tobulinimo principus bei jų apjungimo būdus;
- Aprašyti ontologijų evoliucionavimą;
- Naudojantis gautomis žiniomis savarankiškai apjungti dvi ontologijas į vieną įrankio Protege pagalba.

Darbo vadovė Lekt. L. Tankelevičienė_____

(vardas, pavardė, parašas)

Turinys

Įvadas

Kasdien pasaulyje daugybėje įvairiose žmonių veiklos srityse yra kaupiamas didelis kiekis informacijos, bet šios informacijos panaudoti kitos sitemos negali, nes ji pateikta skirtingais formatais ir vaizdavimo būdais. Tam, kad sukaupta informacija turėtų praktinę naudą mokslo ir pramonės vystymuisi, ją būtina pateikti bendrame, visiems suprantamame pavidale. Šiandien tam tikslui yra suskurta nemažai priemonių, padedančių pateikti žinias, ir viena efektyviausia iš jų yra būtent ontologijos. Augant žinių mainų svarbai, daugelios pramonės ir akademinės sritys padarė ontologijas savo konceptualiu pagrindu. Tačiau veiklų dinamika ir nuolatos besikeičianti informacinė aplinka dažnai padidina pokyčius programinės įrangos reikalavimams, o tai gali būti įvykdyta tik keičiant pagrindines ontologijas. Tai ypač aktualu WWW ir Semantiniam tinklui.

Šio darbo tema yra ontologijų tobulinimo metodų tyrimas, kuri yra labai aktuali besivystančioje informacinėje visuomenėje. Kadangi metodai, kuriuos taikė prieš dešimtmečius dėl nepaliaujamo progreso dabar gali būti taikomi tik dalinai, būtina išsiaiškinti naujuosius ontologijų kūrimo, tobulinimo ir apjungimo būdų aspektus bei panaudoti įgytas žinias savarankiškai apjungiant dvi ontologijas į vieną.

Norint apjungti dvi ontologijas yra iškeliami uždaviniai, padedantys išsiaiškinti kaip yra skirstomos i vaizduojamos ontologijos, išanalizuoti jau esamas, išnagrinėti kūrimo, tobulinimo ir apjungimo būdus, taipogi ontologijų evoliucionavimą. Visa tai padeda ne tik perprasti kas yra ontologijas, bet ir suteikia vertingų žinių norint savarankiškai apjungti dvi ontologijas į vieną.

Ieškant vertingos informacijos šiam darbui labiausiai buvo akcentuojami pastarojo dešimtmečio informacijos šaltiniai, metodai, kuriuos įvairios įmonės taiko šiomis dienomis. Taipogi buvo atsižvelgta ir į senesnių metų literatūrą, kurioje aprašyti nepasikeitę ontologijų kūrimo pagrindai.

1. Ontologijų struktūra ir taikymas

Tai kas gi yra tos ontologijos?

Ontologijos (graikiškai *ων „būtis“, λόγος „žodis“ ar „kalba“*) — informatikoje yra bandymas visapusiškai ir detaliai formalizuoti kokios nors srities informaciją, naudojant konceptualias schemas. Paprastai tokios schemos susideda iš duomenų struktūros, kurioje yra visų objektų relevantinės[1] klasės, sąryšiai ir taisyklės (teoremos, apribojimai) tai sričiai. Šis terminas pirmą kartą informatikoje buvo panaudotas maždaug 1990 metais dirbtinio intelekto srityse kalbant apie bendrą žinių naudojimą (knowledge sharing), programinių agentų tarpusavio sąveiką, visuotinai pripažįstamų (common sense) žinių atvaizdavimą, natūralios kalbos apdorojimą ir kt. Kitaip ontologijas galima apibrėžti kaip aiškią konceptualizacijos specifikaciją (angl. *explicit specification of a conceptualization*[2]). Visos šiuolaikinės ontologijos kuriamos tokiu pačiu principu ir susideda iš ekzempliorių, koncepcijų, atributų ir sąryšių *(žr. Priedus, paveiksliukas NR 1)*.

1.1. Ontologijų elementai

1.1.1. Ekzemplioriai

Ekzemplioriai (angl. *instances*) arba, kitaip tariant, individai (angl. *individuals*) – tai pagrindiniai žemo lygio ontologijų komponentai. Ekzemplioriai gali būti tiek fiziniais (žmonės, namai, planetos), tiek ir astrakčiais (skaičiai, žodžiai) objektais. Ontologijos gali apsieiti ir be konkrečių objektų, bet pagrindinis jų tikslas – klasifikuoti tokius objektus.

1.1.2. Koncepcijos

Koncepcijos (angl. *concepts*) arba klasės (angl. *classes*) – tai abstrakčios grupės, kolekcijos arba objektų rinkiniai. Jie gali susidėti iš ekzempliorių, kitų klasių arba vienų ir kitų jų junginių.

Pavyzdžiui:

- Koncepcija „žmonės“, įterptoji koncepcija – „žmogus“. Kuo yra „žmogus“ – įtertąja koncepcija arba ekzemplioriumi (individu) – priklauso nuo ontologijos.

[1] Relevantinis - semantinis atitikimas paieškos užklausai

[2] Pagal T.R.Gruber – daugybės knygų ir straipsnių apie ontologijas autorius, Ontoligua formalios kalbos kūrėjas 1993 m.

- Koncepcija „individai“, ekzempliorius „individas“.

1.1.3. Atributai

Objektai ontologijoje gali turėti savus atributus. Kiekvienas iš atributų turi mažų mažiausiai po vardą ir reikšmę, ir yra naudojami specifinei objekto informacijai saugoti. Pavyzdžiui automobilis „Renault Safrane“ turi tokius atributus:

- Pavadiniamas: Renault Safrane
- Durų skaičius: 4
- Variklis: benzininiai {2.0, 2.2, 3.0}, dyzelinai {2.1, 2.5}.
- Pavarų dėžė: 5-ių bėgių.

Atributų reikšmė gali būti sudėtingu duomenų tipu. Pateiktame pavyzdyje atributo „Variklis“ reiškmė yra paprastų duomenų tipui priklausantis reikšmių sąrašas.

Jeigu jūs neapibrėžiate koncepcijų atributus, jums teks apibrėžti arba taksonomiją (jeigų tarp koncepcijų yra hiponimo[3] santykis), arba Valdomąjį žodyną (angl. *Controlled Vocabulary*) – jie yra vertingi, bet nesiskaito tikromis ontologijomis.

1.1.4. Ryšiai

Pagrindinis atributo vaidmuo yra nustatyti priklausomybes (santykius) tarp ontologijų. Dažniausiai santykiu būna atributas, kurio reikšmė yra kitas objektas.

Tarkime, kad automobilių ontologijoje yra du objektai: Renault Safrane ir Renault VelSatis. VelSatis yra Safrane įpedinis, tada santykis tarp Renault Safrane ir Renault VelSatis yra apibėžiamas kaip „isSuccessorOf“ su reikšme „VelSatis“ objektui „Safrane“.

1.2. Kaip skirstomos ontologijos

Ontologijos yra skirstomos pagal formalumą, išreiškimo galią[4] ir sudėtingumo lygį.

[3] Hiponimo santykis (angl. Hyponym) – tai koncepcija, rodanti ryšį su kita, labiau išplėsta, koncepcija. Pavyzdys: Trerminas „aviganis“ yra hiponimas terminui „šuo“, o pastarasis – terminui „gyvūnas“.

[4] Pagal „Ontologies and Problem–Solving Methods: Lessons Learned and Future Trends“ - A. Gómez–Pérez, V. R. Benjamins, 1999m.

1.2.1. Formalumas:

1) Formalios:
    - Paremtos prototipais (terminais);
    - Paremtos aksiomomis;
    - mišrios.
2) Neformalios (pavyzdžiui terminų katalogai)

Formalus ontologijos modelis O=<T, R, F> - tai nuoseklus baigtinių aibių trejetas, kur:

- T — taikomosios srities terminai, kuriuos aprašo ontologija O;
- R — santykiai tarp nustatytos taikomosios srities;
- F —interperetacijos funkcijos, nustatytos terminais ir/arba O ontologija.

Ontologijų modelių klasifikacija:

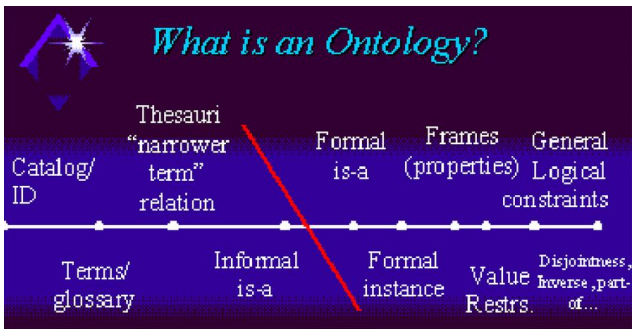
- Paprasti (turi tik konceptus);
- Freimų[5] pagrindu (turi tik konceptus ir savybes);
- Logikos pagrindu (pvz.: Ontolingua, DAML+OIL).

1.2.2. Išreiškimo galia:

- Lengvasvorės - išreiškiančios elementarius tipus, sąvokas, jų hierarchiją, ryšius;
- Sunkiasvorės - susidarančios iš lengvasvorių, bet dar ir išreiškiančios ryšių klasifikaciją, apribojimus, logikos formalizmus ir t.t.)

[5] Freimas (angl. frame) – žinių vaizdavimo būdas dirbtinio intelektos srityje, vaizduojatis veiksmų realioje situacijoje schemą.

1.2.3. Sudėtingumo lygiai[6]:



1 pvksl. Sudėtingumo lygiai[7]

1.3. Ontologijų vaizdavimas

Ontologijas išreiškiančios kalbos turi turėti:

1) Aiškią sintaksę;
2) Formalią semantiką (Tikslų supratimą, automatinės analizės galimybę);
3) Efektyvią automatinę analizę;
4) Pakankamai išraiškingą galią.

Ontologijoms išreikšti gali būti naudojamos tokios kalbos (yra ir daugiau):

„**Silpnos**“ kalbos:

- ER-modeliai, UML-diagramos.
- RDFS (the Resource Description Framework, with schema vocabulary)

„**Stiprios**“ kalbos:

- Aprašomosios logikos, OWL (Web Ontology Language);

[6] Pagal “Ontologies Come of Age” - Deborah L. McGuinness, 2001.

[7] Šaltinis: http://yorkearwaker.wordpress.com/ontologies-are-the-esperanto-for-the-babel-fish-of-the-21st-century/ontologies-are-the-esperanto-for-the-babel-fish-of-the-21st-century-part2/

- Taisyklės (RuleML, LP, Prolog);
- Koncepcijų grafai;
- Predikatų logika.

1.4. Ontologijų tikslas

Kasdien pasaulyje daugybėje įvairiose žmonių veiklos srityse yra kaupiamas didelis kiekis informacijos, kuri, toli gražu, ne visa yra panaudojama. Tam, kad sukaupta informacija turėtų praktinę naudą mokslo ir pramonės vystymuisi, ją būtina pateikti bendrame, visiems suprantamame pavidale. Šiandien tam tikslui yra suskurta nemažai priemonių, padedančių pateikti žinias, ir viena efektyviausia iš jų yra ontologija.

Pagrindinis ontologijos tikslas yra informacijos integracija. Ontologijas sieja du svarbūs aspektai:

1) Jos apibrėžia formalią informacijos semantiką, tam, kad tuos duomenis galėtų apdirbti kompiuteris;
2) Jos apibrėžia realaus pasaulio semantiką, padedant bendros terminologijos pagrindu susieti informaciją, kuri yra pateikta kompiuteio supratimui ir informaciją, kuria mes, žmonės, nesunkiai suprantame.

Ontologijos naudojamos programavime kaip žinių apie realų pasaulį ir jo dalis pateikimo forma. Pagrindinės taikymo sritys:

1) **Biznio procesų modeliavimas** – organizacijų modelių formavimas, apibrėžiantis dalykinius objektus (padaliniai, pareigos, resursai, vaidmenys, procesai, operacijos, informacinės sistemos, ir t.t.) ir nurodantis ryšius tarp jų;
2) **Semantinis tinklas** - ateities interneto tinklas (globali duomenų bazė), kurio infrastruktūra leidžia žmonėms ir kompiuteriams kategorizuoti informaciją ir ją panaudoti.
3) **Dirbtinis intelektas** – tai dirbtinai sukurtas intelektas, galintis atlikti tą patį veiksmą skirtingai, priklausomai nuo to, kokie veiksmai buvo atlikti prieš tai.

Ontologijos informatikoje turi turėti tokį formatą, kurį suprastų ir lengvai apdirbtų kompiuteris. Informacinės ontologijos visada yra kuriamos siekiant konkrečių tikslų – konstruktorinių uždavinių sprendimui.

1.5. Ontologijų privalumai

Ontologijos

- palengvina žinių struktūrizavimą naujose srityse;
- turi intelektualią paiešką apdorojant užklausas (automatiškai jas apibendrina);
- daro patogesnį bibliotekoje saugomų komponentų pakartotiną naudojimą (angl. *reuse*);
- palengvina skirtingų komponentų tarpusavio sąryšį.

1.6. Ontologijų taikymas

Ontologijos yra taikomos įvairiose srityse:

- Bibliotekininkystėje;
- matematikoje;
- natūralios kalbos generavime;
- profesinės terminijos standartizavime;
- duomebaziių koncepcinėse schemose, koncepciniame modeliavime;
- Žinių bazėse;
- Korporaciniame žinių valdyme;

Ir t.t.

1.7. Kaip kuriamos ontologijos

Paprastai ontologijos kūrimo procesą sudaro keturi etapai[8]:

- **terminų rinkimas**[9] (yra identifikuojami visi nagrinėjamos srities vartojami terminai ir jų tarpusavio ryšiai, taipogi apibrėžimai);

[8] Ontology Evolution - Raul Palma, Peter Haase, psl. 1

- **terminų analizė** (peržiūrėjus visus skirtingus terminus, apibūdinančius vienodus objektus arba reiškinius, yra pereinama prie paskutinio etapo – ontologijos vaizdavimo pasirinkta ontologijųvaizdavimo kalba);
- **tikslinimas**;
- **vaizdavimas**.

Bet toks auksčiau aprašytas ontologijos kūrimo procesas yra gana lėtas ir reikalauja daug kruopštumo. Vienas iš būdų pagreitinti ontologijos kūrimą yra terminų analizės ir ontologijos vaizdavimo etapų automatizavimas. Deja, šiuo metu dar nėra sukurta sistemų, leidžiančių automatizuoti minėtus ontologijos kūrimo etapus, bet yra kelios, otologijų kūrimą palengvinančios priemonės (žr. priedus NR.13).

2. Ontologijų rūšys

2.1. Trumpai apie rūšis

Viso yra septynios ontlogijų hierarchijos (rūšys)[10]:

1) Vaizdavimo ontologijos – apibrėžia konceptualizaciją, kuri yra žinių vaizdavimo pagrinde.
2) Bendros ontologijos – jose yra fundamentalūs konceptualizacijos aspektai (pavyzdžiui tokios kategorijos kaip „lytis“, „priežastis“ ir t.t.).
3) Tarpinės ontologijos – turi bendrus supratimus ir santykius (sąsajas), kurie yra konkrečioje dalykinėje srityje. Tai gali būti interfeisas tarp skirtingų dalykinės srities dalių.

---

[9] „Formalių konceptų analizė, kuriant taikomosios srities ontologijas“ - Darius jurkevičius, Olegas Vasilecas, Algirdas Laukaitis

[10] „CREATE PRODUCTION SYSTEM FOR ANALYSES SCIENTIFIC TEXTS“ L.V.Najhanova, N.B.Haptahaeva, 1psl.

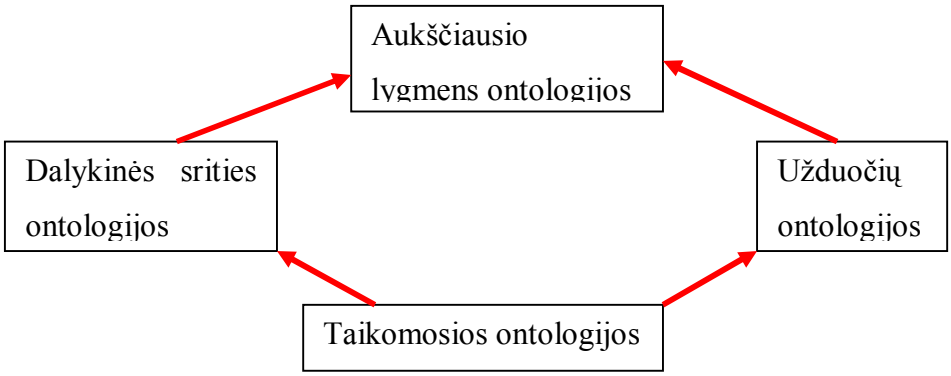
4) Aukščiausio lygmens ontologijos – tai konkreti bendrų ir tarpinių ontologijų koncepcija.

5) Dalykinės srities ontologijos – turi apibrėžtos žinių srities koncepcijas.

6) Užduočių ontologijos – apibrėžia konkrečius uždavinius žinių srityje.

7) Taikomosios (taikymo) ontologijos – tai dalykinės srities ir uždavinių ontologijų specializacija.

Grafiškai pagrindinias ontologijų rūšis galima pavaizduoti taip:



2 pvksl. Pagrindinės ontologijų rūšys

Bendroms ontoligijoms priskiriami: atvejai, esmė, laikas, edvė, priežastys ir panašūs dalykai.

Užduočių ontologijoms priskiriami: tvarkaraščių sudarymas, tiklsų apibrėžimas, klasifikacija.

Dalykinėms ontologijoms: daugybė objektų - skeneriai, spausdintuvai ir t.t.

2.2. Aukščiausio lygio ontologijos

Aukščiausio lygio ontologijos paskirtis yra sukurti vienintelę dažnai naudojamą, „teisingą“ ontologiją, kuri bendrai fiksuoja kelių dalykinių sričių žinias. Egzistuoja kelios aukšto lygio ontologijos - Cyc, DOLCE, SUMO, J. Sowa ir kt. (žr. Priedus „Žymiausių aukščiausio lygių onfologijų aprašymas„).

Buvo daug bandymų sukurti aukščiausio lygių ontologija, pritaikyta visiems gyvenimo atvejams, bet kolkas visos pastangos buvo bevaisės. Dauguma aukščiausio lygio ontologijų panašios ir turi tuos pačius konceptus: esmę, reiškinį, procesą, objektą, vaidmenį, erdvę, laiką, materiją, įvykį, vaiksmą ir t.t.

2.3. Vaizdavimo ontologijos

(Žinių) vaizdavimo ontologijų *(angl. The frame ontology)* sukūrimo tikslas yra aprašyti žinių srities vaizdavimą, sukurti kalbą kitų, žemensnių lygių ontologijų, specifikacijai. Pavyzdžiui: kalbos OWL koncepcijų aprašymas taikant RDF/RDFS priemones. Pateiktame pavyzdyje (pvksl.) apibrėžtos tokios koncepcijos kaip „klasė“, „santykis“, „savybės reikšmės apribojimas“, „domenas“ ir t.t.

2.4. Dalykinės srities ontologijos

Kitas dalykinių sričių ontologijos pavadinimas – domeno ontologija (angl. *domain ontology arba domain-specific ontology*). Jos paskirtis yra artima aukščiausio lygio ontologijų paskirčiai, bet interesų sritis apribota tik viena dalykine srtitimi (domenu), pavyzdžiui: aviacija, medicina, kultūra, nuotolinis mokymas, Interneto technologijos. Dalykinės srtities ontologija apibendrina koncepcijas, kurios naudojamos domenų užduotyse, atsiribojant nuo pačių užduočių (pvz: automobilių ontologija nepriklauso nuo automobilių markių konkrečių savybių). Daugelyje disciplinų dabar yra kuriamos standartinės ontologijos, kurias naudoja dalykinių sričių ekspertai informacijos panaudojimui arba charakterizavimui tam tikroje srityje.

Pavyzdžiui, medicinos srtiyje yra sukurti dideli, struktūrizuoti žodynai, tokie kaip

SNOMED CT (angl. *Systematized Nomenclature of Medicine - Clinical Terms*) – sistematizuota medicinos nomenklatūra – klinikinė terminologija.

UMLS (angl. *Unified Medical Language System)* — semantinis Unifikuotos medicininės kalbos sistemos tinklas.

Taipogi yra kuriamos didelės bendros paskirties ontologijos. Tokiu būdu Jungtinių Tautų Organizacijos plėtros programa kartu su Dun&Bradstreet kompanija plėtoja UNSPSC ontologiją, kuri pateikia prekių ir paslaugų terminologiją.

Dar vienas pavyzdys – dokumentacinė ontologija kultūrinio paveldo srityje CIDOC CRM.

Dalykinės sritiės ontologijas galima nagrinėti kaip specifinį terminų žodiną (tik koknkrečiai sričiai) kartu su aksiomų rinkiniu, kurios apibrėžia teisingą šių terminų panaudojimą ir jų

interpretaciją. Šiuo metu formalaus ontologijų aprašymo priemonės turi kelias alternatyvas (žr. Priedus "Dalykinių sričių ontologijų alternatyvos plačiau").

1) Dalykinės srities ontologijos vaizdavimas „Dublinio branduolio" (angl. *Dublin Core, DC*) metaduomenų elementų rinkinio pagalba;
2) Dalykinės srities ontologijos vaizdavimas pirmo lygio logikos (predikatų skaičiavimas) kalbų pagalba.
3) Dalykinės srities ontologijos vaizdavimas naudojant ontologijų aprašymo standarto kalbą OWL (angl. *Ontology Web Language*) informaciniams WEB antros kartos resursams, XML platformos pagrindu.

2.5. Bendros ir tarpinės ontologijos

Bendros ontologijos (angl. *Generic ontologies*) panašios į dalykinės srities ontologijas, bet jų aprašomos koncepcijos yra bendros kelioms dalykinėms sritims. Dažniausiai tokios ontologijos aprašo tokias koncepcijas kaip būsena, įvykis, procesas, veiksmas, komponentas, jose yra fundamentalūs konceptualizacijos aspektai. Tokios ontologijos turi bazinį terminų rinkinį, specializuotą žodyną arba tezaurą[11], naudojamą dalykinių sričių terminų aprašymui. Kai kurios bendros ontologijos yra kuriamos sistemoje kaip šablonai (moduliai) kurie yra naudojami kitose ontologijose.

Bendros ontologijos panaikina taškinės integracijos būtinumą ir supaprastina programų integraciją, užtikrinant vieną ir tą pačią semantinę reikšmę visoms programoms, tokiu būdu palengvinant sistemos funkcionavimą ir jos atsinaujinimą.

Tarpinės ontologijos naudojamos tiesioginiam mašininiam vertimui**,** turi bendrus supratimus ir santykius (sąsajas), skirtus konkrečiai dalykinei sričiai arba programinei įrangai. Idealiu atveju tarpinės ontologijos naudojamos kaip interfeisas tarp dalykinių sričių ir bendrų ontologijų, bet gali būti ir aukčiausio lygio ontologijomis aprašant konkrečios programinės įrangos žinias.

2.6. Užduočių ontologijos

[11] Tezauras – tai bendrasis arba kurios nors srities žodynas, nurodantis prasminius žodžių ryšius.

Užduočių ontologijos aprašo konkrečių užduočių (pvz.: diagnostika, prekyba, PĮ kūrimas) ar veiklų žodynus (specializuojant aukščiausio lygio ontologijų terminus).

Galima išskirti kelis užduočių ontologijos konceptų tipus:

- Užduotys;
- Operacijos;
- Duomenys (įvedami/išvedami).

Užduočių ontologija yra naudojama konkrečioje taikomojoje programoje ir turi terminus, kurie panaudojami programinės įrangos, atliekančios konkrečią užduotį, kūrimo procese. Ji vaizduoja programos specifiką ir gali turėti kai kuriuos bendrus terminus (pavyzdžiui, grafiniame redaktoriuje bus specifiniai terminai – palitra, užpildymo tipas, sluoksniai ir t.t., o bendri terminai - išsaugoti bylą, atisiųsti bylą ir pan.). Užduočių ontologija naudoja aukščiausio lygio ontologijos terminų specializaciją.

Užduočių ontologijos pavyzdys – verslo procesų modeliavimo kalba (angl. *Business Process Modeling Language*), verslo procesų ir juos palaikančių esybių abstraktus modelis. BPML apibrėžia formalų modelį abstraktiems ir vykdomiems procesams išreikšti, apimantį visus įmonių verslo procesus (tame tarpe įvairaus sudėtingumo veiklas, transakcijas bei jų kompensavimą, duomenų valdymą, konkuravimą, išimčių apdorojimą ir operacinę semantiką). BPML išreiškia ir gramatiką (XML schemos forma) įgalinančią išsaugoti ir keistis apibrėžimais heterogeninėse sistemose ir modeliavimo priemonėse. BPML specifikacijos galutinis juodraštinis variantas ir verslo procesų modeliavimo žymėjimų sistema baigta ruošti ir pateikta viešai[12].

Taikomosios užduoties specifikacijų kūrimo metodas turi būti realizuotas kaip procesorius (interpretatorius), nustatytas konkrečiai koncept-užduočiai ir nuosekliai vykdantis šios užduoties bazines operacijas, be to turi turėti visas programuojamos aplinkos savybes. Tokiu būdu bet kuri procesoriaus sprendžiama užduotis yra užduočių ontologijos konceptas susijęs su kitais konceptais ryšių „priklausomybės-sekimo“ pagalba. Užduočių ontologija gali būti susijusi su savybių ontologija per koncept-duomenų parametrų konkretizacijos mechanizmus.

[12] pagal „MODERNIŲ INFORMACINIŲ SISTEMŲ ONTOLOGIJOS IR PASLAUGŲ REIKALAVIMŲ FORMULAVIMAS“, Saulius Maskeliūnas, psl. 12

2.7. Taikomosios ontologijos

Taikomųjų sistemų ontologijos yra tai dalykinės srities ir uždavinių ontologijų specializacija ir aprašo konceptus, kurie yra tiek taikomųjų sričių, tiek užduočių ontologijų specializacija. Dažnai šie konceptai atitinka roles, kurias atlieka tam tikros taikomosios srities esybės vykdydamos tam tikrą veiklą. Tokios ontologijos tikslas yra aprašyti konkretaus uždavinio arba programos koceptualinį modelį. Taikomosios ontologijos taipogi aprašo konceptus, priklausončius nuo užduočių bei dalykinės srities ontologijos. Tokios ontologijos turi labiau specifinę informaciją. Pavyzdžiai:

1) TOVE (angl.*Toronto Virtual Enterprise*). Projekto tikslas – sukurimas modelio, kuris turi:

    - užtikrinti bendrą terminologiją dalykiniai sričiai, kurios programos gali būti bendrai naudojamos ir suprantamos kiekvienu dalyviu;
    - Tiksliai ir neprieštaringai apibrėžti kiekvino pirmo lygio logikos termino reikšmę;
    - Užtikrinti semantikos užduotį aksiomų, leidžiančių automatiškai gauti atsakymą apie dalykinę sritį, pagalba.

    TOVE turi užtikrinti konkrečios srities integruoto modelio kurimą, kuris susideda iš: operacijų, būsenų ir laiko, organizacijų, resursų, serviso, gamybos, kainos, kiekio, produkto.

2) Plinius – projekto tikslas yra pusiauautomatinis žinių gavimas iš tekstų parašytų narūralia kalba. Kadangi tekstai apima gan platų koncepcijų diapazoną, reikalinga daugybė integruotų ontologijų, apimančių tokiais koncepcijas kaip keraminės medžiagos ir jų savybės, perdirbimo būdai, skirtingi medžiagų defekrtai.

Realaus pasaulio objektų specifikacijai taikomojoje ontologijoje yra nurodoma daugybė šių objektų atributų,, galimų reikšmių sritys ir santykiai, egzistuojantys tarp specifikuojamų objektų. Patys objektai neįeina į taikomųjų ontologijų sąvoką. Vietoj to ontologija turi nuorodas į šaltinius, iš kurių gautos periskirtos atributams reikšmės. Taikomoji ontologija be žinių apie užduotis ir metodus taipogi turi ir konceptualių žinių modelį.

3. Ontologijų evoliucija

Per tam tikrą laikotarpį ontologija turi būti pakeista tam, kad atspindėtų pokyčius realiame pasaulyje, pokyčius vartotojo reikalavimuose, trūkumus pradiniame projekte arba tiesiog tam, kad įtraukti papildomas funkcijas arba leisti ją tobulinti.

Ontologijos yra dinamiški subjektai besivystantys bėgant laikui. Pavyzdžiui, domenai yra nei statiniai nei fiksuoti. Ir konceptualizacija arba formali ontologijos specifikacija gali keistis. Tai kas gi yra ta ontologijos evoliucija?

Ontologijos evoliucija remiasi ontologijos modifikavimo palengvinimu, tuo pačiu išsaugant jos nuoseklumą. Ji gali būti vertinama kaip skirtingų veiklų pasėkmės ontologijos plėtros metu.

Ontologijos evoliucijos tikslas yra teikti apibrėžtą procesą (su įrankių palaikymu) vienam arba keliems ontologijų atnaujinimams ir pakeitimams. Paprastai tai įvyksta jau įdiegtai ontologijai, kurią reikia arba atnaujinti arba pakeisti dėl tam tikrų pasikeitusių reikalavimų.

Ontologijos kūrimas yra dinamiškas procesas, prasidedantis „žaliomis“, neapdorotomis ontologijosmis, kurios vėliau yra peržiūrimos, tobulinamos ir užpildomos detalėmis. Taigi ontologijos turi būti neabejotinai apdorojamos siekiant:

- Ištaisyti klaidas pradinėje ontologijoje;
- Prisitaikyti prie pasikeitusios aplinkos;
- Pasitaisyti po to, kai ontologija pradėjo veikti;
- Vengti būsimų pakeitimų ir palengvinti palaikymą.

Nors ontologijos plėtros metu pokyčiai yra neišviangiami, dauguma dabartinių ontologijos redaktorių, deja, nepalaiko veiksmingos pokyčių kopijavimo galimybės. Kadangi ontologijos redaktoriai yra pagrindinės priemonės, skirtos ontologijos kūrimui, evoliucijos palaikymas jose turi būti būtina opcija ir jie turi atitikti sekančius reikalavimus[13]:

[13] Ontology Evolution within Ontology Editors - L. Stojanovic, B. Motik

- Funkcinius – nurodo visus palaikomus evoliucijos pokyčius;
- Vartotojo priežiūros – leidžia vartotojo procese spręsti apie pokyčius;
- Skaidrumo reikalavimas – susijęs su evoliucijos proceso kontrole per evoliucijos operacijos taikymo sritį prieš tai, kol pati operacija yra panaudojama;
- Grįžtamumo – nurodo kaip evoliucijos pokyčiai gali būti atšaukti;
- „Audito“ reikalavimas – susijęs su ontologijos valdymu pokyčių pokyčių istorijoje;
- Ontologijos tobulinimas – teikia palaikymą nuolatiniam ontologijos gerinimui;
- Naudojimo reikalavimas - leidžia vartotojui valdyti pokyčius lengviau randant ontologijos neatitikimus ir teikti paaiškinimą kaip juos išspręsti.

Tam, kad vartototjas gautų labiausiai jo reikalavimus atitinkančią ontologiją, mes išnagrinėjome ontologijos apdorojimo problemas ir redaktoriaus reikalavimus, nustatančius ontologijos evoliucijos procesą. Tai galima daryti keliais būdais: praturtinti galimų pokyčių sąrašą; leisti vartotojui sukurti vieną iš evoliucijos strategijų, kuri yra naudojama sprendžiant pokyčių problemas; informuoti jį apie visus pokyčių padarinius; leisti atšaukti pokyčius; leisti patikrinti jau atliktus pakeitimus; pasiūkyti vartotojui generuoti pokyčius, nustatyti jų nesuderinamumą ir pateikti atsakymus į klausimus „kaip?“, „kodėl?“, „kas jeigu?“ ir t.t.

4. Ontologijų kūrimas ir tobulinimas

4.1. Vynų ir maisto ontologija. Kūrimas

Vynų ir maisto ontologijos pavyzdys – vienas iš populiariausių pasaulyje. Ši ontologija gali būti kaip pagrindas dirbant su restorano instrumentais, ją ontologiją galima naudoti programoms (sistemoms), kurios siūlys gerus vyno ir maisto tarpusavio derinius.

Daugumos ontologijų centre yra klasės, kurios aprašo dalykinę sritį. Pavyzdžiui, vynų klasė turi aprašytus visus vynus. Konkretūs vynai – tai šitos klasės egzemplioriai. Vynas Bordeaux

– vynų klasės Bordeaux egzempliorius. Klasė gali turėti poklasių, kurios aprašo konkretesnes koncepcijas. Pavyzdžiui, mes galime padalinti visų vynų klasę į putojančius ir neputojančius.

Slotai[14] apibrėžia klasių ir egzempliorių savybes: vynas Chвteau Lafite Rothschild Pauillac – stiprus, gaminamas Chateau Lafite Rothschild gamykloje. Mes turime du slotus, aprašančius vyną šitame pavyzdyje:

1) stiprumas (su reikšme stiprus);
2) gamintojas (su reikšme Chateau Lafite Rothschild).

Mes galime pasakyti, kad klasės lygyje pas egzempliorius klasės *Vynas* yra slotai, kurie aprašo skonį, stiprumą, cukraus lygį, gamintoją ir t.t.

Praktikoje ontologijų kūrimas susideda iš šių etapų:

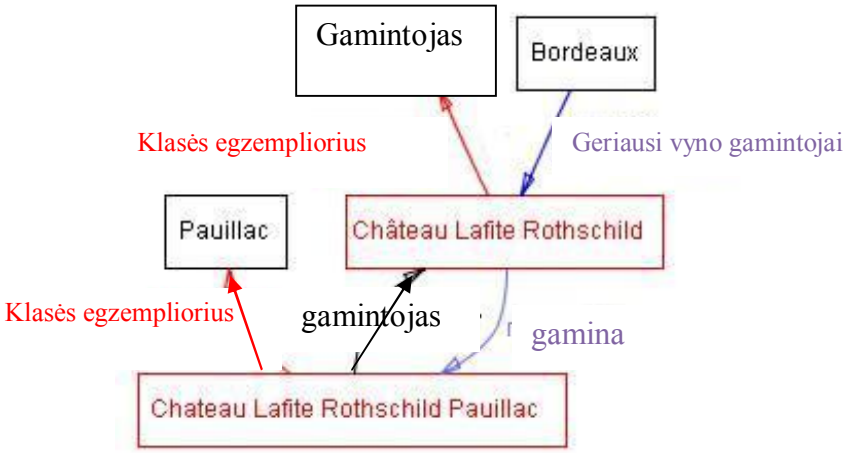
1) klasių apibrėžimas;
2) klasių hiedarchijos sudarymas (klasė-poklasis);
3) slotų apibrėžimas ir jų galimų reikšmių aprašymas;
4) egzempliorių slotų reikšmių užpildymas.

Atlikus šitus etapus jau galima apibrėžti ontologijos žinių bazę.

Pabandykime vaizdžiai pateikti etapus, kad būtų patogiau įsivaizduoti ontologijos kūrimo procesą.

<u>Klasių apibrėžimas</u>:

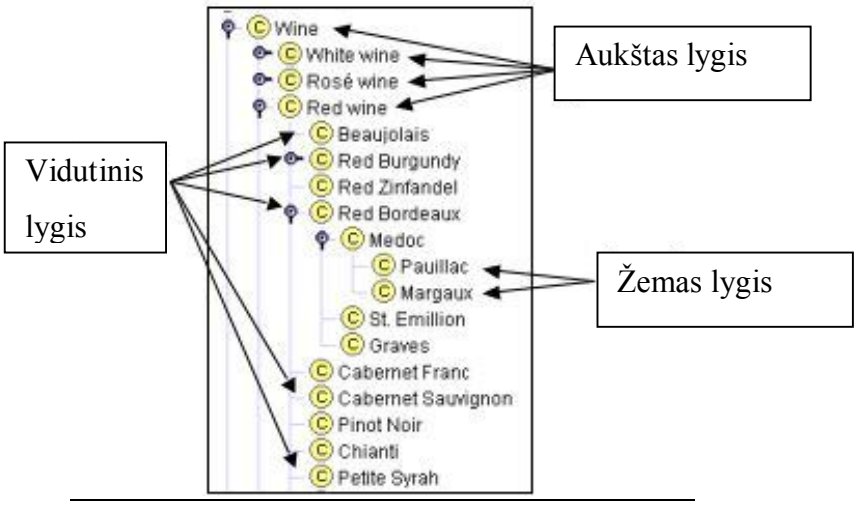
[14] Slotai – tai dalis objekto būsenos. Slotų visuma sudaro objekto struktūrą.



3 pvksl. Vyno ontologijos pavyzdinis klaisų apbrėžimas[15]

Juodai pažymėtos klasės, raudonai – egzemplioriai. Tiesioginiai ryšiai žymi slotus ir vidinius ryšius, tokius kaip „klasės egzempliorius“ ir „klasės poklasė“.

Klasių hierarchijos sudarymas gali būti toks:



4 pvksl. Vyno ontologijos hierarchijos

Egzistuoja keli būdai sudaryti klasių hierarchiją[16]:

1) Mažėjantis principas:

[15] Šaltinis (išverstas): http://ifets.ieee.org/russian/depository/ontology101_rus.doc , p. 4

[16] Pagal Uschold and Gruninger, 1996

Kūrimas prasideda nuo pačių bendriausių dalykinės srities konceptų apibrėžimų su palaispnine koncepcijų konkretizacija. Pavyzdžiui, mes galime pradėti nuo bendrų klasių *Vynas* ir *Maistas* kūrimo. Toliau sukonkretizuoti klasę *Vynas*, sudarant jos poklases: baltas vynas, raudonas ir t.t. Galime ir toliau kategorizuoti klasę *Raudonas vynas*: pvz. Syrah, Red Burgundy, Cabernet Sauvignon ir t.t.

2) Didėjantis principas:

Apibrėžiamos pačios konkrečiausios klasės, hierarchijos lapai, su tolimesniu šių klasių grupavimu į bendresnes koncepcijas. Pavyzdžiui, iš pradžių mes apibrėžiame klases vynams Pauillac ir Margaux. Toliau mes sukuriame bendrą šių dviejų klasių klasę, esančią lygiu aukščiau: *Medoc*, kuris yra *Bordeaux*'o poklasė.

Kombinuotas principas:

Aukščiau paminėtų dviejų principų kombinacija. Iš pradžių mes apibrėžiame žymias koncepcijas, toliau jas apibendriname ir apribojame. Mes galėtume pradėti nuo kelių aukščiausio lygio koncepcijų, tokių kaip Margaux. Toliau mes galime sujungti jį su vidutinio lygio koncepcija, tokia kaip Medoc. Galų gale mums reikėtų suformuluoti visas vyno klases, tuo pačiu formuojant vidutinio lygio koncepcijų seką.

<u>Slotų apibrėžimas ir jų galimų reikšmių aprašymas:</u>

Iš savybių sąrašo mes turime apibrėžti kokią klasę aprašo viena ar kita savybė. Šitos savybės ir taps slotais, pririštais prie klasių. Pavyzdžiui, klasė *Vynas* turė tokius slotus:

- Spalva;
- Stiprumas;
- Skonis;
- Cukrus.

Apskritai ontologijoje slotais gali tapti keli objektų tipai:

- Vidinės savybės (kaip vyno skonis);
- Išorinės (vyno pavadinimas);

- Dalys, jeigu objektas turi struktūrą; jos gali būti kiek fizinės tiek abstrakčios (tarkime patiekalai, įeinantys į pietus);
- Santykiai su kitais individiniais konceptais; tai santykiai tarp atskirų klasių ir kitų elementų (tarkime, vyno gamintojas ir vynuogės, iš kurių padarytas vynas).

Egzempliorių slotų reikšmių užpildymas:

Slotai gali turėti skirtingus facetus[17], štai dažniausiai naudojamų facetų sąrašas:

1) Eilutė – paprasčiausas reikšmių tipas, taikomas pvz. vyno pavadinimui aprašyti;
2) Skaičius (float, integer) – aprašyti vyno kainą;
3) Bollean slotai – turi reikšmes „taip“ ir „ne“. Pavyzdžiui galima sudaryti putojančių vynų klasę, vyno savybėse nustačius reikšmę „taip“, nurodančią, kad vynas yra putojantis.
4) Numeruoti – konkrečios leidžiamos sloto reikšmės. Pavyzdžiui, galime nustatyti, kad skonis gali turėti tik tris reikšmes: stiprus, vidutinis, švelnus. Programoje Protege numeruoti slotai turi tipą „Simbolius“.
5) Slotai-egzemplioriai – padeda apibrėžti santykius tarp individualių konceptų. Turi apibrėžti leidžiamų klasių sąrašą, kurio egzempliorius galima naudoti. Pavyzdžiui, slotas Vyno gamintojas“ gali turėti egzempliorių Vyno klasės.

4.2. Vynų ir maisto ontologija. Tobulinimas

Keturių ontologijos kūrimo etapų užtenka, kad sukurti įprastinę vyno ir maisto ontologiją, tačiau ši ontologija neatitiks visų lūkesčių, tad ją reikia patobulinti, labiau detalizuoti. Vienas iš būdų apibrėžti ontologijos mastą ir ją detalizuoti – sudaryti sąrašą klausimų, į kuriuos turi atsakyti ontologijos pagrįsta duomenų bazė, t.y. klausimai kompetencijai patikrinti[18]. Šitie klausimai bus kaip priminimas: ar yra ontologijoje pakankamai informacijos šiems

[17] Facetas – aprašo reikšmių tipą, leidžiamas reikšmes, reikšmių skaičių (galią) ir kitas savybes, kurias gali priimti slotas.

[18] Gruninger and Fox, 1995

klausimams atsakyti? Ar reikalingas atsakymams ypatingas detalizacijos lygis ir dalykinės srities vaizdavimas?

Vynų ir maisto ontologijoje galimi sekantys klausimai kompetencijai patikrinti:

1) Kokias vyno charakteristikas man reikėtų turėti omany išsirenkant vyną?
2) Vynas Bordeaux raudonas ar baltas?
3) Ar gerai derinasi Cabernet Sauvignon su jūrų produktais?
4) Koks vynas geriau tiks prie keptos mėsos?
5) Kokios vyno charakteristikos lemia jo suderinamumą su maistu?
6) Ar įtakoja vyno pagaminimo metai jo stiprumui?
7) Kokie Napa Zinfandel derliai buvo geri?

Sprendžiant iš šito klausimų sąrašo ontologija turi turėti informaciją apie skirtingas vyno ir vyno tupų harakteristikas, vyno gamybos metus (gerų ir blogų), maisto klasivikacijos, į kurias reikia atsižvelgti norint išsirinkti tinkamą tam tikram maistui vyną. Tačiau mažai tikėtina, kad šita ontologija turės koncepcijas, skirtas vynotekų valdymui arba restorano darbutojams. Tam reikia modifikuoti ir praplatinti esamą ontologiją:

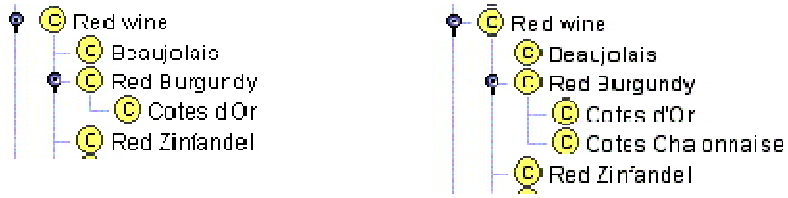
1) Jeigu šita ontologija bus naudojama klijentų apsisprendimui kokį vyna ir maistą užsakyti, reikia įtraukti į ją prekių kainas.
2) Jeigu dalykinė sritis bus aprašyta tokia kalba, kurios nesupranta pirkėjai, tai ontologijos kūrėjams būtina nurodyti sąsajas – atitikmenų lentelę tarp šių dviejų klabų.

Sukurti iš pirmo karto gerą, išsamią ontologiją – praktiškai neįmanoma. Sukurtą ontologiją būtina tobulinti kol gausis norimas rezultatas. Visų pirma siūloma susivarkyti su klasėmis ir poklasiais. Neegzistuoja tvirtų taisyklių, nurodančių kiek tiesioginių poklasių turi turėti klasė. Tačiau daugeliose ontologijose, turinčiose aiškią struktūrą, yra nuo dviejų iki tuzino poklasių. Yra taikomi du vadovaujantys principai:

1) Jeigu klasė turi tik vieną tiesioginę poklasę, tai įmanoma, kad modeliuojant buvo padaryta klaida arba ontologija yra nepilna.

2) Jeigu tyriamoji klasė turi daugiau nei tuziną poklasių, tai įmanoma, kad jai reikia sukurti papildomas tarpines kategorijas.

Kaip pavyzdį galima pateikti klasės Red Burgundy poklases:



5 pvksl. klasės Red Burgundy poklasės [19]

Kaip nustatyti, kad ontologija buvo sukurta teisingai?

Tam, kad galėtume tobulinti ontologijas, reikia pirma nustatyti ar ontologija buvo sukurta teisingai. Kiekvienai dalykinei sričiai gali egzistuoti begalė ontologijų, juk kiekviena nauja ontologija – tai viso labo vienas iš koncepcijų bei santykių tarp jų struktūrizavimo būdų. Tačiau egzistuoja keleta paprastų principų, kurie padeda ontologijų kūrimui, bei jų tobulinimui[20]:

- Negali būti tik vieno būdo aprašyti tam tikrą dalykinę sritį – visada yra alternatyva. Geriausias sprendimas beveik visada priklausys nuo to, kokia sistema bus kuriama arba yra jau sukurta ir nuo būsimų šios sistemos tobulinimų;
- Ontologijos kūrimo procesas visada turi būti iteratyvus;
- Ontologijų koncepcijos turi būti maksimaliai artimos objektams (loginiams bei fiziniams) ir santykiams tarp jų. Teisingai modeliuojant, ontologija gali būti pateikta sakinių forma, kur daiktavardžiais bus objektai, o santykiais – veiksmažodžiai.

Jeigu sukurta ontologija neatitinka bent vieno iš šių punktų, ji reikalauja perdarymo arba patobulinimo.

4.3. Semantic Web

[19] Šaltinis: http://ifets.ieee.org/russian/depository/ontology101_rus.doc , p. 13

[20] Pagal Д.И. Муромцев „Онтологический инжиниринг знаний в системе PROTÉGÉ“

Kita įžymi ontologija yra „Samantic Web“. Pirmą kartą ji buvo pristatyta 2001m., bet jos idėja buvo ne nauja. Jos tikslas yra automatizuoti „intelektualias“ užduotis ir apdirbti resursų reikšmes. Informacijos apdirbimu ir mainais turi užsiimti ne žmonės, o specialūs intelektualūs agentai (programos, patalpintos tinkle). Tačiau šiam tikslui pasiekti, agentai privalo turėti kiekvieno resurso reikšmės bendrą (visiems suprantamą) formalų vaizdavimą. Per 10 „Semantic Web“ gyvavimo metų jos pagrindu buvo realizuota daug projektų, tačiau nepaisant atskirų sėkmingų pristatymų, iki šiol negalima sakyti, kad Semantic Web idėja buvo realizuota praktikoje taip, kaip tai buvo tikėtasi.

Darbai apibrėžiant semantikas Tinkle prasidėjo 1997m., kada konsorciumas W3C apibrėžė specifikaciją RDF[21]. RDF tai paprasta bet galinga kalba naudojama resursams aprašyti, yra sudaryta tripletų pagrindu „Subjektas-Predikatas-Objektas“ ir specifikacijos URI. Šis žingsnis padėjo pagerinti funkcionalumą ir skirtingų duomenų mainų galimybę, kurie Tinkle yra vieni iš svarbiausių. RDF specifikacija vadovaujasi ankstesniais Web standartais:

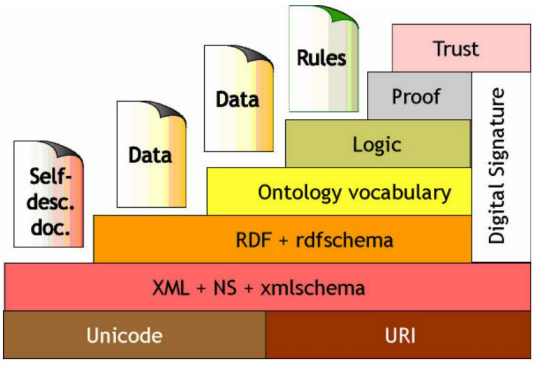
- Unicode skirtas atvaizduoti skirtingas kalbų abeceles;
- URI – apibrėžia unikalius resursų identifikatorius;
- XML ir XML Schema – strukturizavimui ir duomenų mainams siekiant saugoti RDF.

Be RDF buvo sukurtas strukturizuotų žodynų kalba RDF Schema (RDFS). Tai yra minimalus įrankių, skirtų specifikuoti ontologijas, rinkinys.

2005 metais prasidėjo RIF kūrimas – taisyklių mainų formatas. Jo tikslas – sujungti viename standarte keletą formalizmų tam, kad aprašyti taisykles.

Semantic Web yra vaizduojamas kaip sluoksniuotas pyragas:

[21] RDF (angl. Resource Description Framework) – duomenų vaizdavimo modelis, dalis semantinio tinklo koncepcijos



6 pvksl. Semantic Web diagrama[22]

Lygis „Ontology vocabulary“ ir „Logic“ atitinka OWL ir RIF. Lygis „Trust“ kolkas nepriklauso jokiems standartams. Čia ir susidaro viena didžiausių problemų: automatizuotos korektiškumo ir taisyklingumo patikrinimo palaikymas.

Dar viena Semantic Web realizavimo problema: faktinis nebuvimas dirbančių intelektualių agentų. Ne kiekviena programa, apdorojanti RDF yra Semantic Web agentas taip pat, kaip ne kiekviena programa, parašyta Protege programos pagalba, yra dirbtinio intelekto programa.

Pagrindinis Semantic Web uždavinys: naujų ontologijų kūrimas ir esamų tobulinimas.

4.4. Lingvistinė ontologija WordNet

Wordnet yra lingvistinė ontologija ir jo pagrindu buvo padaryta tūkstančiai eksperimentų informacinės paieškos srityje. WordNet‘o 2,1 versijoje yra apie 1500 tūkst. skirtingų leksemų ir žodžių derinių, 117 tūkst. koncepcijų, bendras skaičius porų leksema-reikšmė sudaro net 200 tūkst. Į žodyną įeina 4 kalbos dalys: būdvardis, daiktavardis, veiksmažodis ir prieveiksmis.

Tam, kad realizuoti automatinio konceptualinio indeksavimo ir konceptualinės paieškos shemą, reikia turėti lingvistinį resursą, organizuota kocepcijų ir reikšmių pagrindu. Todėl tokie resursai kaip WordNet gali būti naudojami kaip pagrindas organizuojant konceptualių indeksavimų ir paieškos programų pagrindu. Buvo įvykdyti du tokie eksperimentai:

[22] Šaltinis: http://www.mkbergman.com/231/from-data-federation-pyramid-to-the-semantic-web-birthday-cake/

1) Bandymas ieškoti dokumentų atskirų žodžių pagrindu[23]. Kiekvienam dokumentui iš pradžių buvo atliekama daugiareikšmių daiktavardžių sprendimo procedūra, kuri yra vienintelė rezultato reikšmė ir kiekvienam tekstui yra siejamas jam tinkantis sinsetų vektorius. Po to, kai vektorius sukurtas, jam gali būti taikomos tos pačios operacijos kaip ir pažodiniams vektoriams.

2) Meaning projektas – WordNet resursas, skirtas angliškai kalbančių žmonių rinkai. Buvo nutarta, kad kiekvienas wordnet'as turi išlaikyti savo kalbos specifikas. Tuo pačiu kiekvienas wordnet'as turi nuorodas į angliško wordnet'o reikšmes, kas leidžia ne tik lyginti wordnet'us, bet ir atrasti neeiliškumą ir aiškiai matyti kalbų sistemų skirtumus.

Kuriant tokius didelius struktūrinius resursus kaip WordNet dažnai susiduriama su visa virtinė problemų:

- Vienas ir tas pats reikškinys yra aprašomas keliomis, vienomis nuo kitų nepriklausomis koncepcijos (sinsetų);
- Kokybinių kelių problema, kada kiekvienas santykis panašus į teisingą, o ilgesnis kelias sudaro „keistus" santykius.
- Kada aukščiau esanti koncepcija dalinai charakterizuoja žemiau esančią. Dažnai tai yra susiję su sumaišymu koncepcijų-tipų ir koncepcijų-vaidmenų.

  Taip, pavyzdžiui, N.Guarino [24]kritikuoja santykius WordNet'e. Žmogus yra visada būtybė, bet jis (ji) atlieka agento vaidmeni tik kai kuriose situacijose. Ta pati problema būna ir su obuolio pavyzdžiu – obuolys tai vaisius (augalo vaisius) ir tuo pačiu tai maistas. Problema tame, kad žmogus ir obuolys tai prototipai, o agentas ir maistas – vaidmenys.

4.5. Lingvistinė ontologija WordNet. Tobulinimas

Tobulinant WordNet ontologiją galima taikyti šiuos metodus:

[23] Pagal „Using WordNet for Text Retrieval" - Ellen M. Vorhees

[24] Nicola Guarino – formalių ontologijų ir informacinių sistemų tyrėjas.

- Norint sunormalizuoti koncepcijų-tipų ir koncepcijų-vaidmenų sąvokas, koncepcijas-tipus reikia patalpinti žemiau aukščiau koncepcijų-vaidmenų. Radikalesnis būdas – atskirti tipų ir vaidmenų hierarchijas. Daugelyje lingvistinių ontologijų vaidmenys ir tipai skiriasi savybių paveldėjime, bet WordNet neskiria šių dviejų tipų ir talpina juos į tas pačia hierarchijas.

- Reikia minimizuoti vieno žodžio reikšmių skaičių, kitaip tariant, reikšmių išplėtimas turi būti taikomas tik svarbioms užklausos koncepcijoms. Tai galima padaryti pavyzdžiui, nustatant dokumento, kuriame yra tam tikras žodis, svarbą: žodžiai, turintys daugiau nei N reikšmių, negali dalyvauti tolesniame užklausos vystyme.

- Tam, kad sumodeliuoti daugiareiškmumo problemos sprendimą, galima taikyti tokį būdą: užklausa yra išplėčiama tik tais žodžiais, kurių reikšmės susietos ne mažiau, nei su dvejais užklausos žodžiais. Tokiu būdu užklausos rezultatai bus tikslesni.

Šie WordNet tobulinimo metodai kolkas yra tik teorinėje stadijoje, kadangi dėl ontologijos masyvumo, jų realizacijai reikia daug žmonių ir dar daugiau laiko.

4.6. Reliacinių duomenų bazių integracija

Duomenų bazes mes galime apibrėžti kaip suderintų, tarpusavio surištų duomenų kolekciją. Duomenys turi paslėptą (viduje) reikšmę. Duomenų bazės panašios į žinių bazes, jos taip pat naudojamos aprašyti kai kuriai dalykinia sričiai su tikslu saugoti, apdoroti ir turėti prieigą prie norimos informacijos. Bet yra ir skirtumų: duomenų bazės turi (ir gali apdoroti) didesnius masyvus paprastos informacijos. Tuo tarpu žinių bazėse dažniausiai yra saugoma mažesnė informacijos apimtis, bet ji turi sudetingesnę struktūrą, kas padeda naudoti loginės išvesties galimybes ir gauti tokius pareiškimus, kurie nebuvo taip aiškiai įvedami.

- Kiekvienai žinių bazei galima taikyti 3 operacijas:
- Apibrėžtis (define);
- Pasakyti (tell);
- Paklausti(ask).

Kiekviena operacija gali naudoti vieną arba daugiau kalbų, pavyzdžiui, schemų aprašomoji kalba ir apribojimai, atsinaujinimų kalba, užklausų kalba, atsakymų kalba.

Dar iki duomenų bazės sukurimo, atsiranda būtinybė aprašyti dalykinę sritį tokia kalba, kad aprašymas būtų suprantamas tiek paprastiems vartotojams, tiek ir kūrėjams. Aprašymas realizuojamas pakankamai aukšto lygio kalbos pagrindu. Duomenų bazės srityje ta kalba yra ER-diagramos (ER-modelis). Remiantis ER-modeliu aplinkinis pasaulis vaizduojamas kaip esmių, susijusių su n-vietiniais santykiais ir aprašytais atributais, rinkinys. Iš tokio semantinio modulio yra kuriama loginė schema, aprašanti duomenų bazių struktūrą, duomenų tipus, santykius ir aprobojimus. Dažniausiai naudojama loginei schemai aprašyti yra reliacinis duomenų modelis. Realiaciniame modelyje duomenys saugomi lentelėse (santykiuose), turinčiose eilutes (kortežus), kurios susidaro iš langelių su paprastų duomenų tipų reikšmėmis (datos, skaičiai ir t.t.). Todėl aprašant loginę schemą reikia aprašyt lentelių pavadinimus, stulpelių pavadinimus bei atitinkančius duomenų tipus. Taipogi kiekvienoje lentelėje turi būt apibrėžtas raktas, unikaliai identifikuojantis lentelės eilutę.

Duomenų integracija – fundamentali problema, susidariusi per paskutinius dešimtmečius. Integracijos tikslas – sukurti vieną interfeisą skirtingiems šaltiniams. Integracija turi panaikinti vartotojų būtinybę ieškoti relevantinius duomenų šaltinius, atskirai su jais dirbti, rūšiuoti, rikiuoti ir pan. Integracinės sistemos projektavimas – labai sunkus uždavinys.

Jai spręsti taikomi 3 būdai:

1) Federacinės duomenų bazės – saugo vieną ir tą pačią informaciją, periodiškai sinchronizuojant savo būsenas. Joms reikia $O(n^2)$ ryšių;
2) Centralizuota saugykla – duomenys iš skirtingų šaltinių periodiškai kopijuojami į saugyklą. Tam reikia O(n) ryšių.
3) Mediatorių[25] technologija – pati efektyviausia, turi vieną interfeisą, galinti pasiekti visas duomenų bazes.

[25] Mediatoriai – angl. mediators, wrappers

5. Ontologijų apjungimas

5.1. Užduoties formulavimas

Ontologijų apjungimo uždavinys formuluojamas taip:

Duotos dvi teisingos ontologijos, iš jų reikia sukurti trečią teisingą ontologiją, kuri atvaizduoja įeinančių ontologijų koncepcijas, papildomus apribojimus bei sąryšius, jeigu jie reikalingi.

Tai gali būti padaryta rankiniu būdu tyrinėjant dvi ontologijas, aptinkant sinonimus, nustačius ir pašalinus prieštaravimus (nesutapimus) ir sukuriant naują, trečią, ontologiją. Bet žinių valdymo sistemose šis procesas turi būti kuo labiau automatizuotas.

Bet kokiom dviem ontologijom egzistuoja begalė būdų jas apjungti, tuo pačiu nesukuriant naujų prieštaravimų. Bendrai paėmus, beveik kiekviena koncepcija vienam žodyne gali būti sujungta su bet kuria kita koncepcija kitame žodyne. Dėl šios priežasties ontologijos dažniausiai negali būti apjungiamos be kai kurių apribojimų ir projektuotojo ir/arba sistemos administratoriaus užuominų (patarimų).

Aprašytas metodas reikalauja nedidelio rinkinio natūralių, savaime suprantamų, apribojimų ir projektuotojo arba sisteminio administratoriaus užuominų tam, kad būtų lengviau organizuoti ontologijas. Turint dvi ontologijas, kurios pilnai atitinka šias taisykles, toks metodas automatiškai sukuria pilną apjungtą ontologiją. Apjungimo metodas randa konfliktus tarp apibrėžimų tuo pačiu garantuojat, kad sukurta ontologija bus teisinga arba bus atmesta dėl loginės klaidos.

Ontologija neretai yra interpretuojama kaip grafų klasė. Kiekviena koncepcija atlieka viršūnės vaidmenį, o įtraukimo ryšiai (ryšiai klasė-poklasė) yra lankai. Klasė, neturinti bazinės klasės, yra grafo viršūnė; klasė gali turėti bet kokį kiekį bazinių klasių (t.y. palaikomas daugiakartinis paveldimumas). Ontologija gali naudoti (t.y. importuoti) į savo apibrėžimus elementus iš kitų ontologijų. Kada yra importuotos koncepcijos, tai duotasis metodas vykdomas rekursyviai, kiekvienai importuotai ontologijai. Šiame metode turima omenyje, kad ontologija neturi nuorodų į kitas išorines ontologijas. Tai galima pasiekti pilnai išplėčiant visas ontologijoje importuojamas koncepcijas iki šio metodo taikymo. Toliau bus nagrinėjamos ontologijos, esančios vienu koncepcijų grafu su viena šaknine viršūne.

Ontologijų baziniu apribojimu yra tai, kad jos turi būti organizuotos šakninių grafų arba medžių miško pavidalu. Tai draudžia ciklinius grafus, t.y. viršūnes, kurios yra tėvai ir tuo pačiu protėviai kitoms viršūnėms. Ontologijų apribojimas medžiais leidžia visiškai manipuliuoti ontologijomis, naudojant operacijas nedideliuose šakninių viršūnių rinkiniuose. Siūlomi apribojimai yra natūralūs ypač kai ontologija yra projektuojama siekiant išplėsti kitą, aukštesnio lygio, ontologiją.

Apjungta ontologija yra tikrinama deskriptyvios logikos algoritmo, kuris taikomas norint patikrinti atskiras ontologijas, pagalba. Papildant surištų tarpusavyje koncepcijų hierarchijas, taip pat galimi ir loginiai sąryšiai tarp įvairių koncepcijų ontologijose. Pavyzdžiui, dvi koncepcijos skirtingose ontologijose gali būti sinonimais, arba ne šakninė klasė gali būti kurios nors klasės kitoje ontologijoje poklase. Retkarčiais yra labai svarbu apibrėžti, kad dvi koncepcijos yra skirtingos ir nepersidengia tarpusavyje. Tas apibrėžimas gali būti neišvengiamu, nes dvi panašios, bet skirtingos koncepcijos gali neturėti kokių nors jas atskirinčių charakteristikų ontologijos apibrėžime. Šie santykiai gali būti aiškiai aprašyti pareiškimų forma (vadinamomis aksiomomis loginiame programacime), kurie gali būti išreikšti užuominų pavidalu apjungiant ontologijas (*disjoint* (c_i , c_j) deskriptyvioje logikoje). Aksiomos yra įtraukiamos žinių bazės deskriptyvios logikos modelio pareiškimų pavidalu tada, kai yra tikrinamas apjungtos ontologijos teisingumas.
Konceptualiai aksiomos panašios į globalinius apribojimus, bet yra aprašomos ontologijų kalbos pareiškimais, kurie yra apdorojami taip pat kaip ir koncepcijų ir santykių apibrėžimai.

Aksiomos gali apibrėžti santykius tarp koncepcijų vienoje ontologijoje arba tarp koncepcijų skirtingose ontologijose. Pirmame atvejyje jos gali būti įterptos į ontologiją (kaip pagalbiniai santykiai). Antrame atvejyje jos gali būti atskiroje ontologijoje, kuri importuos ontologijas į kurias nurodo aksiomos. Aksiomos teikia būdą aiškiai valdyti ir netgi kitaip nustatyti jau taikomą grafų apjungimo metodą . Šis būdas turi būti naudojamas tada, kai yra tam tikra specifinė žinių bazė, kuri apibrėžia tarpontologinių santykių sinonimus.

5.2. Apjungimo metodas

Iš pradžių reikia įvesti įprastinės ontologijos paprastą apibrėžimą:

Ontologija tai nuosekli pora, O=(C, R, P_c), kur C – viršūnių rinkinys, R – santykių rinkinys, P_c – aibė asimetrinių, tranzytivių[26], nerefleksyvių[27] binarių santykių, kuri yra dalinės tvarkos aibėje C santykis ir kuri apibrėžia santykius tarp C koncepcijų ir R santykių.

Toliau apjungmo metodui paaiškinti taikysime paprastesnį ontologijos apibrėžimą, paremtą grafo modeliu:

Ontologija tai nuosekli pora, O=(C, R), kur C – viršūnių rinkinys, o R – briaunų rinkinys. Kiekviena koncepcija ontologijoje vaizduojama viršūne C aibėje.

Nukreiptas lankas apibrėžia įteptas koncepcijas arba binarinius santykius tarp jų. Šakninė viršūnė ne turi įeinančių lankų. Viršūnės ir lankai O_1 ir O_2 ontologijų turi unikalius pavadinimus. Teisinga ontologija tai tokia ontologija, kuriai bus įrodomas jos suderinamumas, pavyzdžiui, taikant deskriptyvią logiką.

Siūlomo metodo tikslas yra naujos ontologijos sukūrimas, kuri yra dviejų kitų ontologijų apjungimo rezultatas ir turi šų dviejų ontologijų koncepcijų sąryšius. Įvesties duomenys yra dvi teisingos ontologijos O_1 ir O_2: $O_1=(C_1, R_1)$ ir $O_1=(C_2, R_2)$, o taip pat užuominų rinkinys (santykių): aibė L nuoseklų porų (c_1, c_2), kur $c_1 \in^{28} C_1$ ir $c_2 \in C_2$. Reikalaujama, kad nauja ontologija O_{1+2} būtų teisinga, su visomis ontologijų O_1 ir O_2 koncepcijomis ir santykiais, nustatytais užuominose. Jeigu nauja ontologija turi prieštaravimų, tada metodo rezultatas yra klaidingas.

Apjungimo metodas susideda iš 3-jų žingsių:

- Sujungimo;
- Konkrečių šakninių viršūnių surišimo;
- Naujos ontologijos teisingumo patikrinimo (patikrinimas ar naujai sukurta ontologija neturi prieštaravimų).

[26] Pereinamas, turinti nuorodą į papildymą, kai viena savybė apibrėžia kitą ir t.t.

[27] Elementas nesiejamas pats su savimi

[28] Priklausomumas. Pvz.: $c_1 \in^{28} C_1$ – c_1 yra elementas aibės C_1

1 žingsnis: sujungimas O_1 ir O_2 norint suformuoti O_{1+2}

Pagal apibrėžimą: $O_{1+2} = O_1 \, U^{29} \, O_2$, kur $O_1 \, U \, O_2 = (C_1 \, U \, C_2, R_1 \, U \, R_2)$. Apjungimas visada gali būti išskaičiuotas, bet rezultatas realiai gali turėti prieštaravimų. Norint $O_1 \, U \, O_2$ ontologija gali būti patikrinama teisingumui nustatyti jau šitame žingnyje.

2 žingsnys: Surišimas susietų hierarchijų, aprašytų santykių rinkiniu

Kiekvienai porai $(c_1, c_2) \in L$ ontologijoje O_{1+2} įtraukiamas teiginys subClassOf(c_1,c_2), t.y. grafe sukuriamas naujas lankas. Po šito žingsnio sudėtinė ontologija apibrėžiama taip: $O_{1+2} = (C_1 \, U \, C_2, R_1 \, U \, R_2 \, U \, L)$. Padarius prielaidą, kad šių santykių elementai yra teisingi (t.y. kiekvienas $c_1 \in C_1$ ir $c_2 \in C_2$), ši operacija visada gali būti įvykdyta, bet galimas variantas, kad santykių panaudojimas gali sukurti prieštaravimus. Vėl gi šitame žingsnyje gali būti patikrintas ontologijos $O_1 \, U \, O_2$ teisingumas.

3 žingsnys: ontologijos O_{1+2} teisingumo patikrinimas

Jeigu ontologija O_{1+2} yra teisinga, tai „Sėkmė“ (tikslas įvykdytas), kitu atveju – „nesėkmė“. Algoritmas baigtas.

Apjungiant dvi ontologijas gali susidaryti papildomi loginiai apirbojimai arba tarpusavio santykiai, kurie naudojami apjuntoje ontologijoje. Retkarčiais reikia apibrėžti tarpusavio santykius tarp koncepcijų dvejose ontologijose. Šie santykiai turi būti įtraukti į ontologiją kaip papildomos briaunos. Tai realizuojama sukurs nedidelę, fragmentinę ontologiją, kuri turi aksiomų, įtraukiamų į O_{1+2}, naudojant anksčiau aprašytą metodą. Bendras apjungimo procesas gali turėti kelių ontologijų apjungimą tam, kad sukurti naują, rezultatinę, ontologiją.

5.3. Apjungimo operacijų savybės

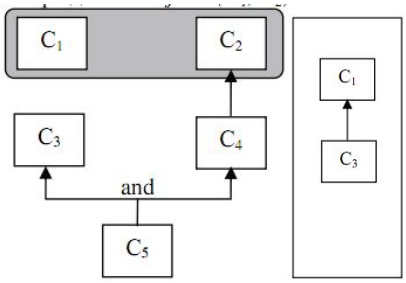
Ontologijų apjungimo metodas apibrėžia operaciją teisingų ontologijų visumoje. Ši operacija turi kelias paprastas savybes, kurios kyla iš deskriptyvios logikos monotoniškumo apibrėžimo.

[29] Apjungimas. Pvz.: A U B – elementų aibė priklausanti A arba B.

5.3.1. Ontologijų apjungimo operacija nėra uždara

Ontologijų apjungimo operacija nėra uždara. Jeigu valid(O_i) parodo, kad ontologija O_i yra teisinga, tai valid(O_1) ∧ valid(O_2) nereiškia, kad valid(O_{1+2}). Tai teisinga, nes O_{1+2} gali turėti sąryšius tarp O_1 ir O_2, kurios sukuria prieštaravimus, netaikomus atskirom ontologijom O_1 ir O_2. Apžvelkime dviejų paprastų ontologijų apjungimo apvyzdį (*7pvksl.*):

Teiginys disjoint (C_1, C_2)



Ontologija O_1 Ontologija O_2

7 pvksl. Dvi paprastos ontologijos, kiekviena iš kurių yra teisinga[30]

O1 ontologija aprašo 5 koncepcijas ir santykius tarp jų. Koncepcija C_4 apibrėžta kaip darinys iš C_2 koncepcijos, o C_5 apibrėžta kaip darinys iš C_3 ir C_4 (daugialypis paveldimumas). Be to, yra apirbrėžtas apribojimas C_1 ir C_1: C_1 negali būti C_2. Šitas apribojimas reiškia, kad objektas gali būti arba C_1 arba C_1, bet negali būti ir C_1 ir C_2 kartu. Ontologija O_1 yra teisinga, t.y. neturi prieštaravimų.

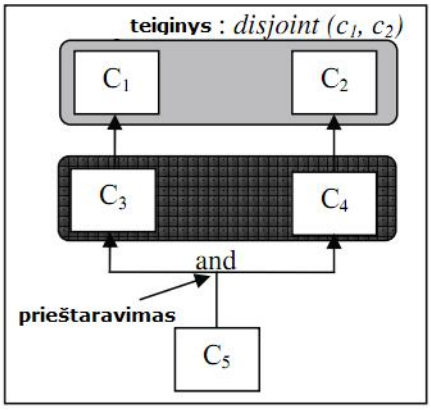
Ontologija O_2 apibrėžia santykius tarp C_1 ir C_3, kurios yra tos pačios koncepcijos kaip ir C_1 bei C_3 O_1 ontologijoje. O_2 ontologijoje koncepcija C_3 yra darinys iš C_1, t.y. C_3 yra koncepcija C_1. Ontologija O_2 yra teisinga ontologija.

Kada šitos dvi ontologijos yra apjungiamos, jos formuoja trečią onologiją O_{1+2} (*8 pvksl.*). Abidvi pradinės ontologijos apibrėžia santykius tarp koncepcijų C_1 ir C_3, todėl ontologija O_{1+2} turi visus šiuos santykius.

Logiškai analizuojant ontologiją O_{1+2} randamas prieštaravimas. Įskaitant, kad C_1 ir C_2 apibrėžtos kaip nesusikertančios, iš paveldėjimo santykių seka, kad C_3 ir C_4 taip pat turi būti

[30] Šaltinis: http://www.duskyrobin.com/tpu/2006-07-00031.pdf p.140

nesusikertančiomis. Koncepcija C_5 buvo apibrėžta kaip darinys iš C_3 ir C_4, bet tai neįmanoma įvykdyti, nes yra nustatytas teiginys disjoint(C_3, C_4). Todėl apjungta ontologija yra neteisinga.



8 pvksl. Prieštaravimo atsiradimas po dviejų teisingų ontologijų apjungimo[31]

Šis pavyzdys rodo, kad apjungiant dvi teisingas ontologijas nebūtinai yra gaunamas geras (teisingas) rezultatas, t.y. apjungimo operacija nėra uždara. Prieštaravimas atsiranda dėl to, kad pradinės ontologijos turi prieštaraujančių teiginių (bent jau vienai koncepcijai). Logika aptinka prieštaravimą netgi tada, kai nei viena iš pradinių ontologijų neturi savyje prieštaravimų, turinčių C_5 koncepciją.

5.3.2. Apjungimo operacija nėra komutatyvi

Nors dvi sudėtinės ontologijos ne visada yra teisingos, bet kada santykišų suma $O_1 + O_2$ yra teisinga, tai suma $O_2 + O_1$ taip pat turi būti teisinga, t.y. valid(O_{1+2}) $\Leftrightarrow$[32] valid(O_{2+1}). Intuityviai aišku, kad O_{1+2} yra tas pats rinkinys kaip ir O_{2+1}, todėl jie abudu arba teisingi arba abudu neteisingi. Formaliai tai yra išsiveda iš monotoniškumo apibrėžimų deskriptyvioje logikoje (žr. Priedus: „Apjungimo operacija nėra komutatyvi“) .

5.3.3. Apjungimo operacija yra asociatyvi

Apjungimo operacija taip pat turi asociatyvių savybių, t.y. valid((O_{1+2})+O_3) $\Leftrightarrow$ valid(O_1+(O_{2+3})). Šitas teiginys išvedamas iš įrodymo aprašyto aukščiau. Panagrinėkime tris ontologijas O_1=(C_1, R_1), O_2=(C_2, R_2), O_3=(C_3,R_3) kartu su ryšiais L_{1+2}, L_{2+3}, L_{1+3}. Trys

[31] Šaltinis: http://www.duskyrobin.com/tpu/2006-07-00031.pdf p.140

[32] Lygiavertiškumas. Pvz.: A $\Leftrightarrow$ B rodo, kad A yra tiesa tada ir tik tada, kad yra B yra irgi tiesa.

ontologijos yra apjungiamos dvejais žingsniais (žr. Priedus: „Apjungimo operacija yra asociatyvi“).

5.4. Apjungimo operacija nėra tranzityvi

Galima pamatyti, kad iš neuždarumo savybės, apjungimo operacija nėra uždara. T.y. iš teisingumo valid(O_{1+2}) ∧ valid(O_{2+3}) nėra išvedama valid(O_{1+3}).

Aprašytas dviejų ontologijų apjungimo metodas leidžia automatizuoti bendros ontologijos modelio formavimą, ir palaikyti bendros apjungtos ontologijos loginį suderinamumą, kas yra labai svarbu dirbant su žinių organizacijos modeliu. Taikomosios valdymo sistemos gali naudoti bendrą ontologiją semantinio Web-portalo rėmuose tam, kad atlikti įvairias užduotis valdant žinias – kategorizacijos, rekomendacijų, paieškos (žr. Priedus – paveiksliukas NR. 14). Ontologijų apjungimas įeina į tokių sistemų kaip OntoMerge, Chimaera, OBSERVER, OntoMorph ir PROMPT funkcijas.

5.4.1. Ontologijų apjungimas taikant vaidmens modeliavimo būdą

Šiame skyriuje bandysime parodyti kaip galima pusiau automatiškai apjungti kelias ontologijas norint gauti vieną bendrą. Ontologijų apjungimas grįstas vaidmens modeliavimu būdu dažnai yra lengvesnis negu klasių apjungimas, kadangi vaidmens tipai apima tik vieną individo rūpestį, tuo tarpu klasės – kelis. Be to vaidmens modeliai paaiškina kuriems klasės apribojimams ir santykiams jie priklauso.

Ontologijų kūrimo iš komponentų problematika:

Tikslas: sukurti naują ontologiją iš jau egzistuojančių blokų;

Koks yra pakartotinio panaudojimo vienetas?

OWL import teiginys:

- Viena ontologija importuoja kitą
- Nėra dalinio pakartotinio panaudojimo

Mūsų tikslas apjungti dvi vaidmenimis grįstas ontologijas, apimančias picos ir pastos dalykines sritis. Jos yra sumodeliuotos tokiu būdu, kad visi atskirų elementų

bendradarbiavimai yra abstraktūs ir tarp klasių neegzistuoja savybės. Pagrindinė idėja yra tame, kad apjungti vaidmenų modelius iš abiejų ontologijų ir po to apjungti klases, pagrįstas vaidmenų tipų santykiais tam, kad sukurti naują ontologiją. *18 pvksl*. Yra iliustruoti skirtingi įvairių vaidmenų modelių santykiai. Vienas vaidmens modelis (Origin) priklauso abejoms ontologijoms, kas yra idealus atvejis norint sujungti šias ontologijas. Skonio (Taste) ir Kvapo (Flavor) vaidmens modeliai aprašo iš esmės tą patį rezultatą, bet skirtingomis raiškos priemonėmis. Taigi jie turėtų būti traktuojami kaip lygiaverčiai. Kiti vaidmens modeliai Maistas (Meal) ir Gaminys (Product) nėra tarpusavyje susiję (žr. Priedus- paveiksliukas NR. 15).

Brūkšninė linija rodo vaidmens tipų ekvivalentiškumą. Šita informacija jau gali būti panaudota norint sujungti ontologijų klases. Atviras vaidmens tipas Šalis (Country) iš pastos ontologijos yra aiškus rėmas klasei Šalis Picos ontologijoje, nes jis turi identišką vaidmens tipą. Kvapas (Flavor) ir Skonis (Taste) taip pat turi kiekvienas savo vaidmens tipą. Kadangi šios savybės yra ekvivalentiškos vaidmenų modelių palyginime, tai ir jas turinčios klasės irgi yra ekvivalentiškos. Priešingai clases Pica (Pizza) ir Pasta (Pasta) tik iš dalies sutampa. Abidvi kiek pica tiek pasta gali turėti kilmę (jos dalinasi Dalyko (Thing) vaidmens tipu iš Origin vaidmens modelio). Be to, vaidmens tipai Degustavimas (Tasting) ir Valgymas (Food) yra ekvivalenčiai vaidmenų modelio palyginime. Tačiau pica yra maistas (Meal), tuo tarpu pasta yra produktas (Product). Kitaip tariant, likę vaidmenų tipai Maistas ir Productas nesutampa. Tam, kad sumodeliuoti šią situaciją, sistema turi pasiūlyti bendrą supeklasę Picai ir Pastai. Šio sujungimo rezultato procesas parodytas 19 pvksl., kur ekvivalentiškos klasės yra sujungtos į vieną klasė ir ta nauja klasė yra įdiegta kaip Picos ir Pastos superklasė. Suteikimas jai prasmingo pavadinimo negali būti automatizuotas(žr. Priedus- paveiksliukas NR. 16).

Vaidmenų modeliu grįstas apjungimas susidaro iš dviejų pagrindinių žingsnių:

1) Vaidmens modelių palyginimo:

   Visų pirma, vaidmenų modeliai pradinių ontologijų turi būti palyginti. Šitame žingsnyje galimi du atvejai:

   - Dvi ontologijos naudoja tą patį vaidmens modelį – šituo atveju jos yra ekvivalentiškos;

- Dvi ontologijos naudoja skirtingus vaidmens modelius – šituo atveju reikia atlikti palyginimą.

2) Klasių sujungimo:

Galimi 4 etapai:

- Visi vaidmens tipai dviejų klasių yra ekvivalenčiai, tokiu atveju klasės yra ekvivalentiškos;
- Dvi klasės turi tiek ekvivalenčių vaidmens tipų tiek ir skirtingų. Tokiu atveju šis žingnis negali būti pilnai automatizuotas, kadangi reikia nagrinėti ar reikia apibendrinančios superklasės ir jei taip, kaip ją užvadinti;
- Klasės neturi ekvivalentiškų tipų, taigi negali būti išvis palyginamos;
- Du modelių tipai yra ekvivalenčiai, o vienas nepriklauso nei vienai iš klasių (yra atviras). Tokiu atveju šito atviro modelio elementai gali atlikti reikiamą vaidmenį.

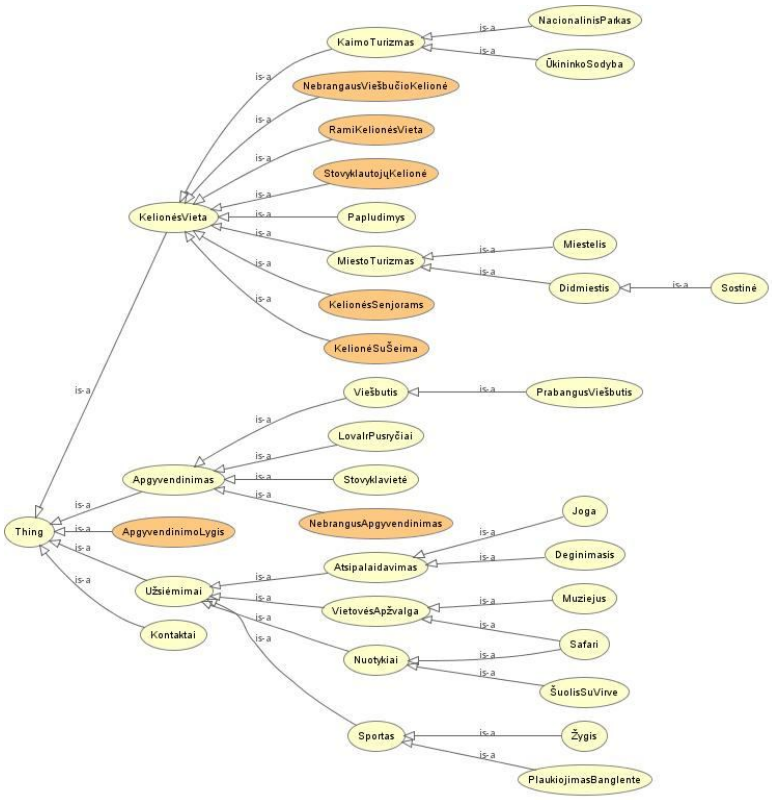
6. Projektinė dalis

6.1. Darbo aprašymas

Ontologijų analizavimui, kūrimui bei redagavimui naudojama nemokama, atviro kodo programa Protege. Naudojant jos plėtinius yra galima analizuoti ontologijas grafiškai. Ontologijos failų pagrindą sudaro XML formato kodo blokai rašomi ant RDF kodo blokų. OWL technologija yra naudojama apdoroti informacijai esančiai internete. Ji buvo sukurta tam, kad ją galėtų interpretuoti kompiuteriai (skaityti žmonėms be papildomų žinių apie struktūrą būtų labai sunku). OWL kalba yra W3C standartas.

Norint grafiškai matyti klases ir ryšius tarp jų, reikalingas OWLViz plėtinys (plug-in). Įsirašius šį papildinį, protege programoje atsiranda papildomas OWLViz peržiūros langas, tačiau jį jungiat gauname klaidos pranešimą. Norint turėti pilnai funkcionuojantį ir veikiantį grafinį įrankį, reikia įsirašyti papildomą programinį paketą GraphViz. Tada Protege programoje reikia atlikti nustatymus „File -> Preferences...“ ir ten OWLViz skirtuke nurodyti kelią iki programiniame pakete GraphViz esančio „dot.exe“ failo. Jo pagalba galime automatiškai gauti klasių tarpusavio ryšių schemas.

6.2. Kelionių ontologijos analizė



Kelionių ontologija susideda iš 35 klasių, 6 objektų būdo savybių, 4 duomenų būdo savybių ir 14 individualių objektų.

Klasė - tai tam tikro tipo objekto aprašymas, kuriame matomi galimi veiksai su juo bei to objekto duomenų struktūra.

Objekto būdo savybė – tai savybė sujungianti dvi klases joms panašumo turinčioje vietoje.

Duomenų būdo savybės – tai savybės kurios jungia tam tikros klasės instanciją su RDF žodžiais ir XML schemos duomenų tipais.

Individualus objektas – tai objektas turintis tam tikras konkrečias reikšmes.

6.2.1. Klasių analizė

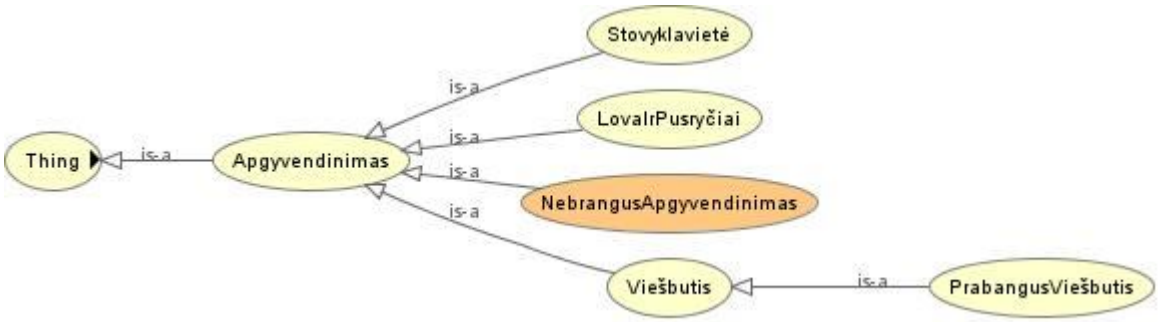
Visos kelionių ontologijos pagrindinė klasė yra „Thing“. Ją sudaro 5 subklasės:

- Apgyvendinimas.
- ApgyvendinimoLygis
- KelionėsVieta
- Kontaktai
- Užsiėmimai

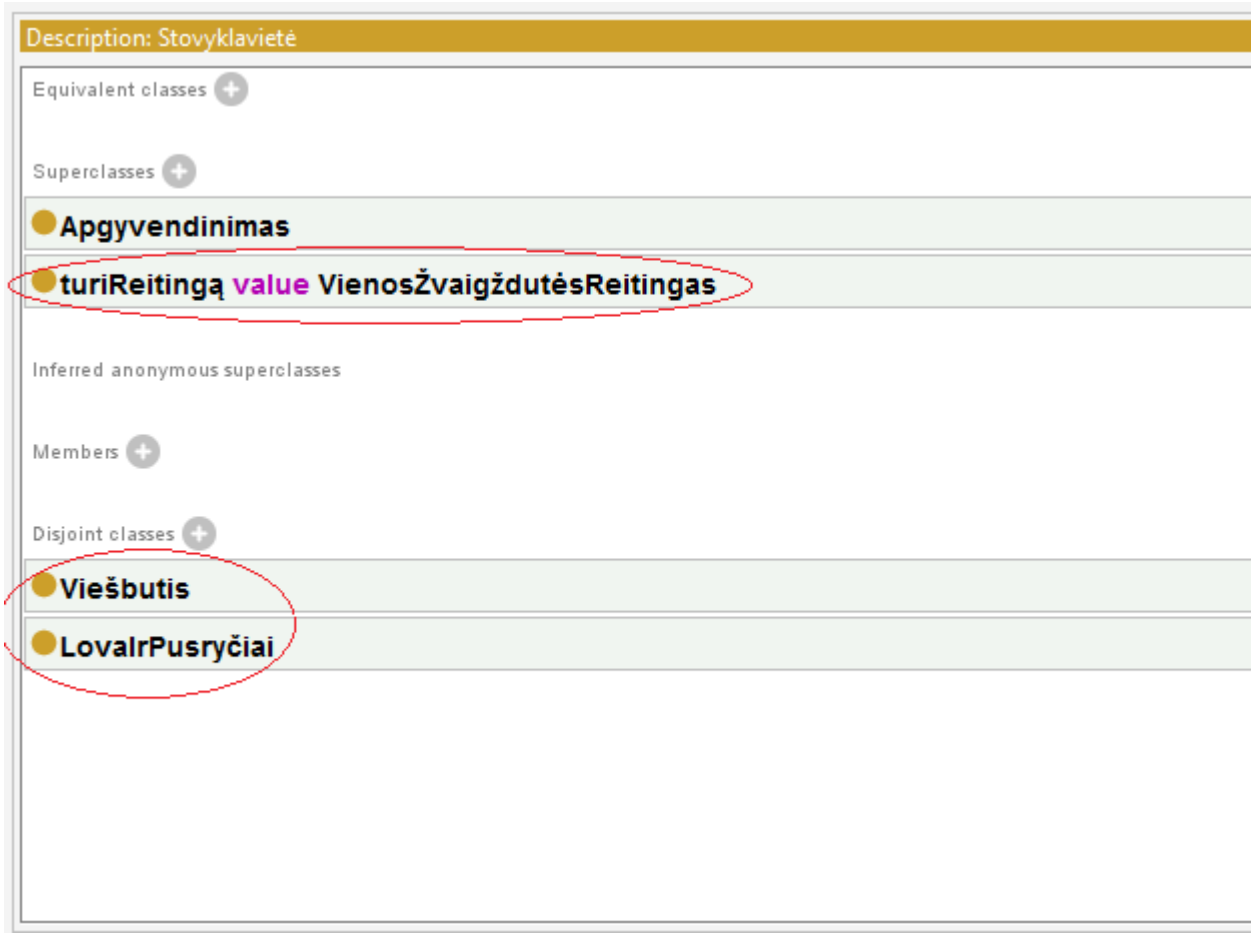
Vizualiai klasių herarchijos medis atrodytų šitaip:

Kiekvieną klasę nagrinėjant atskirai galima aprašyti kiekvienos jos savybes. Klasė „Thing“ yra visos ontologijos klasė, neturinti jokių atributų. Pirmoji jos subklasė „Apgyvendinimas“

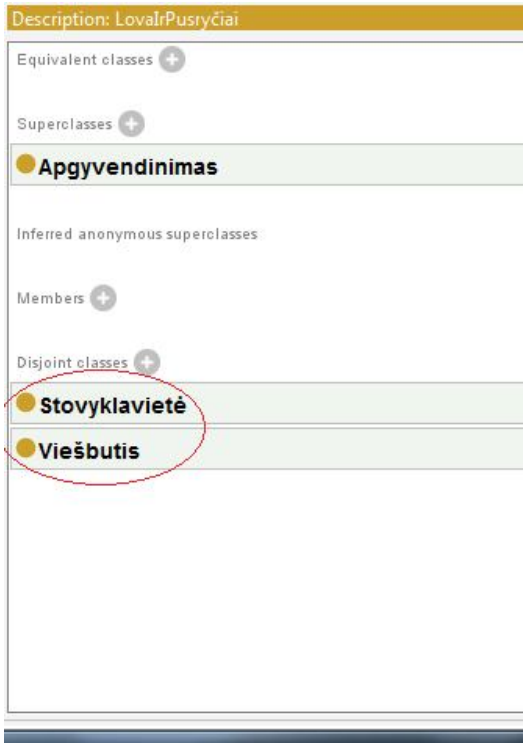
pati turi 4 subklases, bet jokių atributų ar individų ji neturi. Žvelgiant šia linkme gilyn, matome jog subklasė „Viešbutis“ turi dar vieną subklasę „PrabangusViešbutis“:



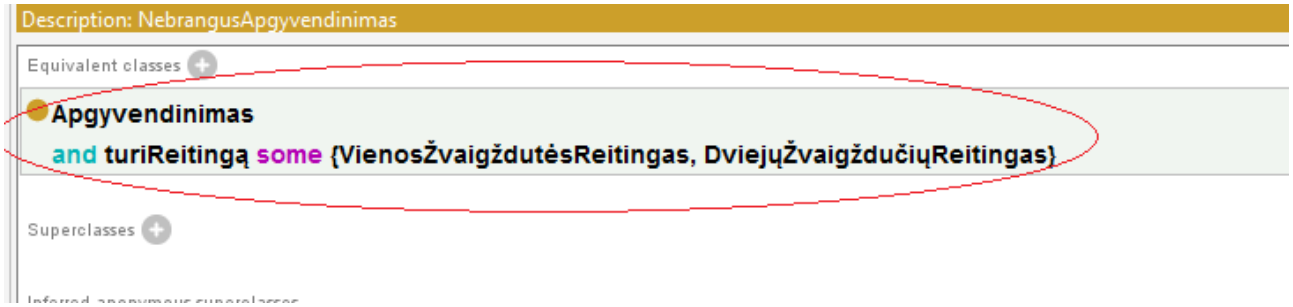
Subklasė „Stovyklavietė“ turi atributą „turiReitingą“, kurio reikšmė priskirta „VienosŽvaigždutėsReitingas“. Taip pat ši klasė negali būti sugrupuota su klase „Viešbutis“ arba su klase „LovaIrPusryčiai“:



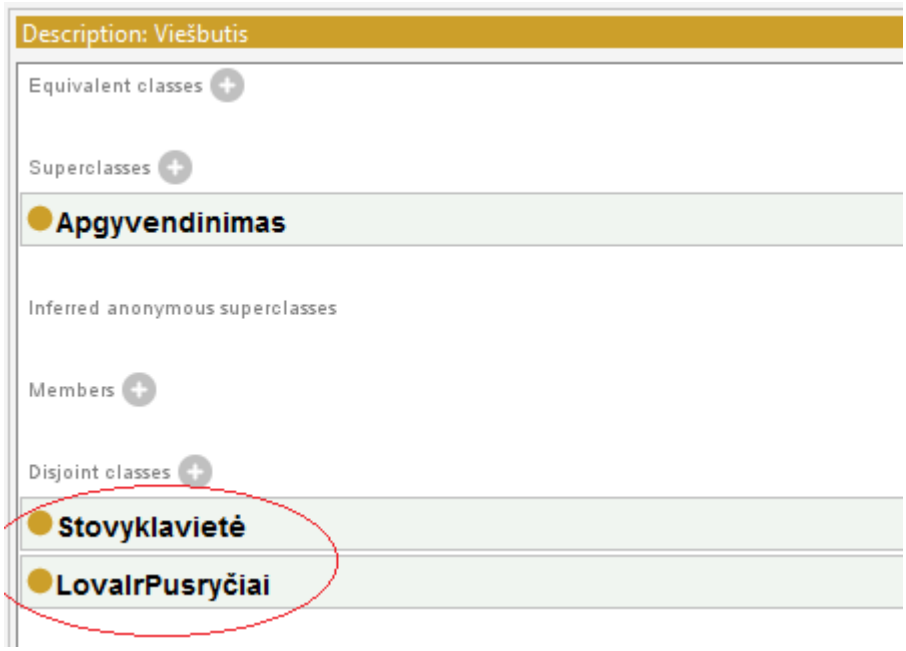
Subklasė „LovaIrPusryčiai“ negali būti jungiama su klasėmis „Stovyklavietė“ ir „Viešbutis“:



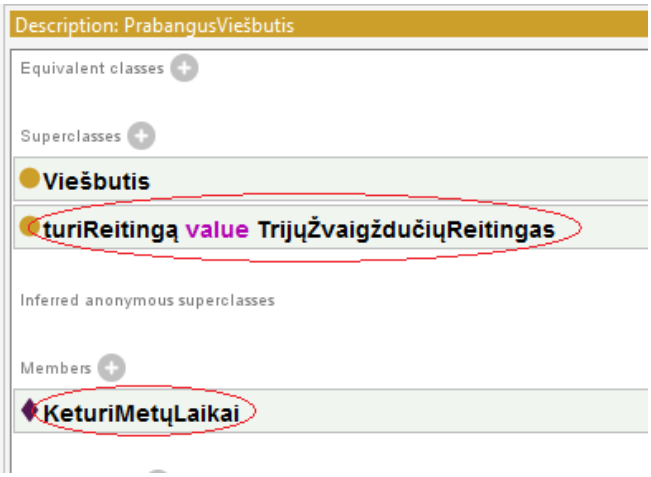
Subklasė „NebrangusApgyvendinimas“ yra lygi (ekvivalenti) klasei „Apgyvendinimas“ kuri privalo turėti atributą „turiReitingą“ kuris bus lygus arba „VienosŽvaigždutėsReitingas“ arba „DviejųŽvaigdučiųReitingas“ abiems minėtiesėms individams:

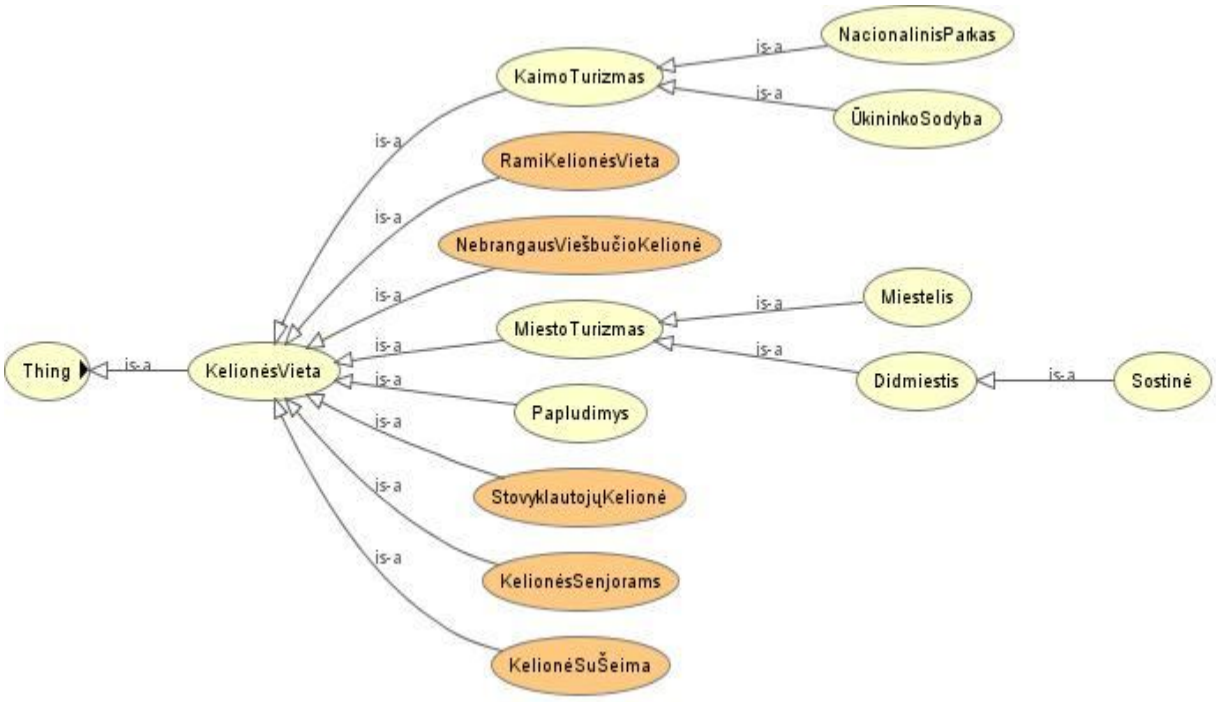


Klasė „Viešbutis“ turi vieną subklasę „PrabangusViešbutis“ ir negali būti jungiama su „Stovyklavietė“ ir „LovaIrPusryčiai“ klasėmis:



Subklasė „PrabangusViešbutis“ turi atributą „turiReitingą“ kurio reikšmė yra lygi individui „TrijųŽvaigždučiųReitingas“. Taip pat ši klasė turi individą „KeturiMetųLaikai“:



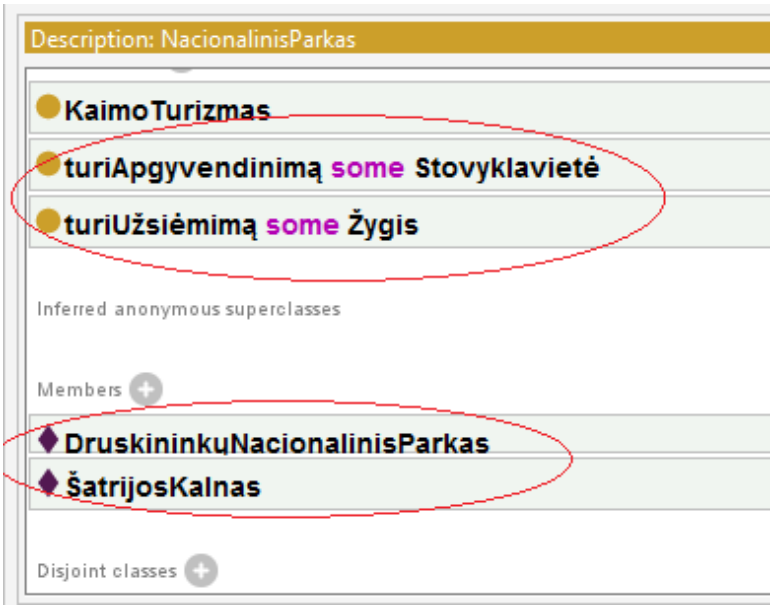


Kita tėvinės klasės „Thing“ subklasė yra „KelionėsVieta“. Ji yra labai didelė, turinti net 8 subklases, kurios savo ruožtu turi dar 5 subklases. Trumpai apžvelkime kiekvieną iš jų:

- „KaimoTurizmas“ klasė turi 2 subklases ir 2 individus – „Dargaičiai“ ir „Jūrvingė“ bei yra nejungtina su klase „MiestoTurizmas“:

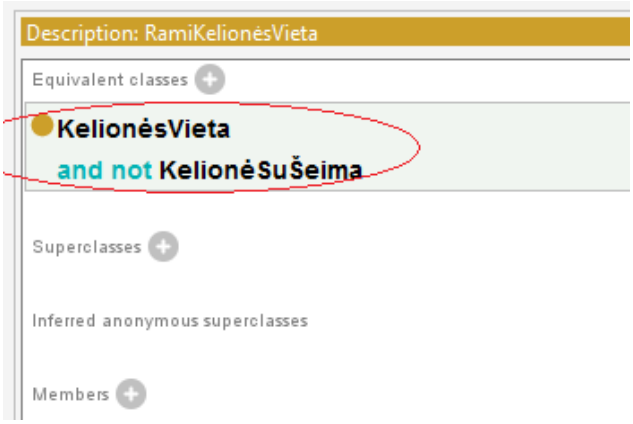
  - „NacionalinisParkas“ yra „KaimoTurizmas“ klasės subklasė ir turi 2 atributus bei 2 individus:
    - Atributą „turiApgyvendinimą“ kuris gali būti lygus „Stovyklavietė“

- Atributą „turiUžsiėmimą“ kuris gali būti lygus klasei „Žygis“
- Individą „DruskininkųNacionalinisParkas“.
- Individą „ŠatrijosKalnas“

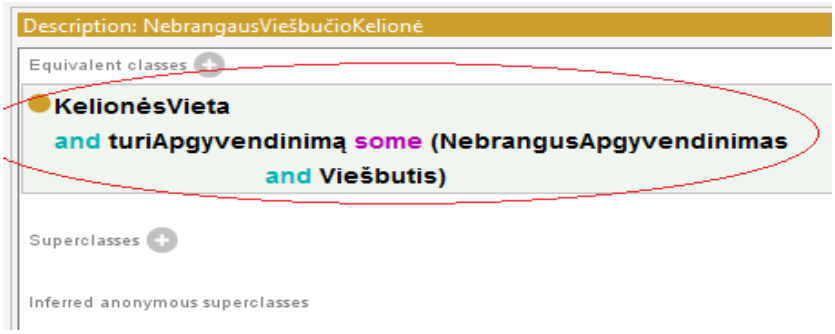


- „ŪkininkoSodyba“ klasė neturi nei atributų, nei individų.

- Klasė „RamiKelionėsVieta“ yra ekvivalenti klasei „KelionėsVieta“, tačiau negali būti ekvivalenti klasei „KelionėSuŠeima“ – jos negali būti naudojamos, kaip viena kitos pakaitalas:

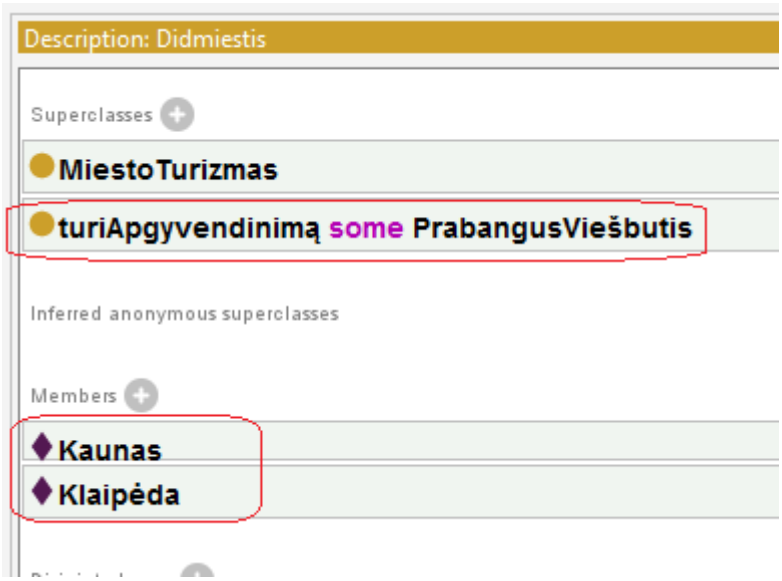


- „NebrangausViešbučioKelionė“ taip pat ekvivalenti klasei „KelionėsVieta“. Taip pat ši klasė turi atributą „turiApgyvendinimą“ kuris gali būti lygus klasei ekvivalenčiai klasėms „NebrangusApgyvendinimas“ ir „Viešbutis“:



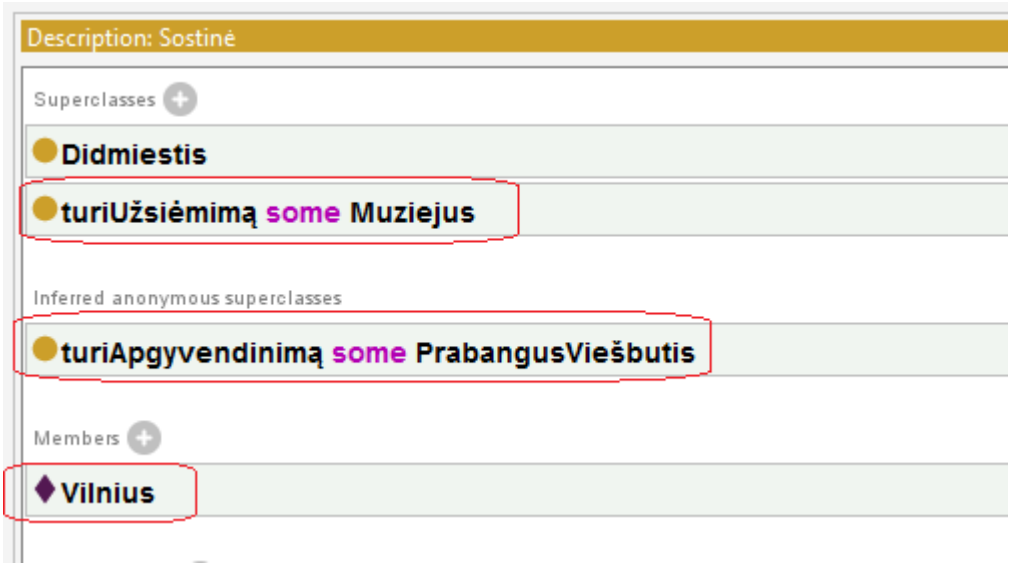
- Klasė „MiestoTurizmas“ negali būti tapatinama su „KaimoTurizmas“ klase, bei turi 2 subklases, o viena jų („Didmiestis“) – turi savo subklasę „Sostinė“:

  - „MiestoTurizmas“ subklasė „Didmiestis“ turi savo subklasę „Sostinė“. „Didmiestis“ klasės atributas „turiApgyvendinimą“ gali būti lygus „PrabangusViešbutis“ klasės instancijai, bei turi 2 individus „Kaunas“ ir „Klaipėda“:

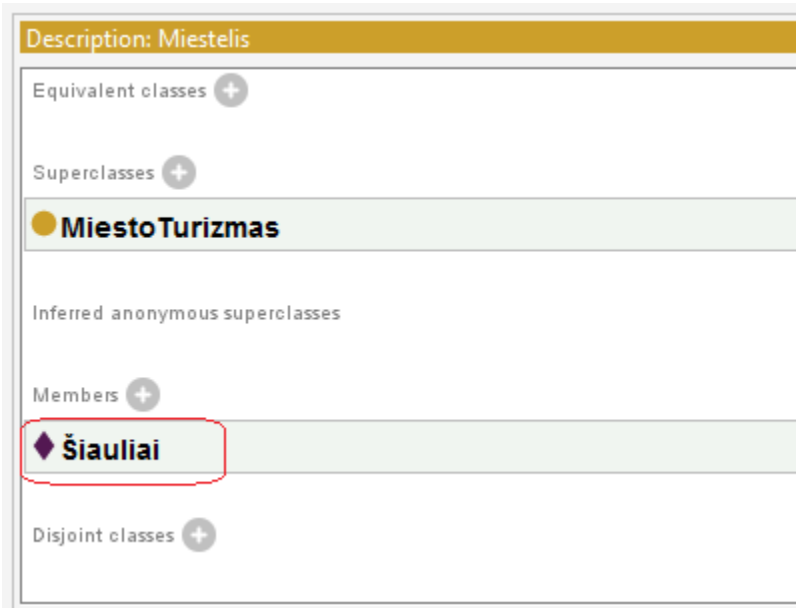


    - „Sostinė“ klasė turi atributą „turiUžsiėmimą“ kuris gali būti lygus klasei „Muziejus“. Turint omenyje, kad šios klasės superklasė yra „Didmiestis“, galime daryti išvadą, jog ši klasė gali turėti ir paveldėtą atributą „turiApgyvendinimą“ kurio reikšmė gali būti

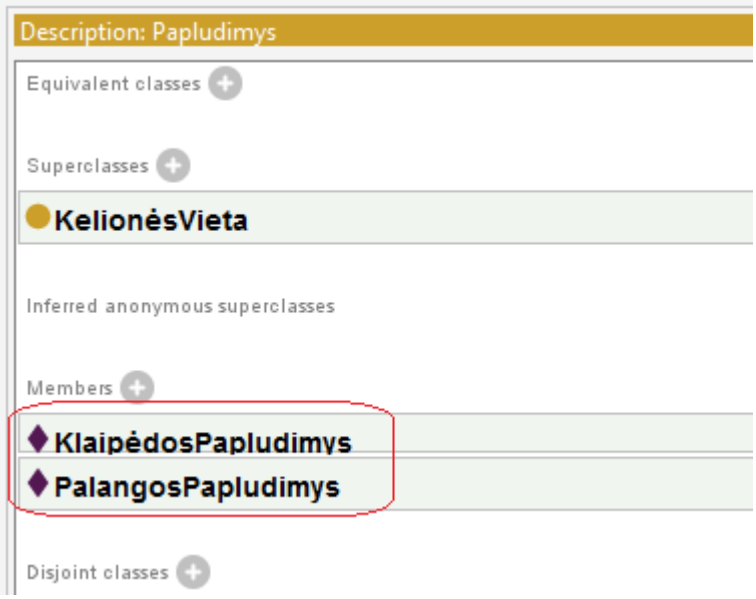
„PrabangusViešbutis“ klasės instancija. Taip pat ši klasė turi individą „Vilnius“:



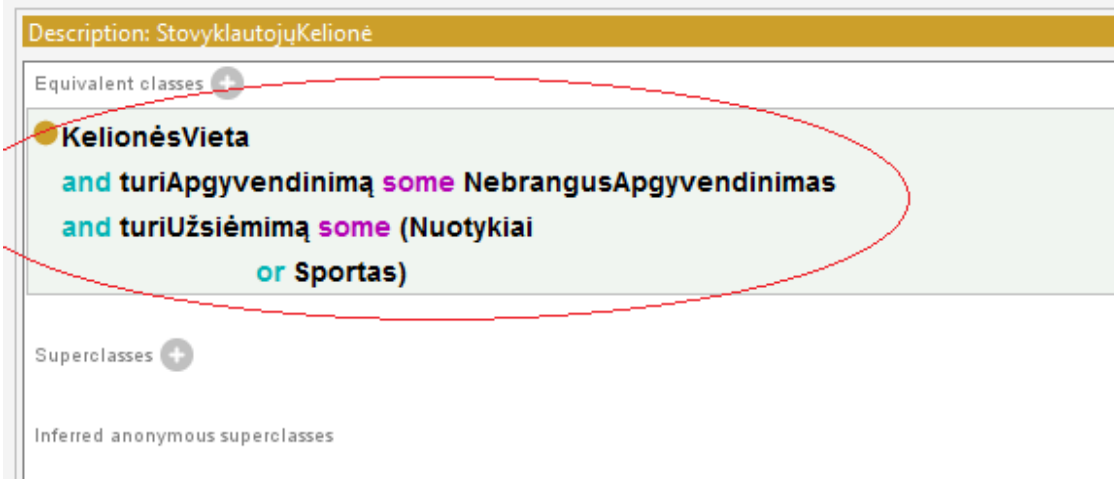
- Kita „MiestoTurizmas“ subklasė „Miestelis“ tiesiog turi vieną individą – „Šiauliai“:



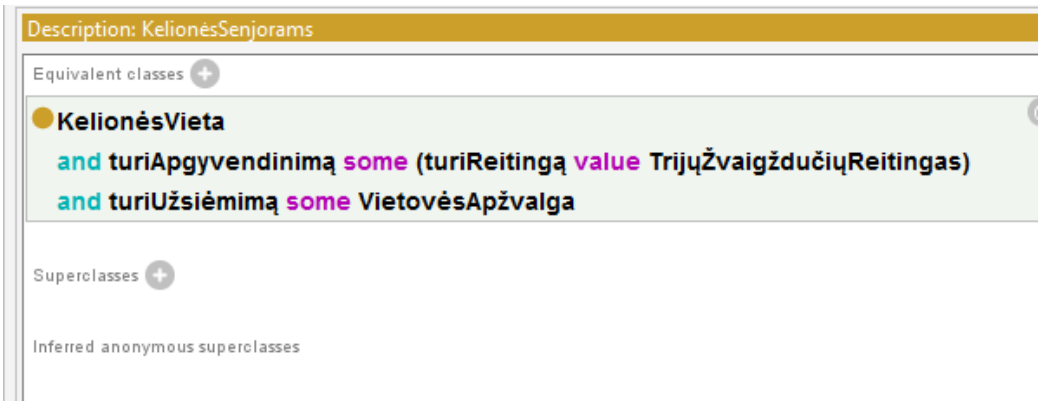
- Dar viena „KelionėsVieta“ subklasė yra klasė „Papludimys“. Ji turi 2 individus – „KlaipėdosPapludimys“ ir „PalangosPapludimys“:



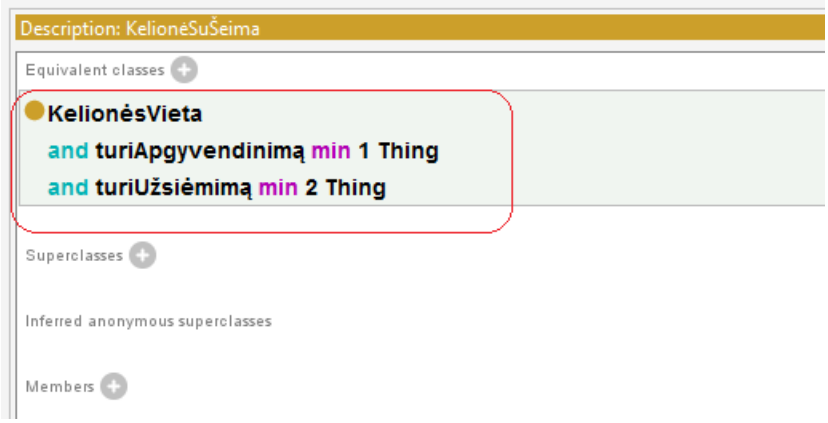
- Klasė „StovyklautojųKelionė“ yra ekvivalenti klasei „KelionėsVieta“ ir turi 2 atributus: atributą „turiApgyvendinimą“ kuris gali būti „NebrangusApgyvendinimas“ klasės instancija ir atributą „turiUžsiėmimą“ kuris gali būti arba klasės „Nuotykiai“ arba klasės „Sportas“ instancija:



- Klasė „KelionėsSenjorams“ taip pat ekvivalenti klasei „KelionėsVieta“ beti turi atributą „turiApgyvendinimą“ su reikšme, kuri savo ruožtu turi atributą „turiReitingą“, kuris yra lygus „TrijųŽvaigždučiųReitingas“ individui. Dar vienas atributas priklausantis „KelionėsSenjorams“ klasei yra „turiUžsiėmimą“ „VietovėsApžvalga“ klasės instancija:

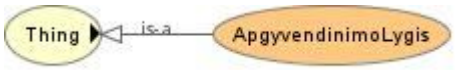


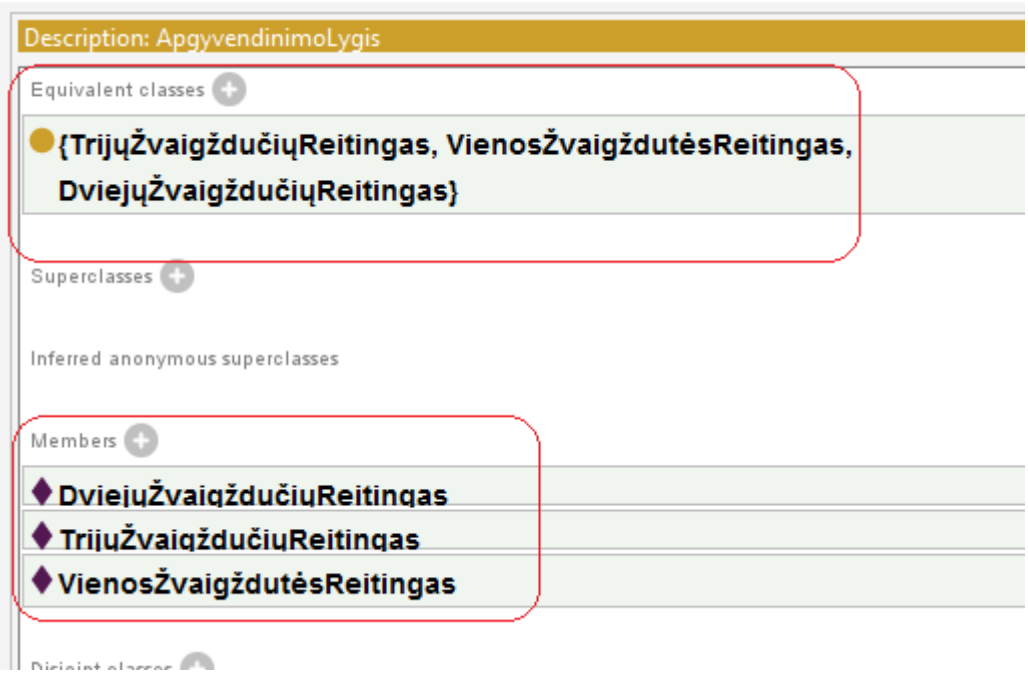
- Paskutinė klasės „KelionėsVieta“ subklasė yra „KelionėSuŠeima“. Ji yra ekvivalenti pastarajai klasei ir turi 2 atributus: „turiApgyvendinimą“, kurio reikšmė gali būti bet kuri klasės įeinančios į šio atributo savybės reikšmių diapazoną instancija ir turėti mažiausiai vieną reikšmę; „turiUžsiėmimą“, kurio reikšmė gali būti bet kuri klasės įeinančios į minėtojo atributo savybės reikšmių spektrą instancija ir turėti mažiausiai 2 reikšmes:



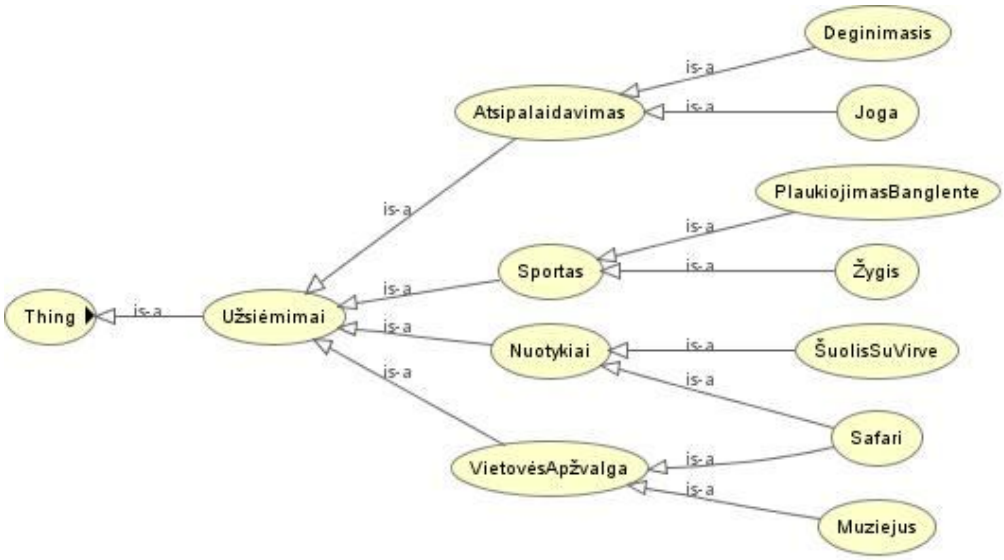
Trečioji pagrindinės klasės „Thing“ subklasė yra „Kontaktai“. Ši klasė neturi nei subklasių, nei atributų, nei individų:

Ketvirtoji ontologijos „Thing“ klsės subklasė yra „ApgyvendinimoLygis“. Ši klasė turi ekvivalenčius ryšius su trimis individais: „TrijųŽvaigždučiųReitingas“, „VienosŽvaigždutėsReitingas“ ir „DviejųŽvaigždučiųReitingas“. Ši klasė taip pat juos turi kaip savo individus:

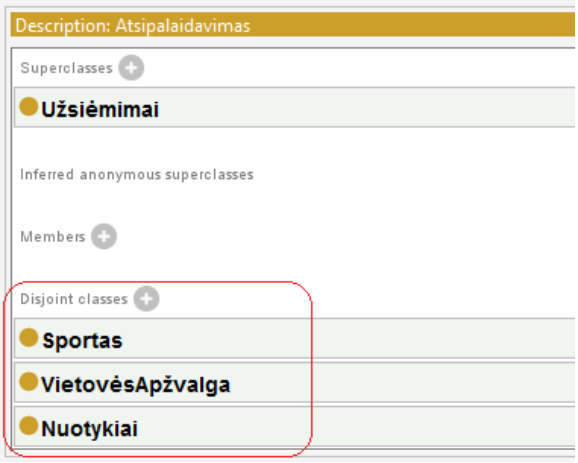




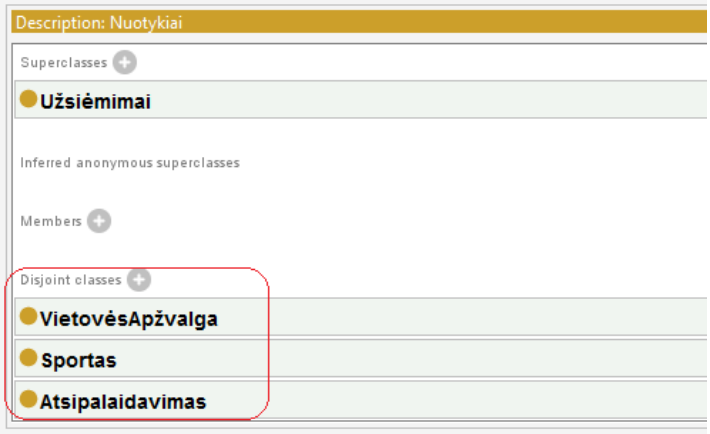
Paskutinė penktoji „Thing“ subklasė yra „Užsiėmimai“:



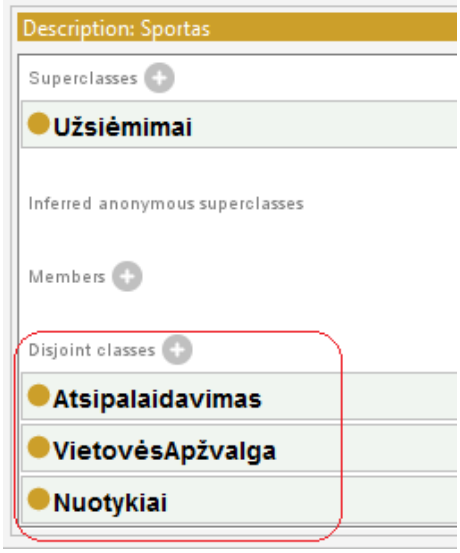
„Atsipalaidavimas“ yra pirmoji jos subklasė. Kuri pati turi 2 subklases: „Deginimasis“ ir „Joga“. Taip pat ši klasė negali būti jungiama su klasėmis „Sportas“, „VietovėsApžvalga“ ar „Nuotykiai“ klase:



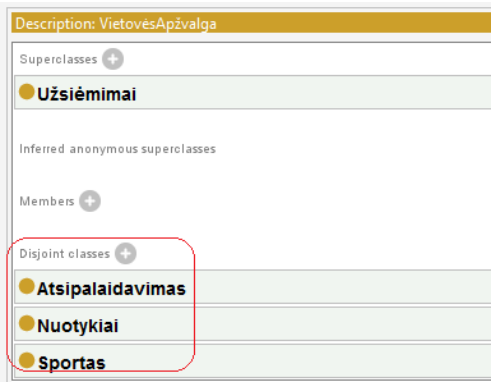
Antroji „Užsiėmimai“ subklasė yra „Nuotykiai“. Joje randame 2 subklases „Safari“ ir „ŠuolisSuVirve“. Taip pat klasės „Nuotykiai“ negalima jungti su klasėmis „VietovėsApžvalga“, „Sportas“, „Atsipalaidavimas“:



Subklasė „Sportas“ negali būti jungiama su klasėmis „Atsipalaidavimas“, „VietovėsApžvalga“, „Nuotykiai“. Taip pat ji turi 2 subklases: „PlaukiojimasBanglente“ ir „Žygis“:

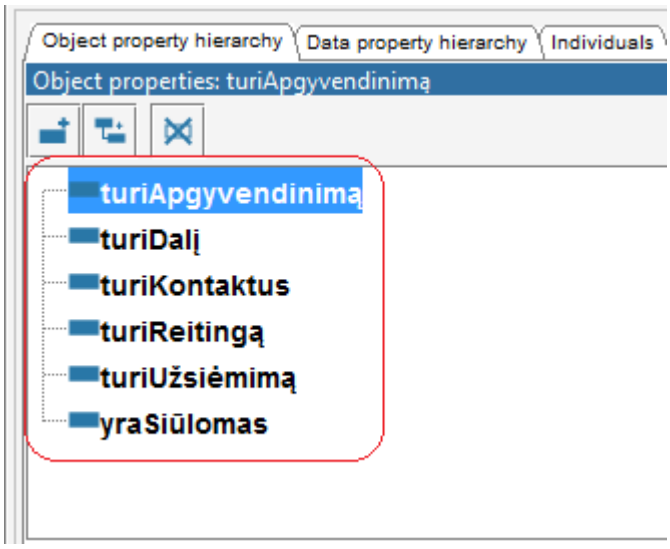


Paskutinė Mūsų apžvelgiama klasė „VietovėsApžvalga“ yra ir paskutinė „Užsiėmimai“ klasės subklasė. Ji nesaveikauja su klasėmis „Atsipalaidavimas“, „Nuotykiai“, „Sportas“. Taip pat turi 2 subklases: „Muziejus“ ir „Safari“:



6.2.2. Objektų atributų anazilė

Kiekvienas objekto atributas turi klasę ar klasių grupę, kuriose jis gali būti vartojamas, savo reikšmių diapazoną bei charakteristikas.



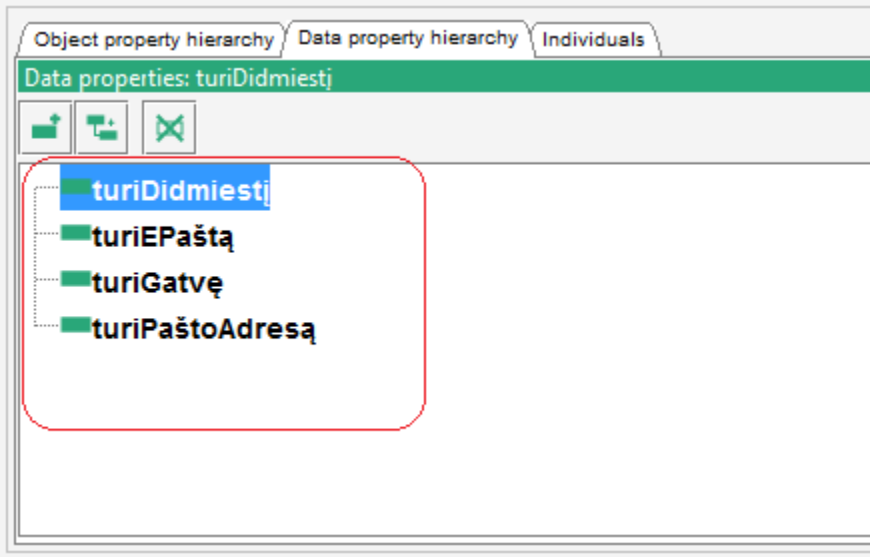
Ši ontologija turi 6 atributus:

- „turiApgyvendinimą“
    - Gali būti vartojamas „KelionėsVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Apgyvendinimas“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
- „turiDalį“
    - Gali būti vartojamas „KelionėsVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „KelionėsVieta“ klasė ir jos subklasės.
    - Turi pereinamumo charakteristiką.
- „turiKontaktus”
    - Gali būti vartojamas „Užsiėmimai“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Kontaktai“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.

- “turiReitingą“
    - Gali būti vartojamas „Apgyvendinimas“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „ApgyvendinimoLygis“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
- „turiUžsiėmimą“
    - Gali būti vartojamas „KelionėsVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Užsiėmimai“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
    - Priešingas atributas „yraSiūlomas“ atributui.
- „yraSiūlomas“
    - Gali būti vartojamas „KelionėsVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Užsiėmimai“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
    - Priešingas atributas „turiUžsiėmimą“ atributui.

6.2.3. Duomenų atributų analizė

Duomenų atributas turi klasę ar klasių grupę, kuriose jis gali būti naudojamas, savo tipą bei funkcinę charakteristiką.



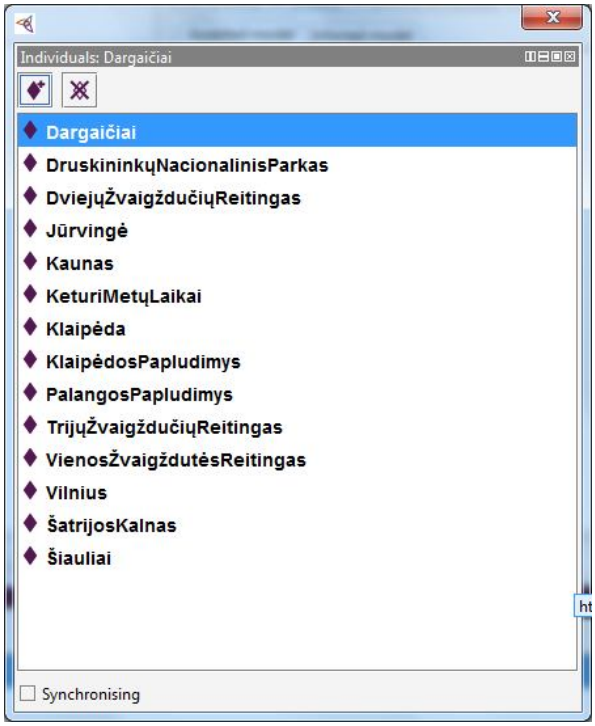
Ši ontologija turi 4 duomenų atributus:

- „turiDidmiestį“
    - Šis atributas gali būti naudojamas „Kontaktai“ klasėje, bei jos subklasėse.
    - Atributo tipas yra „string“ – tekstinė eilutė.
    - Turi funkcinę charakteristiką.
- „turiEPaštą“
    - Šis atributas gali būti naudojamas „Kontaktai“ klasėje, bei jos subklasėse.
    - Atributo tipas yra „string“ – tekstinė eilutė.
    - Turi funkcinę charakteristiką.
- „turiGatvę“
    - Šis atributas gali būti naudojamas „Kontaktai“ klasėje, bei jos subklasėse.
    - Atributo tipas yra „string“ – tekstinė eilutė.
    - Turi funkcinę charakteristiką.
- „turiPaštoAdresą“
    - Šis atributas gali būti naudojamas „Kontaktai“ klasėje, bei jos subklasėse.

- Atributo tipas yra „int“ – sveikasis skaičius.
- Turi funkcinę charakteristiką.

6.2.4. Individų analizė

Šioje ontologijoje yra 14 individų:



- „Dargaičiai“
  - „KaimoTurizmas“ klasei priklausanti individualybė.
- „DruskininkųNacionalinisParkas“
  - „NacionalinisParkas“ klasei priklausantis individas.
- „DviejųŽvaigždučiųReitingas“
  - „ApgyvendinimoLygis“ klasei priklausantis individas.
  - Yra klasių „VienosŽvaigždutėsReitingas“ ir „TrijųŽvaigždučiųReitingas“ priešingybė.
- „Jūrvingė“
  - „KaimoTurizmas“ klasei priklausintis individas.

- „Kaunas“
    - „Didmiestis“ klasei priklausantis individas.
- „KeturiMetųLaikai“
    - „PrabangusViešbutis“ klasei priklausantis individas.
- „Klaipėda“
    - „Didmiestis“ klasei priklausantis individas.
    - Turi 3 duomenų atributus:
        - „turiApgyvendinimą“ su reikšme „KeturiMetųLaikai“
        - „turiDalį“ su reikšme „PalangosPapludimys“
        - „turiDalį“ su reikšme „KlaipėdosPapludimys“
- „KlaipėdosPapludimys“
    - „Papludimys“ klasei priklausantis individas.
- „PalangosPapludimys“
    - „Papludimys“ klasei priklausantis individas.
- „TrijųŽvaigždučiųReitingas“
    - „ApgyvendinimoLygis“ klasei priklausantis individas.
    - Yra klasių „VienosŽvaigždutėsReitingas“ ir „DviejųŽvaigždučiųReitingas“ priešingybė.
- „VienosŽvaigždutėsReitingas“
    - „ApgyvendinimoLygis“ klasei priklausantis individas.
    - Yra klasių „TrijųŽvaigždučiųReitingas“ ir „DviejųŽvaigždučiųReitingas“ priešingybė.
- „Vilnius“

- „Sostinė“ klasei priklausantis individas.

- „ŠatrijosKalnas“
  - „NacionalinisParkas“ klasei priklausantis individas.

- „Šiauliai“
  - „Miestelis“ klasei priklausantis individas.

6.3. Atostogų ontologijos analizė

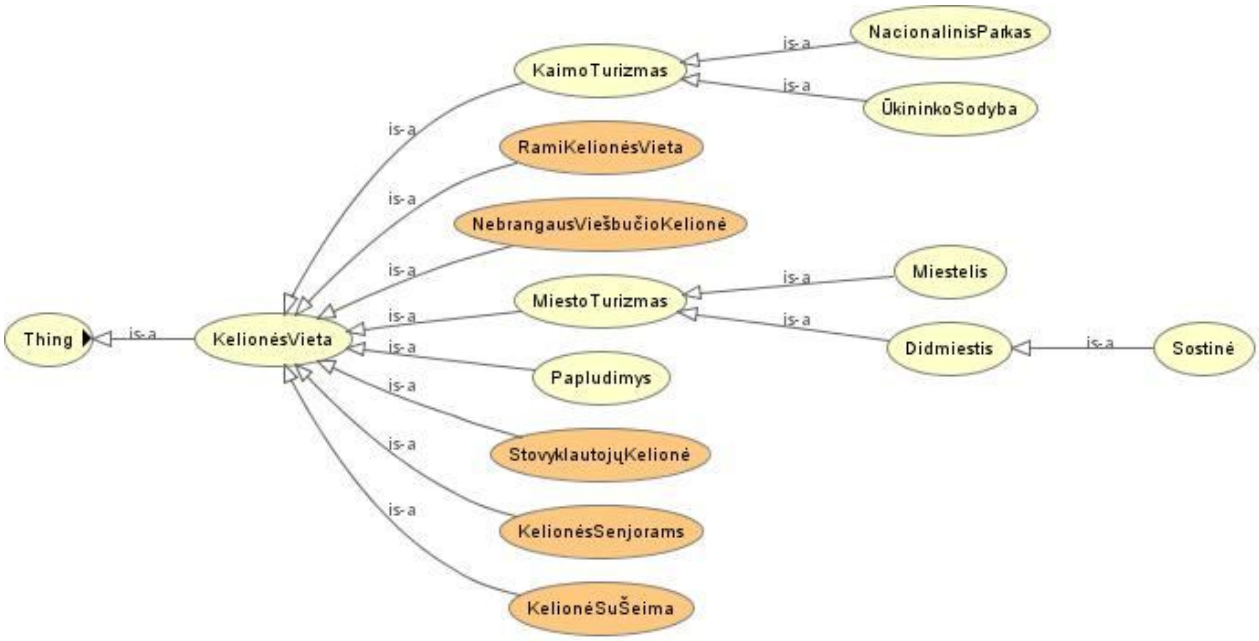
Atostogų ontologiją sudaro 34 klasės, 6 objekto atributai ir 14 individualybių.

Atostogų ontologijos analizė (žr. Priedus „Atostogų ontologijos analizė“) yra analogiška Kelionių ontologijos analizei.

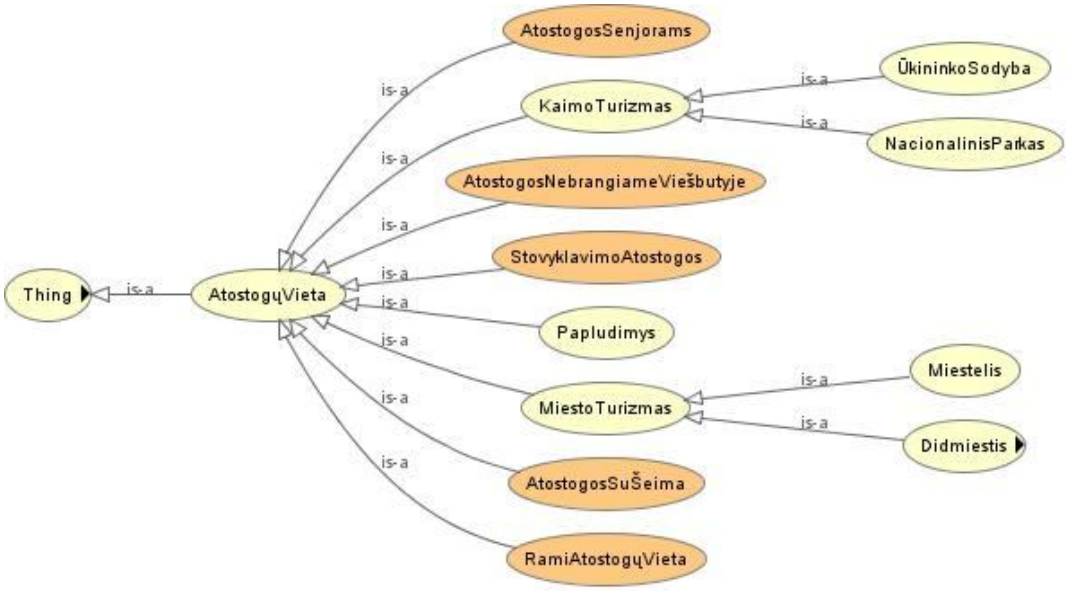
6.4. Ontologijų apjungimas

Ontologijų apjungimui pasirinkau 2 panašių dalykinių sričių ontologijas. Tai „Kelionių“ ontologija ir „Atostogų“ ontologija. Jos turi 3 sutapatinamas sritis:

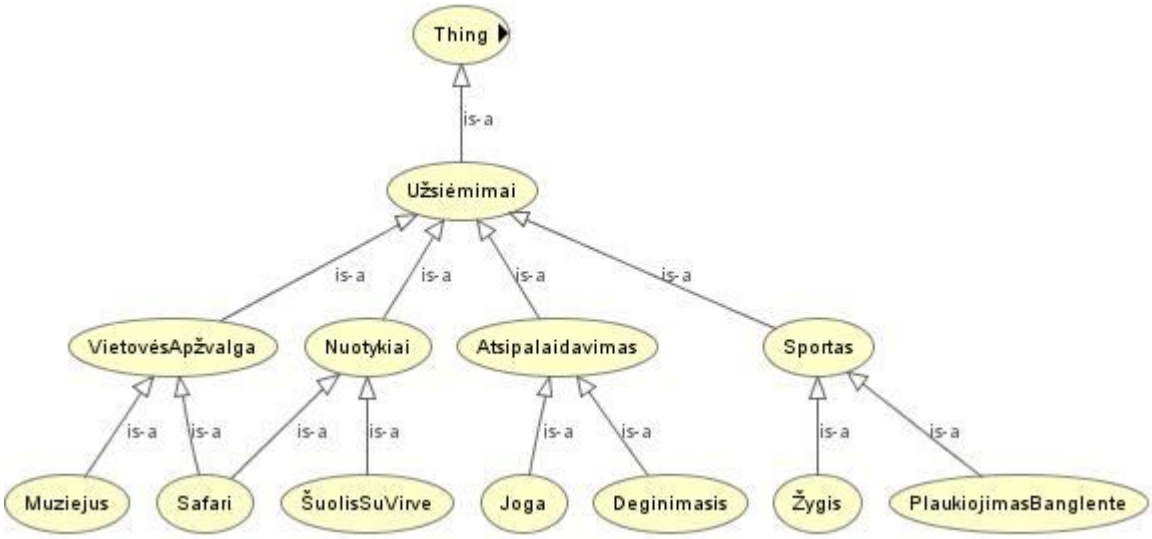
„Kelionės“ -> „KelionėsVieta“:



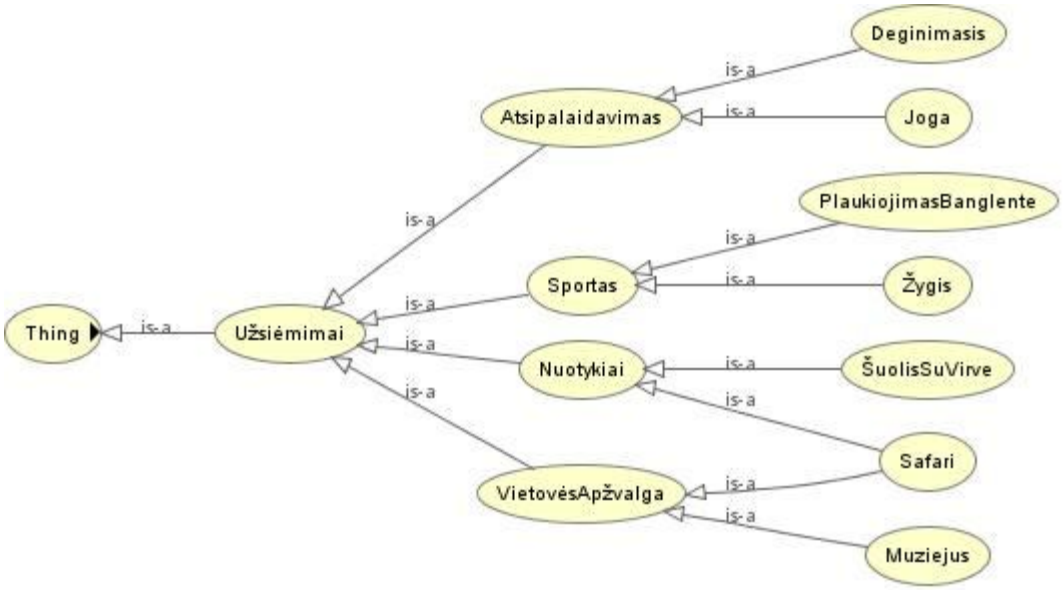
Ir „Atostogos“ -> „AtostogųVieta“:



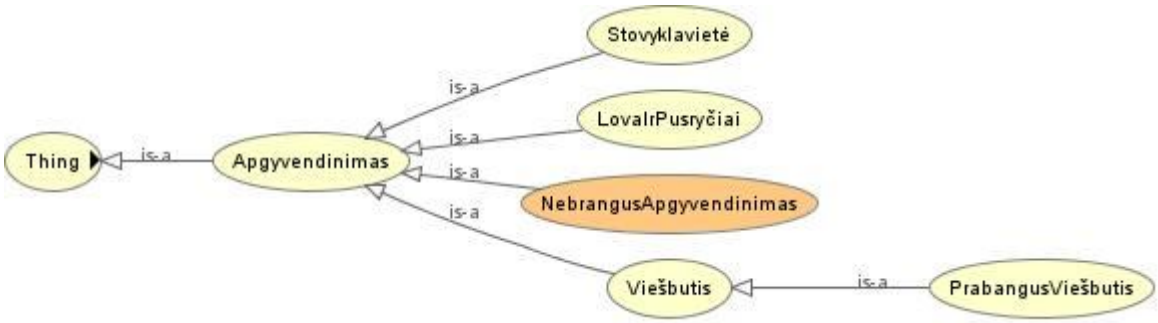
„Kelionės“ -> „Užsiėmimai“:



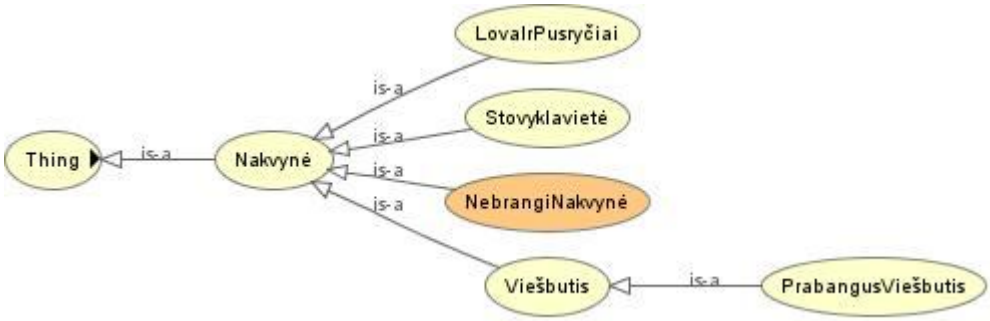
Ir „Atostogos“ -> „Užsiėmimai“:



„Kelionės“ -> „Apgyvendinimas“:



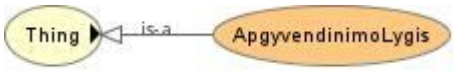
Ir „Atostogos“ -> „Nakvynė“:



Be šių trijų klasių „Kelionės“ turi dar 2 klases, kurios negalima sulieti su kitomis „Atostogos“ ontologijoje esančiomis klasėmis:

Tai klasė „Kontaktai“:

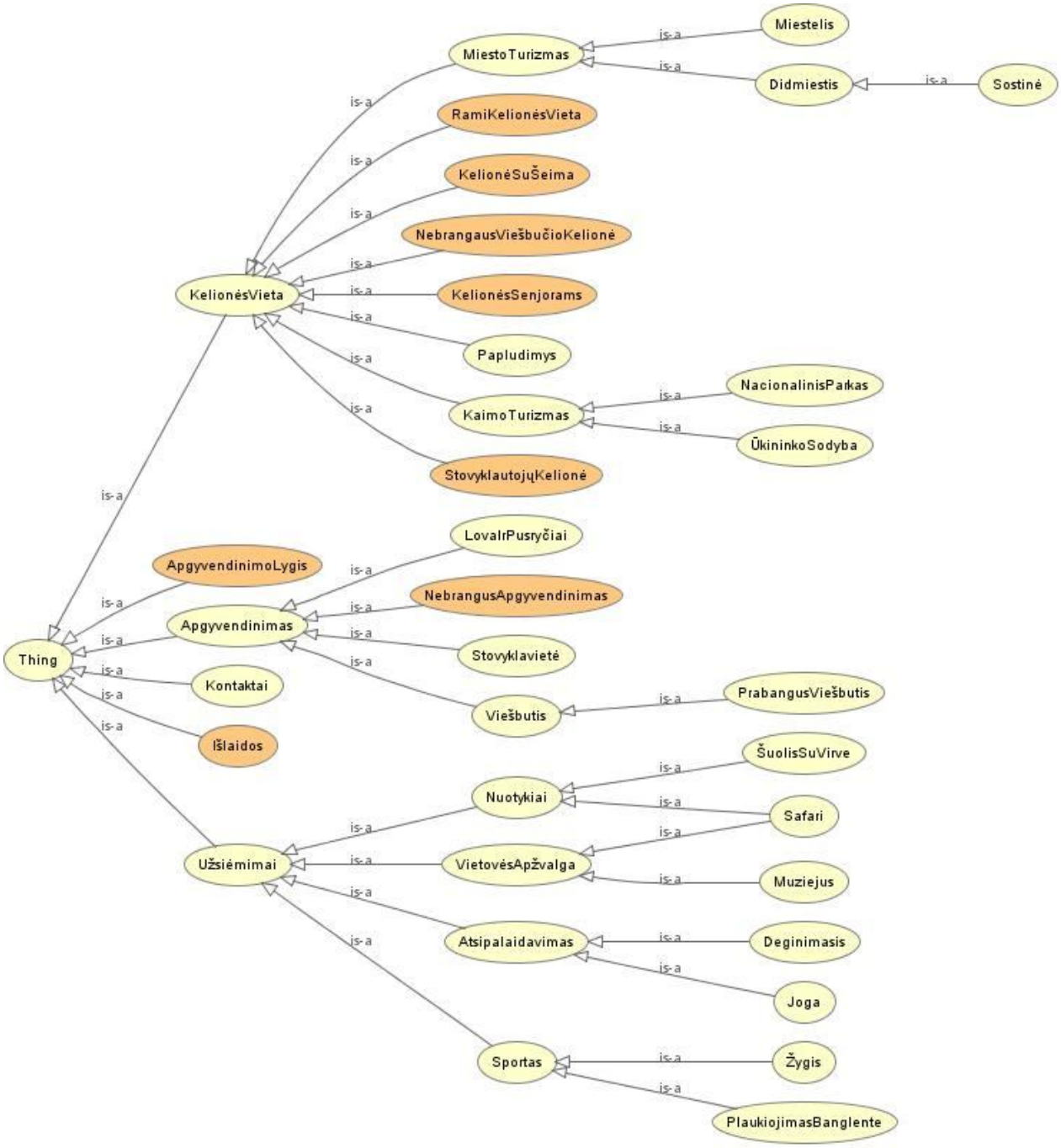
Ir klasė „ApgyvendinimoLygis“:



O ontologija „Atostogos“ turi vieną klasę „Išlaidos“, neturinčią atitikmens „Kelionės“ ontologijoje:

Žinant šiuos ryšius galime abi ontologijas sujungti į vieną bendrą. Ji turės 3 klases, kurios buvo sugretintos iš atskirų ontologijų ir 3 klases kurios susideda iš dviejų „Kelionės“

ontologijoje nejungiamų klasių ir vienos „Atostogos“ ontologijos klasės. Taip sukuriama ontologija yra bendra savo paskirtimi ir turi visas objekto būdo savybes, duomenų būdo savybes bei individus, priklausančius abiems ontologijoms esant atskirai. Apjungus ontologijas gavome vieną naują ir pavadinome ją „KelioniųAgentūra ontologija, kurią sudaro: 36 klasės 7 objekto būdo savybės, 4 duomenų būdo savybės ir 17 individualių objektų.:



7. Išvados

Dėl intensyvaus intormacijos augimo Internete vis aktualiau tampa kuo efektyviau panaudoti didelės apimties informacinius išteklius. Akivaizdu, kad informacijos struktūrizacija, taikoma ontologijose, teikia papildomas informacinės paieškos galimybes. Tačiau neretai tarp įvairių ontologijų rūšių yra tiesiog susipainiojama. Atskiriamos šios pagrindinės ontologijų rūšys:

- aukščiausio lygio ontologijos, apibrėžiančios pagrindines organizacijos koncepcijas;
- vidutinio lygio ontologijos, apibrėžiančios specifines, tik tos organizacijos koncepcijas;
- dalykinės ontologijos, kuriose vykdomas konkrečios organizacijos darbas.

Ontologijų rūšių žinojimas labai padeda dirbant su ontologijomis, kadangi kiekviena iš jų turi savo specifiką, aspektus, taikymo būdus, todėl ir apjunginėjant dvi ontologijas būtina atsižvelgti į jų konkrečias savybes ir taikymo sritis.

Skyriuje „Ontologijų apjungimas“ nagrinėjamas dviejų ontologijų apjungimo metodas, kuriame bendra ontologija kuriama papildant ją kitomis, pradinėmis ontologijomis kartu su koncepcijomis ir santykiais tarp jų. Kiekviename etape yra patikrinamos apjungtos ontologijos teisingumas deskriptyvios logikos pagalba. Tai leidžia įgyti teorinių žinių norint savarankiškai apjungti dvi ontologijas į vieną. Praktinėje dalyje pasinaudojus įgautomis žiniomi buvo apjungtos 2 panašaus pobūdžio ontologijos. Jos buvo sugretintos ieškant vietų, kuriose ontologijos dalykinės sritys sutampa. Sekantis žingsnis buvo gretinti tas vietas ieškant atributų panašumų, bei rankiniu būdų sprendžiant iškilusius konfliktus. Portege įrankis sudaro vartotojui prieinamą ir suprantamą aplinką, kurios dėka galima patogiai keisti ontologijas norima kryptimi. Kaip ir dauguma naujųjų technologijų pradžioje iškyla tam tikrų sunkumų naudojant lietuviškas raides. Tačiau šios problemos būna išsprendžiamos.

Ontologijos yra daugiau nei sudėtingas priėjimas prie informacijos aprašymo ir klasifikacijos. Jos gali būti naudojamos, pavyzdžiui, elektroninių bibliotekų, realizuotų kaip atskiros intelektualinės sistemos, funkcionavimui ir tobulėjimui.

Intensyviai užsiimant ontologijų integravimo algoritmais, tyrėjai vis dėlto vienareikšmiškai teigia, kad absoliučiai automatizuoti ontologijų kūrimo bei apjungimo procesų negalima ir artimoje ateityje nenumatomos tokios galimybės, tačiau pasistengti kiek tik įmanoma maksimaliai juos automatizuoti yra būtina tam, kad sutaupyti jėgų ir laiko.

Literatūros sąrašas

1) „CREATE PRODUCTION SYSTEM FOR ANALYSES SCIENTIFIC TEXTS“ L.V.Najhanova, N.B.Haptahaeva

2) T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 1993.

3) T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Presented at the Padua workshop on Formal Ontology, March 1993, later published in International Journal of Human-Computer Studies, Vol. 43, Issues 4-5, November 1995, psl. 907-927.

4) Deborah L. McGuinness. “Ontologies Come of Age”. In: D. Fensel, J. Hendler, H. Lieberman, W. Wahlster (eds.) The Semantic Web: Why, What, and How, MIT Press, 2001.

5) N. Guarino. Formal Ontology and Information Systems. In: N. Guarino (ed.), Formal Ontology in Information Systems. Proceedings of FOIS’98, Trento, Italy, June 1998. IOS Press, pp. 3-15.

6) formalių konceptų analizė,kuriant taikomosios srities ontologijas Darius jurkevičius Olegas vasilecas algirdas laukaitis

7) „Онтологии и тезаурусы: модели, инструменты, приложения“ Б.В. Добров, В.В. Иванов, Н.В. Лукашевич, В.Д. Соловьев

8) „The role of common ontology in achieving sharable, reusable knowledge bases“ Gruber T.R., 1991, psl. 601-602.

9) Guriano N. Understanding, Building, and Using Ontologies / A Commentary to “Using Explicit Ontologies in KBS Development” // International Journal of Human and Computer Studies, 1997

10) „Основные аспекты построения онтологий верхнего уровня и предметной области“ - Л. В. Найханова

11) Guriano N., Gangemi A., Pisanelli D. M., Steve G. An Overview of the ONIONS Project: Applying Ontologies to the Integration of Medical Terminologies // Data & Knowledge Engineering, 1999.

12) „Ontology Evolution“ by Raul Palma, Peter Haase

13) „Ontologijų naudojimo ypatumai kuriant moderniąsias informacines sistemas.“ Donatas Čiukšys, Albertas Čaplinskas; 2003. Vilnius

14) „Ontology Evolution and Versioning The state of the art“ by Burcu Yildiz

15) Maikevich N. V., Khoroshevsky V. F., „Intelligent Processing of Web Resources: Ontology-Based Approach and Multiagent Suppor, In: Proceedings of 1st International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'99)“, 1999, St.-Petersburg, Russia.

16) „Обзор баз знаний онтологического типа“ - А.А. Никоненко

17) „ONTOLOGIJŲ NAUDOJIMAS UNIVERSITETO E.PUBLIKACIJŲ ŽINIATINKLIO PASLAUGŲ TAISYKLIŲ SISTEMAI KURTI“ Vilniaus Gedimino technikos universitetas, Rūta Dubauskait , Olegas Vasilecas

18) Тузовский А.Ф. Онтолого-семантический подход к созданию систем управления знаниями организаций // Интеллектуальные системы (INTELS’2006, Краснодар, 2006. Psl. 286—290.

19) McGrath R.E. Semantic infrastructure for a ubiqutious computingenvironment – 2005.

20) Тузовский А.Ф. Работа с онтологической моделью организации на основе дескриптивной логики // Известия Томского политехнического университета. – 2006. psl. 134–137.

21) The Description Logic handbook: theory , implementation, applications // Ed. F. Baader. – Cambridge: Cambridge University Press, 2003. – 564 psl.

22) Anand R., McGrath R., Campbell R., Mickunas M. Use of Ontologies in a Pervasive Computing Environment // Knowledge Engineering Review. – 2004. – psl. 209–220.

23) МЕТОД ОБЪЕДИНЕНИЯ ОНТОЛОГИЙ ПРЕДМЕТНЫХ ОБЛАСТЕЙ ЗНАНИЙ - А.Ф. Тузовский, 2006, psl. 138-141

24) Großer Beleg „Ontology Composition using a Role Modeling Approach“, 2007, psl. 36-40

25) E. Mena, A. Illarramendi, V. Kashyap, A. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across preexisting ontologies. Distributed and Parallel Databases, An International Journal, Vol.8, No.2, 2000

26) „Вопросы согласования неоднородных онтологических моделей и онтологических контекстов“, Скворцов Н. А.

27) OWL Web Ontology Language Guide, W3C Recommendation 10 February 2004

28) „Handbook on ontologies“, S. Staab, R.Studer, 2004, psl. 93-117

29) „RELATION DATA BASE ONTOLOGY. LINGUISTIC ASPECT”, Биряльцев Е.В. , Гусенков А.М.

30) „ОНТОЛОГИЯ КАК СИСТЕМАТИЗАЦИЯ НАУЧНЫХ ЗНАНИЙ:СТРУКТУРА, СЕМАНТИКА, ЗАДАЧИ“, Кузнецов О.П. Суховеров В.С. Шипилина Л.Б.

Anotacija

Paskutiniais metais ontologijos tapo labai populiariomis Žiniatinklyje. Ontologijos vis plačiau naudojamos įvairiose srityse tokiose kaip verslo procesų intergavimas, paieška, biomedicininė informatika, dirbtinis intelektas, semantinis žiniatinklis, programinės įrangos kūrimas, žinių apsikeitimas, navigacija ir informacijos architektūra tam, kad pavaizuoti žinias apie pasaulį arba atskiras jo dalis. Ontologijas naudoja žmonės, duomenų bazės, programos kurioms reikalingos konkrečios žinios apie vieną ar kitą sritį (tai gali būti medicina, nekilnojamasis turtas, finansų valdymas ir t.t.). Jos aprašo:

- *Egzempliorius* (taip pat žinomi kaip *individai*): pagrindiniai žemo lygio ontologijų komponentai (objektai);
- *Klases* (taip pat žinomi kaip *konceptai*): abstrakčios grupės, kolekcijos arba objektų rinkiniai;
- *Atributus*: savybės, charakteristikos, parametrai kuriuos gali turėti objektai ir jais keistis;
- *Ryšius*: būdai kaip objektai gali būti susiję tarpusavyje.

Pagrindinės priežastys dėl kurių yra naudojomos ontologijos yra tai, kad žmonės nori pasidalinti bendromis informacijos struktūros koncepcijomis tarp jų pačių arba tarp programų agentų, jie nori perpanaudoti konkrečių dalykinių sričių žinias, analizuoti dalykines sritis ir, aišku, askirti dalykines sritis nuo veiklos žinių. Ontologijos daro mūsų gyvenimą lengvsniu ir padeda suprasti specifinę dalykinę informaciją kuri mus supa.

Summary

In recent years ontologies have become common on the World Wide Web. They are increasingly used in many domains such as business process integration, artificial intelligence, the semantic web, software engineering, information integration, search, biomedical informatics, navigation and information architecture as a form of knowledge representation about the world or some part of it and are also used for knowledge sharing. Ontologies are used by people, databases, and applications that need to share specific domain

information (it can be medicine, real estate, automobile repair, financial management, tool manufacturing etc.). They describe:

- *Instances* (a.k.a. *individuals*): the basic or "ground level" components;
- *Classes* (a.k.a. *concepts*): abstracht groups, collections or types of objects;
- *Attributes*: properties, features, characteristics, or parameters that objects can have and share;
- *Relations*: ways that objects can be related to one another.

The main reasons why ontologies are developed are that people want o share common understanding of the structure of information among them or software agents, they want to enable reuse of domain knowledge, analyze domain knowledge, and, of course, to separate domain knowledge from the operational knowledge. Ontologies make our lifes easier and helps to understand specific domain information among us.

PRIEDAI

Apjungimo operacija nėra komutatyvi

Panagrinėkime dvi ontologijas, $O_1 = (C_1, R_1)$ ir $O_2 = (C_2, R_2)$ ir santykių rinkinį L. Apjungimo operacija compose (O_1, O_2, L) sukuria ontologiją O_{1+2}, kuri yra apibrėžta taip: $O_{1+2}=(C_1 \cup C_2, R_1 \cup R_2 \cup L)$. Analogiškai compose ($O_2$, O_1, L) sukuria ontologiją O_{2+1}, kuri yra apibrėžta taip:

$O_{1+2}=(C_2 \cup C_1, R_2 \cup R_1 \cup L)$. Šitie du grafai sukuria vieną ir tą patį teiginių deskriptyvioje logikoje rinkinį, kuriame gali skirtis tik elementų nuoseklumas. Dėl to, kad deskriptyvi logika yra monotoniška, išvestis iš žinių bazės $ŽB_{1+2}$ bus teisingas ir žinių bazei $ŽB_{2+1}$. Todėl satisfiable($ŽB_{1+2}$) $\Leftrightarrow$ satisfiable($ŽB_{2+1}$) ir valid(O_{1+2}) $\Leftrightarrow$ valid(O_{2+1}).

Apjungimo operacija yra asociatyvi

Trys ontologijos yra apjungiamos dvejais žingsniais:

1) Norint sukurti ontologiją $O_{(1+2)+3}$, pirmame žingsnyje apjungiama 1 ir 2 ontologijas: O_{1+2}=compose (O_1, O_2, L_{1+2})=($C_1 \cup C_2$, $R_1 \cup R_2 \cup L_1$+2). Tarkime, kad teiginys valid(O_{1+2}) yra teisingas, t.y. satisfiable($ŽB_{1+2}$) yra teisingas.

2) Antrame žingsnyje pridedame ontologiją O_3:

   $O_{(1+2)+3}$=compose (O_{1+2}, O_3, ($L_{1+3} \cup L_{2+3}$))=(($C_1 \cup C_2$) $\cup C_3$, ($R_1 \cup R_2 \cup L_{1+2}$) $\cup R_3 \cup L_{1+3} \cup L_{2+3}$)=(($C_1 \cup C_2 \cup C_3$), ($R_1 \cup R_2 \cup R_3 \cup L_{1+2} \cup L_{1+3} \cup L_{2+3}$).

Tarkime, kad teiginys valid($O_{(1+2)+3}$) yra teisingas. Tam, kad sukurti $O_{1+(2+3)}$ galima atlikti du analogiškus apjungimus:

1) O_{2+3}=compose (O_2, O_3, L_{2+3})=($C_2 \cup C_3$, $R_2 \cup R_3 \cup L_{2+3}$)

   Ir

2) $O_{1+(2+3)}$=compose (O_1, O_{2+3},($L_{1+3} \cup L_{2+3}$))=(($C_2 \cup C_3$) $\cup C_1$, ($R_2 \cup R_3 \cup L_{2+3}$) $\cup R_1 \cup L_{1+3} \cup L_{1+2}$)=(($C_1 \cup C_2 \cup C_3$), ($R_1 \cup R_2 \cup R_3 \cup L_{1+2} \cup L_{1+3} \cup L_{2+3}$).

Tai toks pats rinkinys kaip ir $O_{(1+2)+3}$, skiriasi tik elementų eiliškumas. Todėl teiginys ($O_{1+(2+3)}$) yra teisingas. Iš monotoniškumo ir teisingumo deskriptyvioje logikoje apibrėžimų seka, kad satisfiable($ŽB_{(1+2)+3}$) ⇔ satisfiable($ŽB_{1+(2+3)}$), todėl valid($O_{(1+2)+3}$) ⇔ valid($O_{1+(2+3)}$). Ši savybė rodo, kad kiekvienai O_{1+2+3}, gautai tokiu būdu kaip aprašyta aukščiau, yra teisinga: valid(O_{1+2+3}) ⇒ valid(O_{1+2}) ∧ valid(O_{1+3}) ∧[33] valid(O_{2+3}) ∧ valid(O_1) ∧ valid(O_2) ∧ valid(O_3).

Ontologijų kūrimo priemonių sąrašas

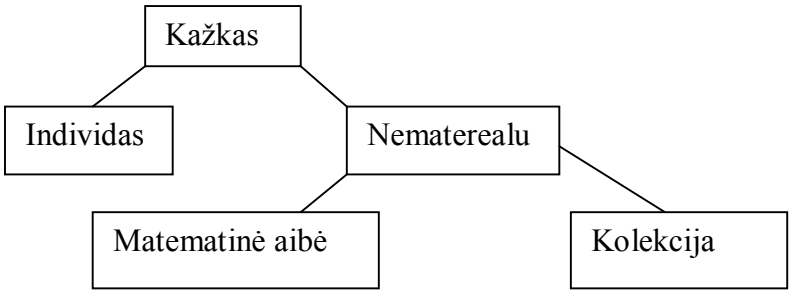
Šiuo metu populiariausios ontologijų kūrimo priemonės yra:

1) **Ontoligua** – šita aplinka yra skirta kolektyviniam bazinių žinių sistemos panaudojimui, kūriant savas ontologijas. Ontologijų kūrėjui yra pateikiama modulių biblioteka, kurios pagalba realizuojamas ontologijų plėtra.
2) **OntoEdit** – skirta proektuoti, pritaikyti ir importuoti/eksportuoti žinių modelius formatuose RDF, DAML+OIL, Flogic taikomojom sistemom.
3) **OilEd** – ontologijų redaktorius, skirtas sukurtų ontologijų patikrinimui.
4) **Protege** – tai biblioteka, teikianti prieinamumą kitoms programoms peržiūrėti žinių bazes, redaguoti ir papildyti jas. „Protege 4.0.2“ (buvo išleista 2009-12-03) yra laisvai platinama, plačiai naudojama, lengvai išplečiama ir naujausia iš Stanfordo universitete sukurtų stabilių versijų, skirta žinių įgijimui ir suderinama su OKBC, HTML, XML, RDF Schemų, JDBC ir kt. formatais.
5) Yra ir **kitos** ontologijų kūrimo aplinkos: Apollo, LinkFactory, OntoSaurus, OpenKnoME, SymOntoX, WebODE, WebOnto.

[33] Konjukcija, Apjungimas, operacija „IR“. Pvz.: A∧B yra tiesa tik tada, kai ir A ir B yra abudu tiesa.

Žymiausių aukščiausio lygių onfologijų aprašymas

1) **Cyc**[34] – didelės ontologijnės bazės sukūrimo projektas, leidžiantis programoms spęsti dirbtinio intelekto srities uždavinius, naudojantis loginėmis išvadomis. Projektas buvo pradėtas 1984m. Ir yra susijas su žinių inžinerija – faktų apie pasaulį aprašymu rankiniu būdu ir efektyvia loginių išvadų mechanizmo realizacija. Tačiau šiuo metu vyksta tobulinimas, padėsiantis Cyc sistemai savarankiškai bendrauti su vartotojais natūralia kalba ir paspartinti bazės papildymo procesą sistemos mokymosi[35] būdu.



1 pvksl. Kolekcijų Cyc hierachijos fragmentas

2) **DOLCE** (angl. Descriptive Ontology for Linguistic and Cognitive Engineering)– tai pirmas WonderWeb ontologijos bibliotekų pagrindų modulis. Jis apima naturalios kalbos bei sveiko žmogaus ontologijos kategorijas. DOLCE priskiriamas prie informacijos technologijų, kuris taikomas inteleketualių agentų[36], naudonačių skirtingas terminologijas, susitarimui. Šita ontologija nepretenduoja būti universalia, bendra ir standartine. Pagrindinis kūrėjo tikslas yra sukurti modelį, padedantį palyginti ir išaiškinti ryšius su kitomis bibliotekos WFOL (bazinės WonderWeb ontologijų bibliotekos) onotlogijomis.

---

[34] Cyc pavadinimas kylo nuo angliško žodžio *en**CYC**lopedia.*

[35] Sistemos mokymasis (angl. *machine learning)* - tai dirbtinio intelekto sritis, kuri apima metodų kūrimą, mokinančių kompiuterius „mąstyti“, t.y. programų kūrimo būdas, kai sukurta sistema prisitaiko prie duomenų („apsimoko“).

[36] Intelektualus agentas (angl. intelligent agent – IA) – tai savarankiškai vykdanti užduotį programa, kurią jai patikėjo kompiuterio vartotojas per tam tiktrą (dažniausiai ilgą) laiko tarpą.

DOLCE – tai individų ontologija, kurią gerai apibrėžia šis pavyzdys:

Jei koncepcija „šuo“ turi daug ekzempliorių, tai koncepcija „laikas“ – tai individas. Būtent tokių individų pagrindu ir remiasi DOLCE (žr. Priedus – paveiksliukas NR.7).

3) **SUMO** (angl. *Standard Upper Merged Ontology*) - IEEE SUO ir Teknowledge projektas, pretenduojantis į aukčiausio lygių ontologijų standarto laipsnį.

   SUMO apibrėžia tik pačius bendriaus ir abstrakčiausius konceptus, turi išsamią fundamentalių kocepcijų hierarchiją ir aksiomų, apibrėžiančių šias koncepcijas, rinkinį (žr. Priedus – paveiksliukas NR. 12).

4) **J.Sowa** – buvo pateikta autoriaus knygoje "Knowledge Representation : Logical, Philosophical, and Computational Foundations“ ir apibrėžia bazines ontologines kategorijas (žr. Priedus – lentelė NR. 1). Pagal J. Sowa tam, kad ontologija išliktų atvira, ji turi būti pagrįsta ne fiksuotos konceptų hierarchijos, o turėti „rėmą“, aprašantį skirtumus pagal kuriuos hierarchija yra automatiškai generuojama.

5) **Kitos** ontologijos.

Dalykinių sričių ontologijų alternatyvos plačiau

1) **Dublio branduolys** – metaduomenų elementų standartas (formatas), paprastas ir efektyvus rinkinys, skirtas aprašyti dalykinės srities ontologiją. Šitų elementų reikšmių terminuose galima aprašyti skirtingus tekstinius bei kitokio tipo dokumentus. Šios alternatyvos patrauklumas yra jos paprastume, bet būtent dėl šio paprastumo ji yra ribota.

Standartas suskirstytas į du lygius:

- Paprastas (nekvalifikuotas, angl. *simple*), susidedantis iš 15 elementų;
- Kompetetingas (kvalifikuotas, angl. *qualified*), susidedantis iš 18 lementų ir kvalifikatorių grupės, kurie patikslina elementų semantiką norint padidinti resursų paieškos naudingumą.

Pirmoji Dublino branduolio versija, turinti 13 elementų buvo pristatyta 1995 m. Simpoziume Dubline (JAV), kurį surengė Online Computer Library Center (OCLC) ir National Center for

Supercomputing Applications (NCSA) informacinių bibliotekos sistemų resursų aprašymui. Dublino branduolio plėtrai ir palaikimui yra net speciali organizacija pasivadinusi Dublin Core Metadata Initiative (DCMI).

Paprastas dublino elemento rinkinys susideda iš šių elementų (viso 15):

1. Title — pavadinimas;
2. Creator — kūrėjas;
3. Subject — tema;
4. Description — aprašymas;
5. Publisher — leidėjas;
6. Contributor — pagalbininkas;
7. Date — data;
8. Type — tipas;
9. Format — dokumento formatas;
10. Identifier — identifikatorius;
11. Source — išteklis;
12. Language — kalba;
13. Relation — santykiai;
14. Coverage — apimtis;
15. Rights — autorinės teisės

Kvalifikuotas rinkinys be aukščiau išvardintų komponentų turi dar papildomus tris:

16. Audience — auditorija (žiūrovai);
17. Provenance — kilmė;
18. RightsHolder — teisių turėtojas.

Kiekvienas elementas gali kartotis.

Plačiai naudojama OMF (angl. *Open Source Metadata Framework*) specifikacija yra sukurta Dublino branduolio pagrindu.

2) **Pirmo lygio logikos kalbos**

Ganėtinai populiaria tapo nurodytos kategorijos kalba KIF (angl. *Knowledge Interchange Format*) - žinių mainų formatas, sukurtas 90-ųjų m. Pradžioje ir turintis S-išraiškų sintaksę. Iš pradžių jis buvo kuriamas kaip formali kalba skirta teikti žinių tarp skirtingų sistemų mainus. Pati KIF kalba yra antros eilės predikatų skaičiavimas skirtas tarpmašininiams duomenų mainams.
Aplinka Ontolingua, skirta kolektyviniam bazinių žinių sistemos panaudojimui, kūriant savas ontologijas, buvo sukurta KIF pagrindu.
Ontologiją kanoniniu formatu galima lengvai transliuoti skirtingoms sistemoms, naudojančioms skirtingą sintaksę žinių vaizdavimui.

3) **Ontologijų aprašymo standarto kalba OWL**

Ši kalba yra kuriama nuo 2001 metų, pagrįsta aprašymo logikomis ir skirta intelektualių sistemų informacinių resursų paieškai aplinkoje WEB (antros kartos). Antros kartos WEB tikslas paversti WEB į semantinio lygio sistemą, teikti aumatizuotos semantinės interpretacijos ir informacinių resurso apdorojimo galimybes.

OWL kalba leidžia prašyti klases ir santykius tarp jų web-dokumentuose ir programose. OWL buvo sukurtas anklstesnių kalbų, tokių kaip OIL ir DAML+OIL pagrindu ir dabar yra rekomenduotas Pasaulinio tinklo konsorciumu.

Kalboje pateiktas modelis “objektas-savybė“, pati kalba skirta ne tik web-puslapių aprašymu, bet ir bet kuriam realiam objektui.

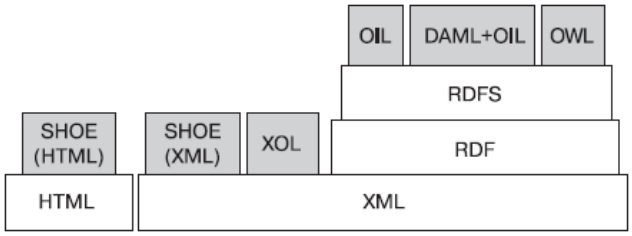
Yra 3 OWL versijos:

- **OWL Lite** skirtas vartotojams, kuriems reikia klasifikuotos hierarchijos ir nesudetingų apribojimų. Jis palaiko elementų kiekio apribojimus (kardinalumą), o kardinalumo reikšmės gali būti tik 0 arba 1. OWL Lite yra formaliai mažiau sudėtinga negu OWL DL.

- **OWL DL** skirta tiems vartotojams, kuriems reikalingas greitumas ir maksimalus išraiškingumas saugojant skaičiavimų pilnumą. OWL DL turi visas OWL kalbos konstrukcijas, bet jos naudojamos su tam tikrais apribojimais (pavyzdžiui klasė gali būti daugelių klasių poklase, bet pati negali būti kitos klasės atstove). OWL DL taip pavadinta dėl jos atitikimo deskriptyviai logikai – disciplinai, kurioje sukurtos logikos, sudarančios formalų OWL pagrindą.

- **OWL Full** skirta tiems vartotojams, kam reikalingas maksimalus išraiškingumas ir sintaksinė RDF laisvė be skaičiavimo garantijų. Pavyzdžiui, OWL Full gali būti laikomas vienu metu ir kaip individų rinkinys ir kaip vienas individas. OWL Full leidžia kurti ontologijas, išplėčiančias konkrečius žodynus, bet mažai tikėtina, kad koki nors programinė įranga galės pilnai palaikyti visas OWL Full savybes.

OWL yra 1.0, 1.1 ir 2.0 versijų.



9 pvksl. Pagrindinės WEB ontologijų aprašymo kalbų plėtros šakos

Kiekviena ontologija turi antraštę ir „kūną“. Antraštėje yra informacija apie pačią ontologiją (versija, pastabos) ir apie importuotas ontologijas. Po antraštės seka ontologijos kūnas, aprašantis klases, savybes ir egzempliorius.

Bazinės OWL savybės yra šios[37]:

- **Klasės** (owl:Class) yra naujas OWL įvestas terminas, kadangi ne visos dialektų klasės OWL DL ir OWL Lite yra RDFS-klasėmis (šiuo atveju owl:Class yra poklasė rdfs:Class ir yra viena kitos sinoniumas).

OWL klasės gali būti aprašytos 6 būdais:

1) Klasių identifikatoriumi (URI);

[37] „ Ontologijos ir tezaurai: modeliai, instrumentai, programos“ – B.Dobrov, V. Ivanov, N. Lukaševič, V. Solovjov

2) Visų klasių egzempliorių aprašymu;
3) Savybių reikšmių apribojimu;
4) 2 ir daugiau klasių apibrėžimų aprašymu;
5) 2 ir daugiau klasių apibrėžimų apjungimu;
6) Konkrečios klasės papildymu.

- **Savybės**

OWL turi dvi savybių kategorijas:

1) Savybės-objektai (arba objektinės savybės) – apjungia individus (klasių egzempliorius) tarpusavyje.

2) Savybės-reikšmės – apjungia individus su duomenų reikšmėmis.

- **Individai (klasių egzemplioriai)**

Individai apibrėžiami individų aksiomų pagalba (faktų):

1) Faktas apie individų narystę klasėse ir faktas apie individų savybių reikšmes;

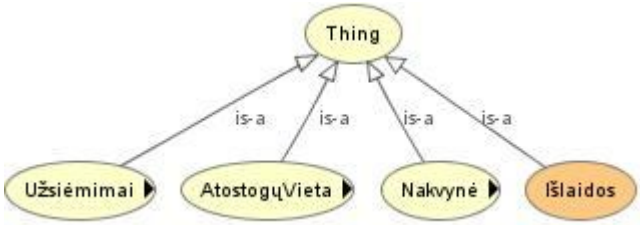
2) Identiškumo/skirtingumo faktas.

Atostogų ontologijos analizė

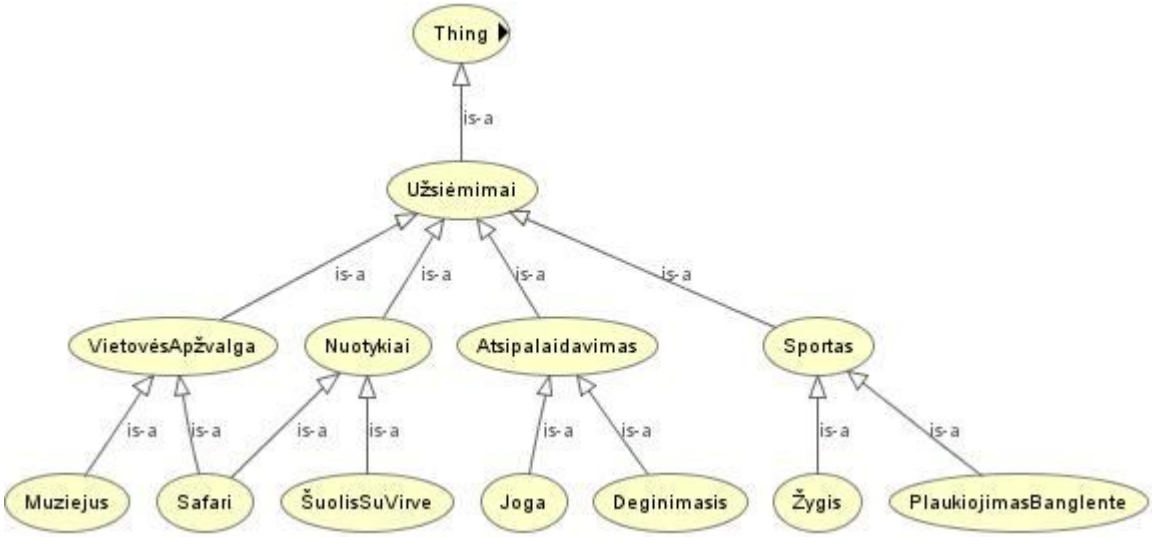
7.1.1. Klasių analizė

Pagrindinė „Atostogos“ ontologijos klasė „Thing“ yra sudaryta iš 4 subklasių:

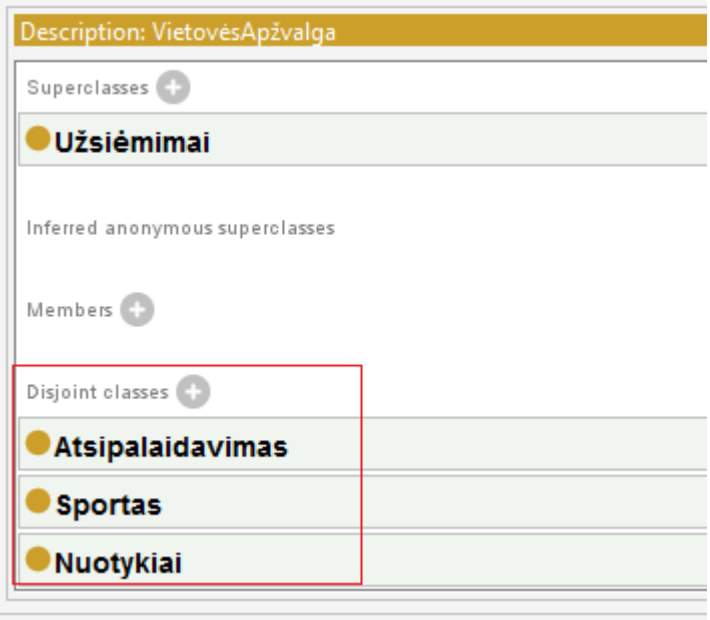
- Užsiėmimai
- Nakvynė
- AtostogųVieta
- Išlaidos



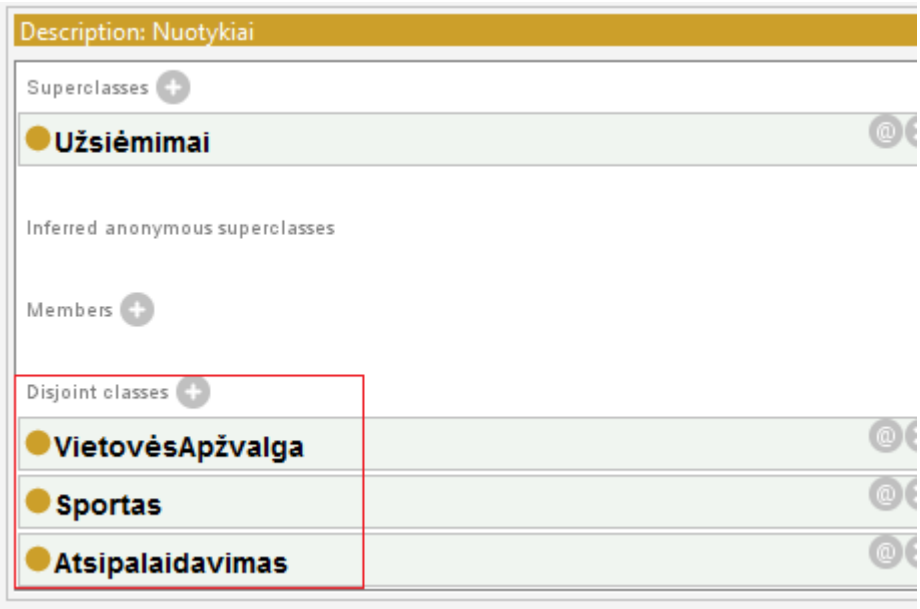
Pirmoji „Thing“ klasės subklasė yra „Užsiėmimai“. Ją sudaro 4 subklasės, iš kurių kiekviena turi po savo 2 subklases, o klasė „Safari“ yra iš karto dviejų klasių subklasė. Kitaip sakant klasė „Safari“ turi dvi superklases.



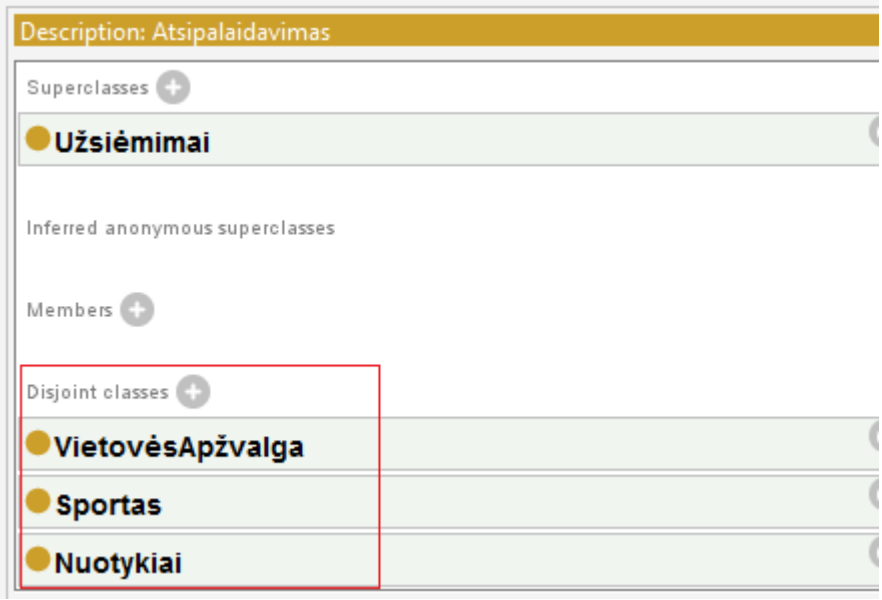
„Užsiėmimai“ klasės klasės pirmoji subklasė „VietovėsApžvalga“ turi 2 subklases „Muziejus“ ir „Safari“. Taip pat ji negali būti jungiama su klasėmis „Atsipalaidavimas“, „Sportas“ ir „Nuotykiai“.



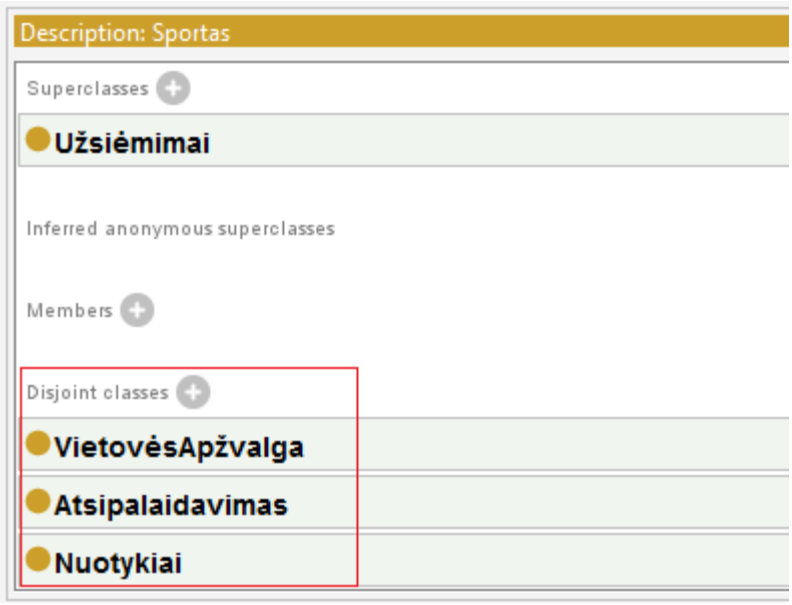
Antroji subklasė „Nuotykiai“ irgi turi 2 subklases „Safari“ ir „ŠuolisSuVirve“. Klasė negali būti jungiama su „VietovėsApžvalga“, „Sportas“ ir „Atsipalaidavimas“ klasėmis.



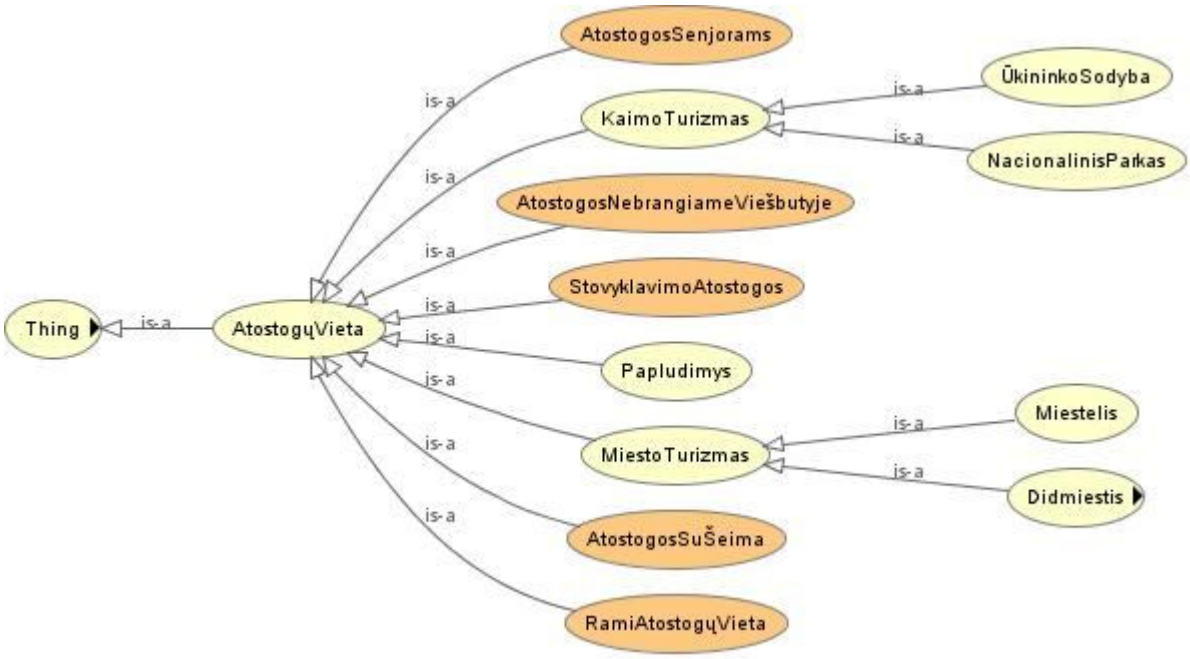
Trečioji klasė „Atsipalaidavimas“ kaip ir prieš tai buvusios jos lygio klasės turi 2 subklases „Deginimasis“ ir „Joga“. Ši klasė yra nejungtina su klasėmis „VietovėsApžvalga“, „Sportas“ ir „Nuotykiai“.



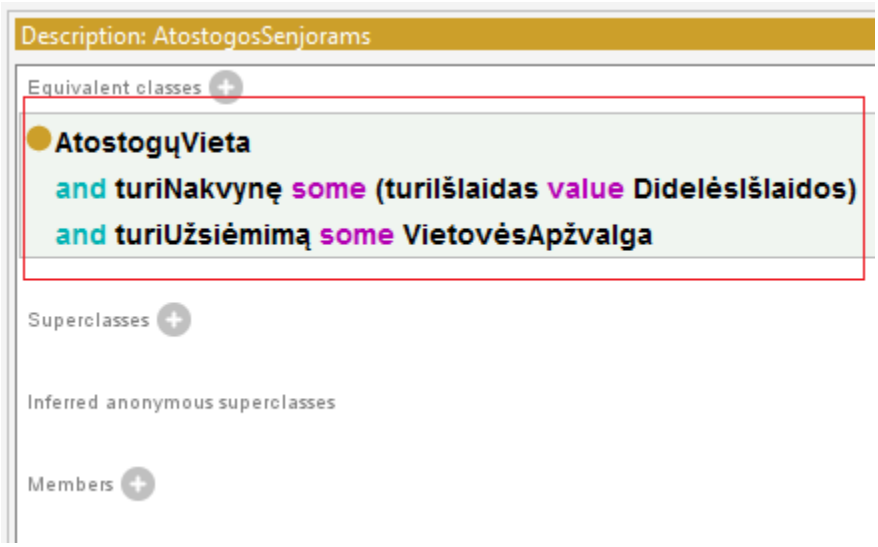
Paskutinė subklasė „Sportas“ turi 2 subklases „PlaukiojimasBanglente“ ir „Žygis“. Ji kaip ir prieš tai minėtos klasės, turi 3 negalimus klasių jungimo variantus: tai klasės „VietovėsApžvalga“, „Atsipalaidavimas“ ir „Nuotykiai“.



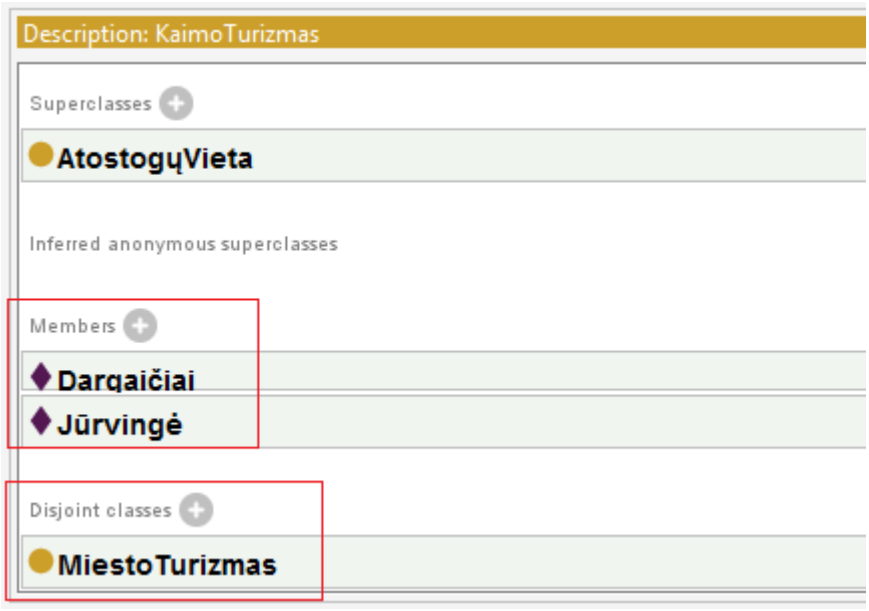
Antroji „Thing“ klasės subklasė yra „AtostogųVieta“, kurią sudaro 8 subklasės. Dvi iš jų turi po savo 2 subklases.



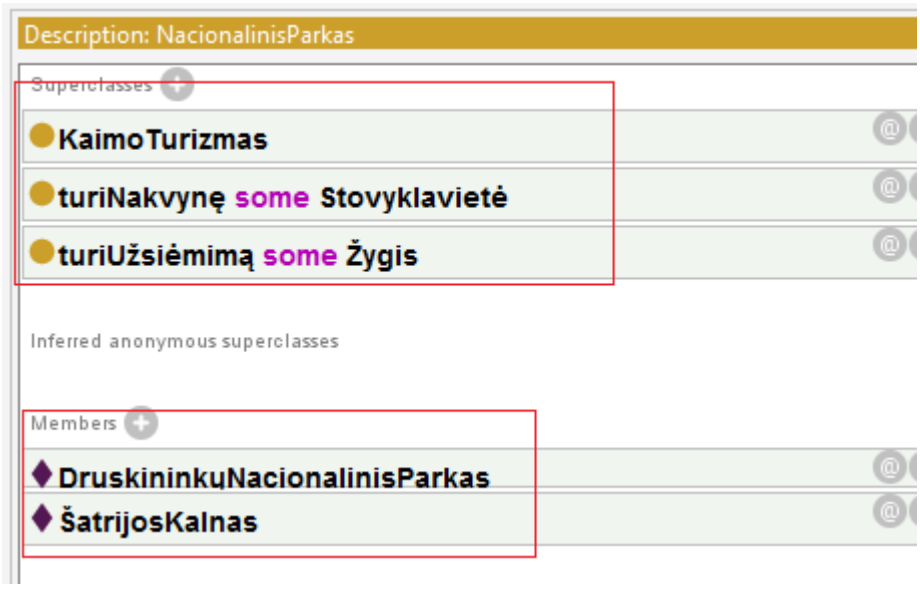
- „AtostogosSenjorams“ klasė yra ekvivalenti klasei „AtostogųVieta“ ir turi 2 atributus. Pirmasis atributas „turiNakvynę“ kurio reikšmė gali būti bet kokia klasė ekvivalenti „Nakvynė“ klasei ir kuri turi atributą „turiIšlaidas“ su reikšme „DidelėsIšlaidos“. Antrasis atributas „turiUžsiėmimą“ gali turėti reikšmę „VietovėsApžvalga“.



- Subklasė „KaimoTurizmas“ turi 2 subklases „NacionalinisParkas“ ir „ŪkininkoSodyba“. Taip pat „KaimoTurizmas“ klasė turi 2 individus „Dargaičiai“ bei „Jūrvingė“ ir yra nejungiama su klase „MiestoTurizmas“.

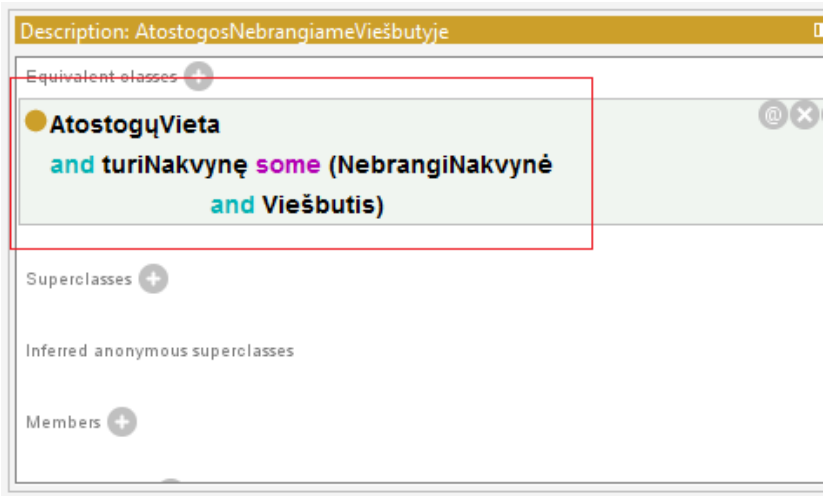


- „NacionalinisParkas“ klasė turi 2 atributis „turiNakvynę“ ir „turiUžsiėmimą“, kurie atitinkamai gali būti lygūs „Stovyklavietė“ ir „Žygis“ klasėms. Individai „DruskininkųNacionalinisParkas“ bei „ŠatrijosKalnas“ taip pat priklauso šiai klasei.

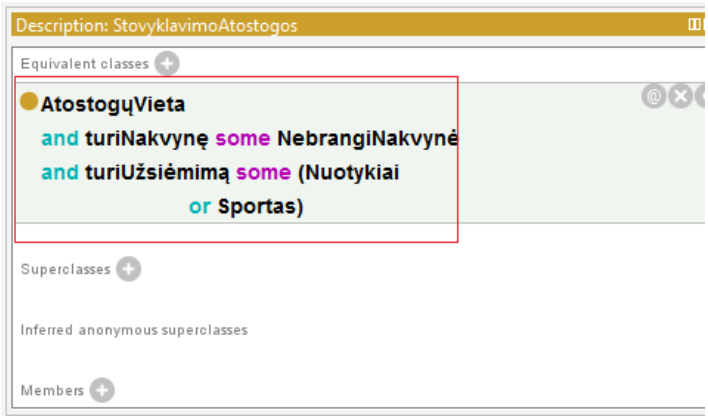


- Subklasė „ŪkininkoSodyba“ neturi jokių atributų.

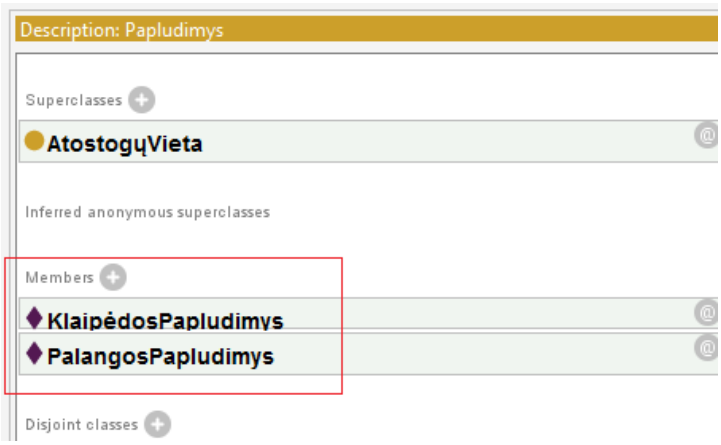
- „AtostogosNebrangiameViešbutyje“ klasė turi atributą „turiNakvynę“ kurio reikšmė yra klasės instancija ekvivalenti klasėms „NebrangiNakvynė“ ir „Viešbutis“.



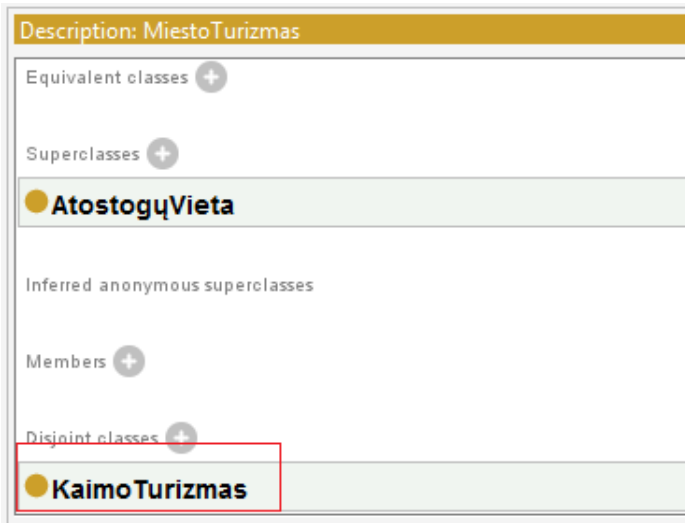
- Subklasė „StovyklavimoAtostogoas“ turi 2 atributus: „turiNakvynę“ ir „turiUžsiėmimą“. „turiNakvynę“ atributas yra lygus klasės „NebrangiNakvynė“ instancijai. Antrasis atributas gali turėti „Nuotykiai“ arba „Sportas“ klasių instanciją.



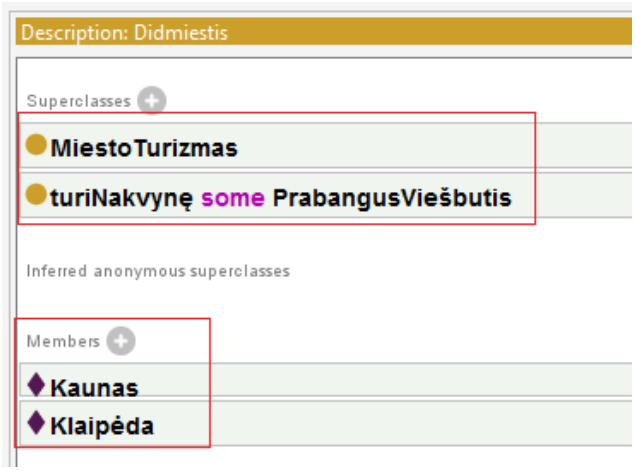
- Klasė „Papludimys“ turi 2 individus „KlaipėdosPapludimys“ ir „PalangosPapludimys“.



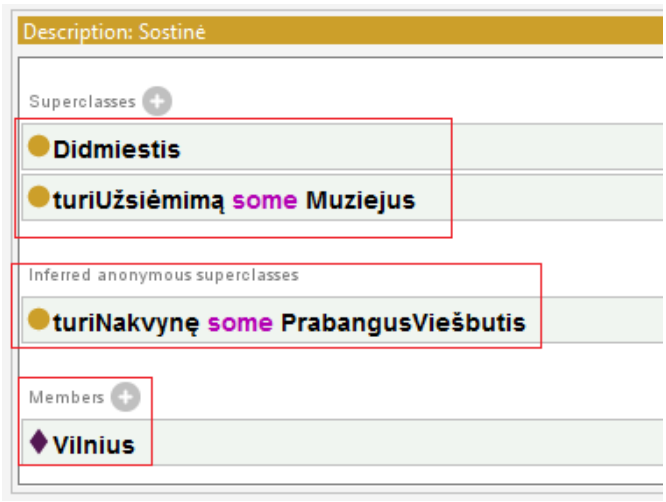
- „MiestoTurizmas“ klasė turi 2 subklases – „Miestelis” ir “Didmiestis”. “Didmiestis” turi dar vieną subklasę „Sostinė“. „MiestoTurizmas“ klasės negalima jungti su klase „KaimoTurizmas“.



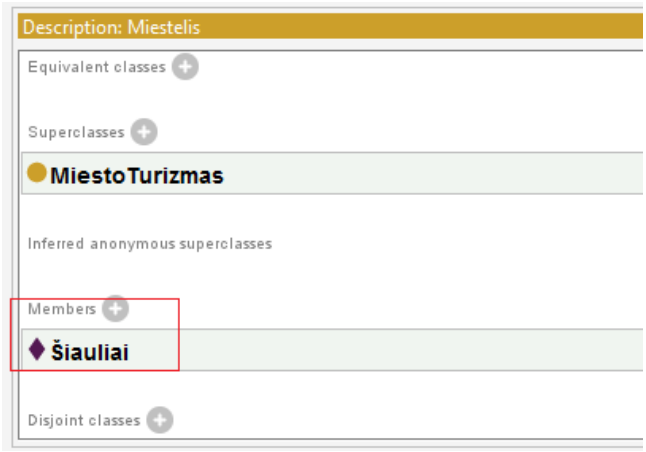
  - „Didmiestis“ atributas „turiNakvynę“ yra „PrabangusViešbutis“ klasės instancija. Taip pat ši subklasė turi 2 individus „Kaunas“ ir „Klaipėda“.



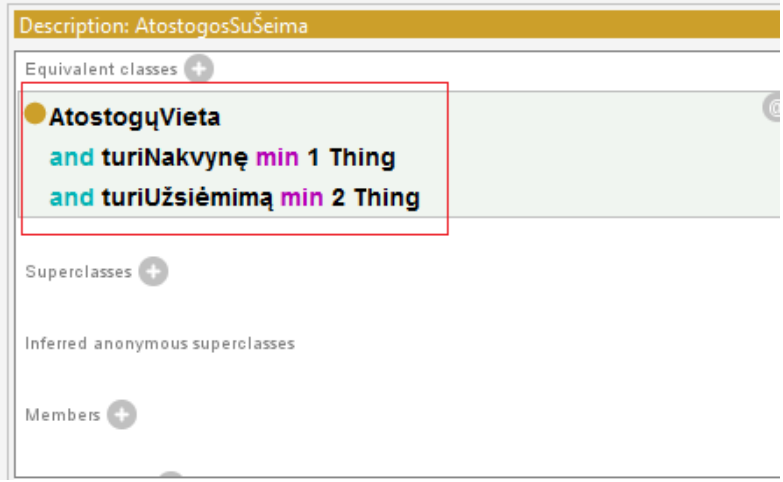
- „Didmiestis“ subklasė „Sostinė“ iš savo tėvinės klasės (superklasės) paveldėjo atributą „turiNakvynę“ bei turi dar vieną savąjį „turiUžsiėmimą“ kuris gali būti lygus „Muziejus“ klasei. „Sostinė“ turi ir vieną individą „Vilnius“.



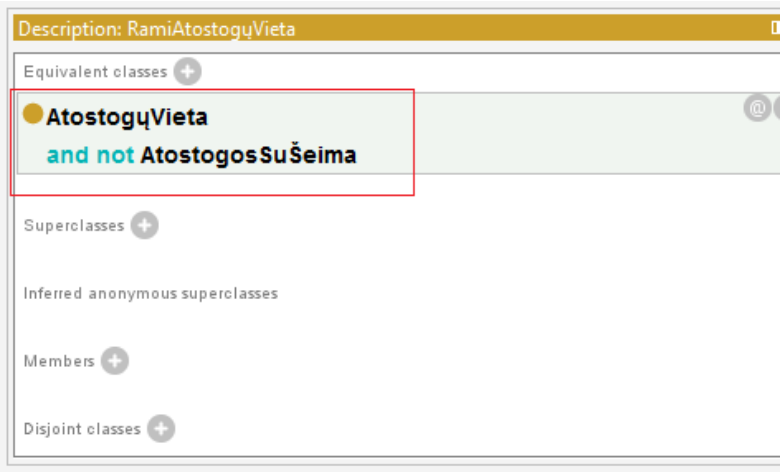
- „Miestelis“ klasė turi vieną individą „Šiauliai“.



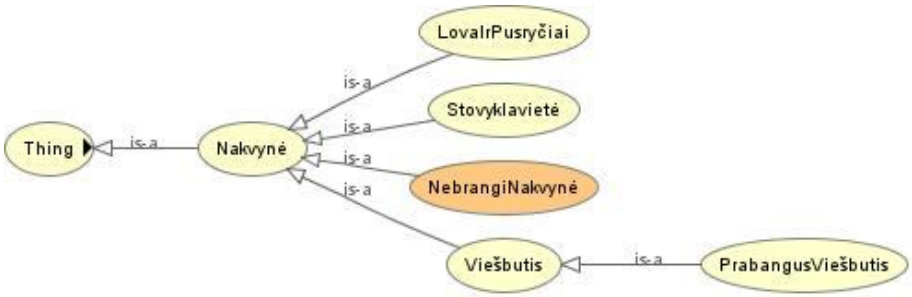
- „AtostogosSuŠeima“ klasė yra ekvivalenti „AtostogųVieta“ klasei ir turi 2 atributus: „turiApgyvendinimą“, kurio reikšmė gali būti bet kuri klasės įeinančios į šio atributo savybės reikšmių diapazoną instancija ir turėti mažiausiai vieną reikšmę; „turiUžsiėmimą“, kurio reikšmė gali būti bet kuri klasės įeinančios į minėtojo atributo savybės reikšmių spektrą instancija ir turėti mažiausiai 2 reikšmes.



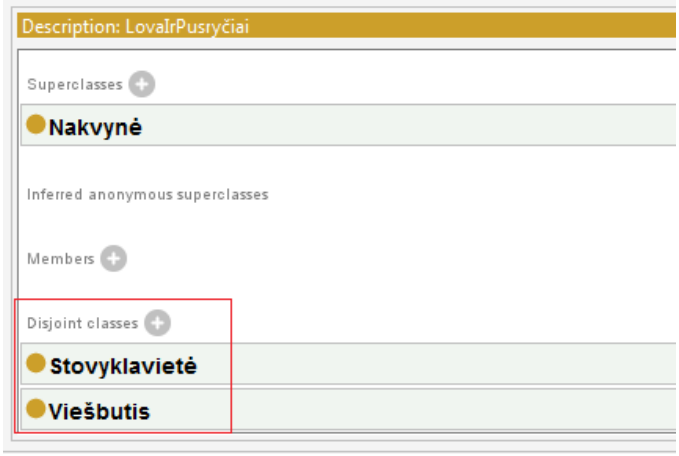
- Paskutinė „AtostogųVieta“ subklasė yra „RamiAtostogųVieta“. Ji yra ekvivalenti klasei „AtostogųVieta“ ir neekvivalenti klasei „AtostogosSuŠeima“.



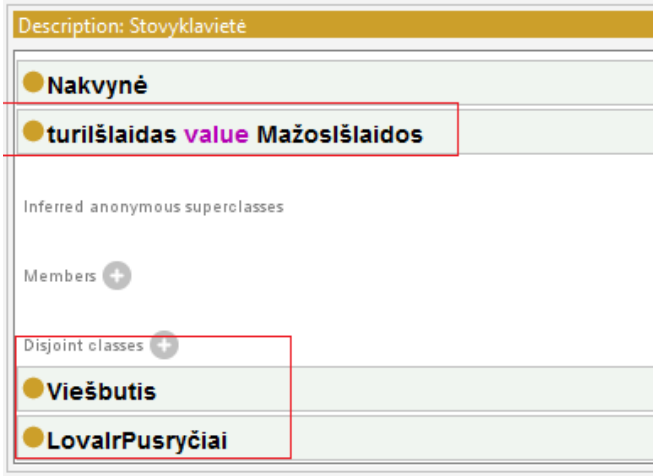
Trečioji „Thing“ klasės subklasė yra „Nakvynė“. Ji turi 4 subklases, kurių viena – „Viešbutis“ turi dar vieną subklasę „PrabangusViešbutis“.



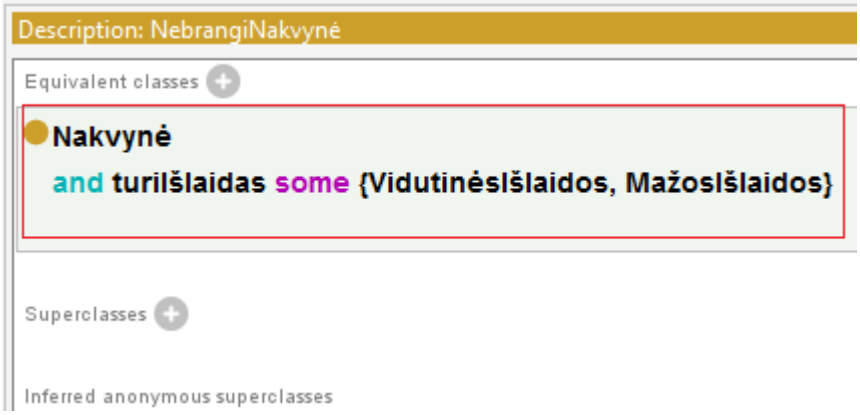
„LovaIrPusryčiai“ yra pirmoji „Nakvynė“ klasės subklasė. Ji negali būti jungiama su klasėmis „Stovyklavietė“ ir „Viešbutis“.



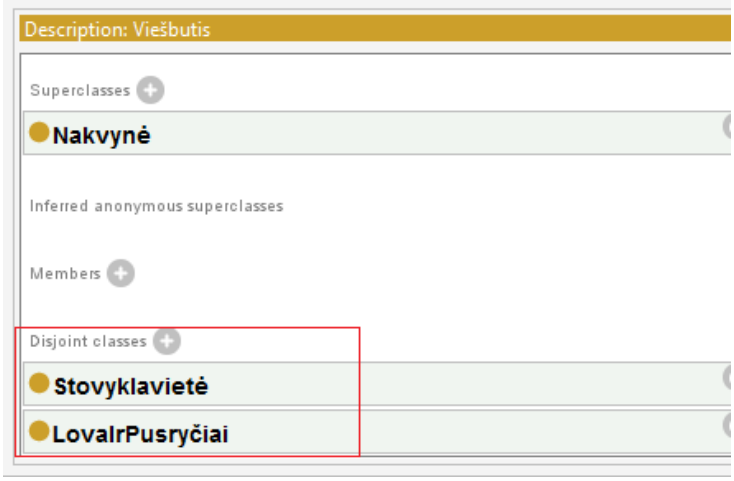
Kita „Nakvynė“ subklasė yra „Stovyklavietė“. Jos atributo „turiIšlaidas“ reikšmė yra „MažosIšlaidos“. Taip pat ši klasė negali būti jungiama su klasėmis „Viešbutis“ bei „LovaIrPusryčiai“.



„NebrangiNakvynė“ klasė yra ekvivalenti „Nakvynė“ klasei, tačiau tik tuo atveju, jei instancija turi atributą „turiIšlaidas“ kuris yra lygus klasei „VidutinėsIšlaidos“ arba klasei „MažosIšlaidos“.

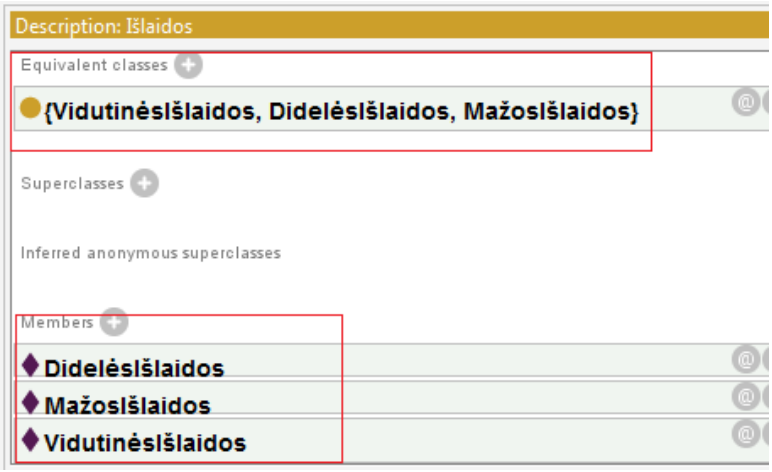


Paskutinė „Nakvynė“ klasės subklasė yra klasė „Viešbutis“. Ji turi vieną subklasę „PrabangusViešbutis“ bei negali būti jungiama su klasėmis „Stovyklavietė“ ir „LovaIrPusryčiai“.



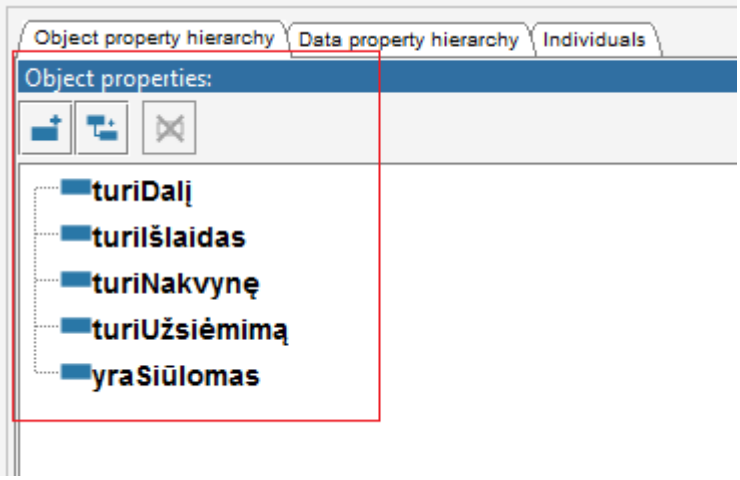
„Viešbutis“ subklasė „PrabangusViešbutis“ turi atributą „turiIšlaidas“ kurio reikšmė „DidelėsIšlaidos“ klasės instancija ir individą „KeturiMetųLaikai“.

Paskutinė ontologijos pagrindinės klasės subklasė yra „Išlaidos“. Ji neturi subklasių. Tačiau turi 3 individus kuriems ši klasė yra ekvivalenti.



7.1.2. Objektų atributų analizė

Kiekvienas objekto atributas turi klasę ar klasių grupę, kuriose jis gali būti vartojamas, savo reikšmių diapazoną bei charakteristikas.



Ši ontologija turi 6 atributus:

- „turiNakvynę“
    - Gali būti vartojamas „AtostogųVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Nakvynė“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
- „turiDalį“
    - Gali būti vartojamas „AtostogųVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „AtostogųVieta“ klasė ir jos subklasės.
    - Turi pereinamumo charakteristiką.
- “turiReitingą“
    - Gali būti vartojamas „Nakvynė“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Išlaidos“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
- „turiUžsiėmimą“
    - Gali būti vartojamas „AtostogųVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Užsiėmimai“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
    - Priešingas atributas „yraSiūlomas“ atributui.
- „yraSiūlomas“
    - Gali būti vartojamas „AtostogųVieta“ klasėje ir jos subklasėse.
    - Į reikšmių diapazoną įeina „Užsiėmimai“ klasė ir jos subklasės.
    - Neturi jokių charakteristikų.
    - Priešingas atributas „turiUžsiėmimą“ atributui.

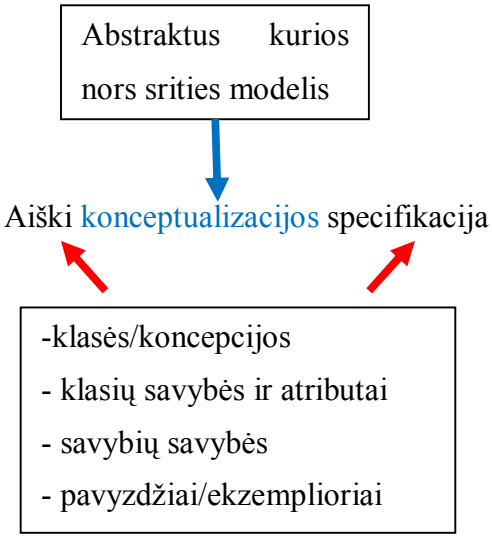
7.1.3. Individų analizė

- „Dargaičiai“
    - „KaimoTurizmas“ klasei priklausanti individualybė.
- „DruskininkųNacionalinisParkas“
    - „NacionalinisParkas“ klasei priklausantis individas.
- „VidutinėsIšlaidos“
    - „Išlaidos“ klasei priklausantis individas.
    - Yra klasių „DidelėsIšlaidos“ ir „MažosIšlaidos“ priešingybė.
- „Jūrvingė“
    - „KaimoTurizmas“ klasei priklausintis individas.

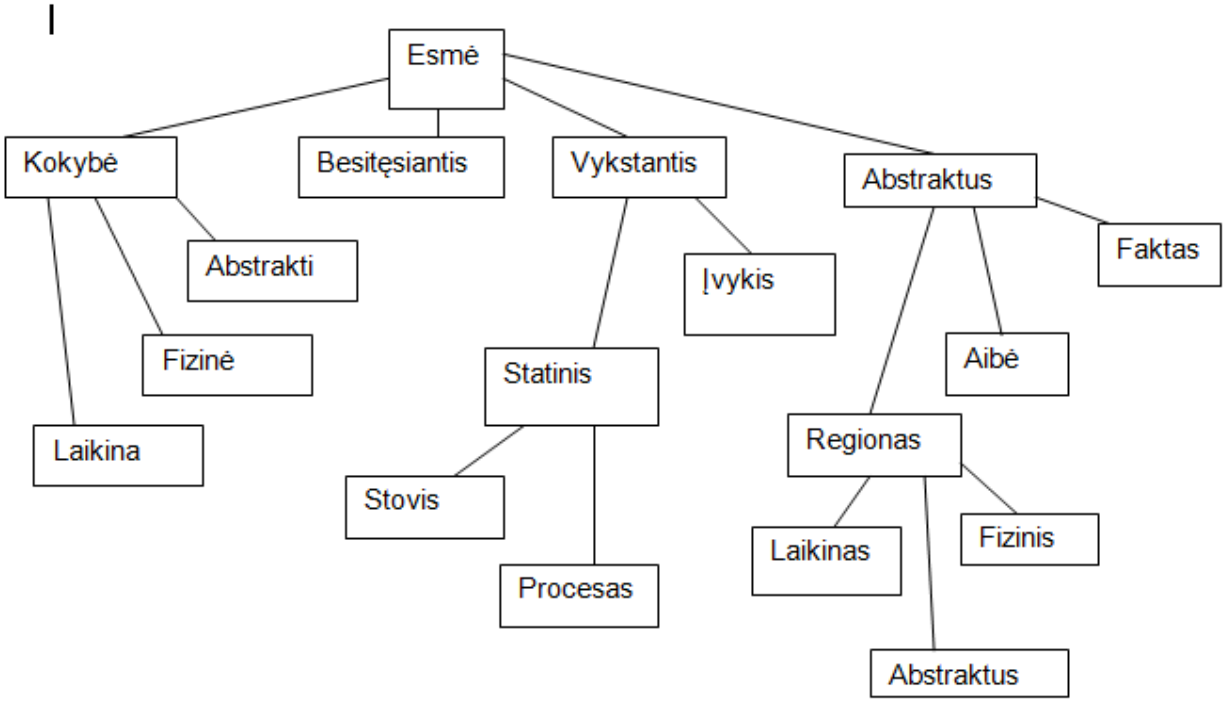
- „Kaunas“
    - „Didmiestis“ klasei priklausantis individas.
- „KeturiMetųLaikai“
    - „PrabangusViešbutis“ klasei priklausantis individas.
- „Klaipėda“
    - „Didmiestis“ klasei priklausantis individas.
    - Turi 3 duomenų atributus:
        - „turiNakvynę“ su reikšme „KeturiMetųLaikai“
        - „turiDalį“ su reikšme „PalangosPapludimys“
        - „turiDalį“ su reikšme „KlaipėdosPapludimys“
- „KlaipėdosPapludimys“
    - „Papludimys“ klasei priklausantis individas.
- „PalangosPapludimys“
    - „Papludimys“ klasei priklausantis individas.
- „DidelėsIšlaidos“
    - „Išlaidos“ klasei priklausantis individas.
    - Yra klasių „MažosIšlaidos“ ir „VidutinėsIšlaidos“ priešingybė.
- „MažosIšlaidos“
    - „Išlaidos“ klasei priklausantis individas.
    - Yra klasių „DidelėsIšlaidos“ ir „VidutinėsIšlaidos“ priešingybė.
- „Vilnius“
    - „Sostinė“ klasei priklausantis individas.

- „ŠatrijosKalnas“
    - „NacionalinisParkas“ klasei priklausantis individas.
- „Šiauliai“
    - „Miestelis“ klasei priklausantis individas.

Paveiksliukas NR. 10



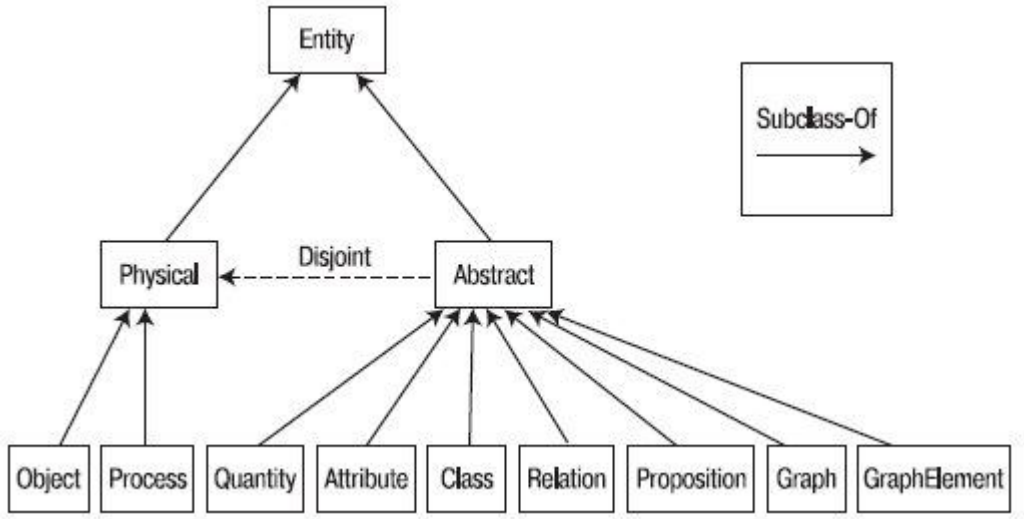
Paveiksliukas NR. 11



7 pvksl. Aukščiausi DOLCE hierarchijos lygiai[38]

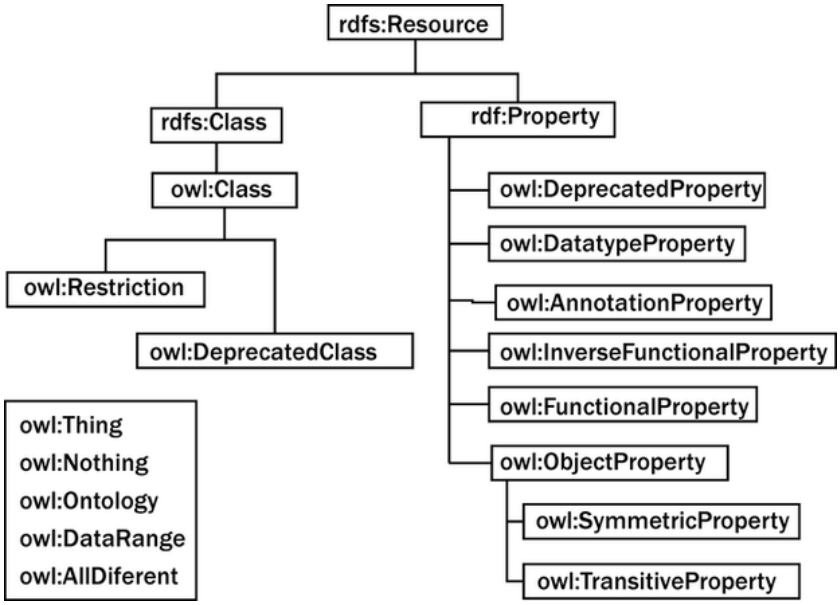
[38] Šaltinis (išverstas) (norint peržiūrėti būtina užsiregisturoti): http://www.intuit.ru/department/expert/ontoth/4/

Paveiksliukas NR. 12



8 pvksl. Aukščiausio lygio SUMO ontologijos [39]

Paveiksliukas NR. 13

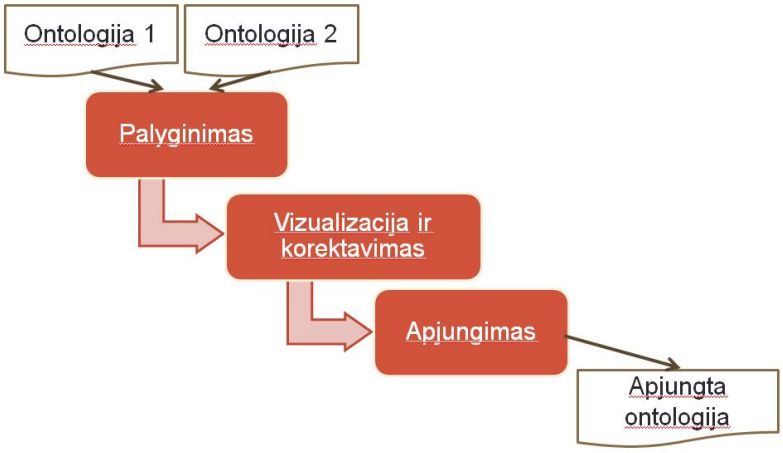


9 pvksl. Žinių vaizdavimo ontoligija OWL kalbai[40]

[39] Šaltinis: http://www.intuit.ru/department/expert/ontoth/2/2_4.jpg

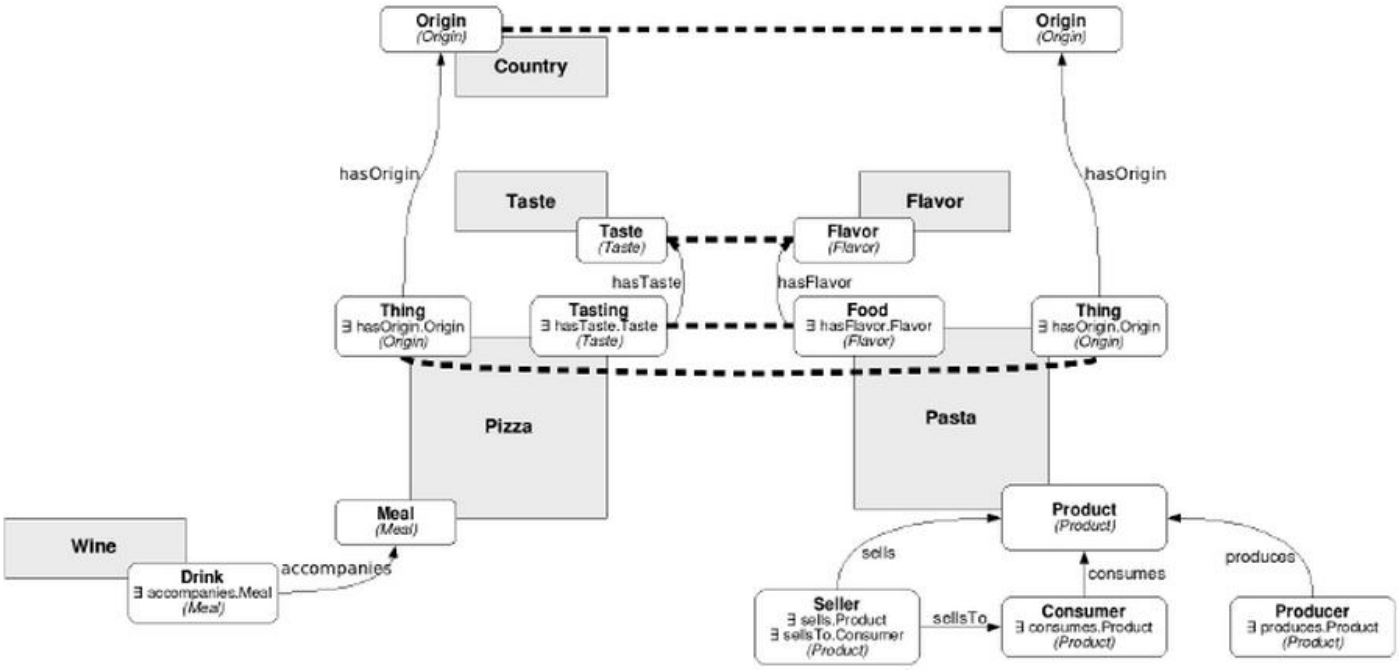
[40] Šaltinis: http://www.intuit.ru/department/expert/ontoth/2/2_3.png

Paveiksliukas NR. 14



17 vksl. Dviejų ontologijų apjungimo prototipas

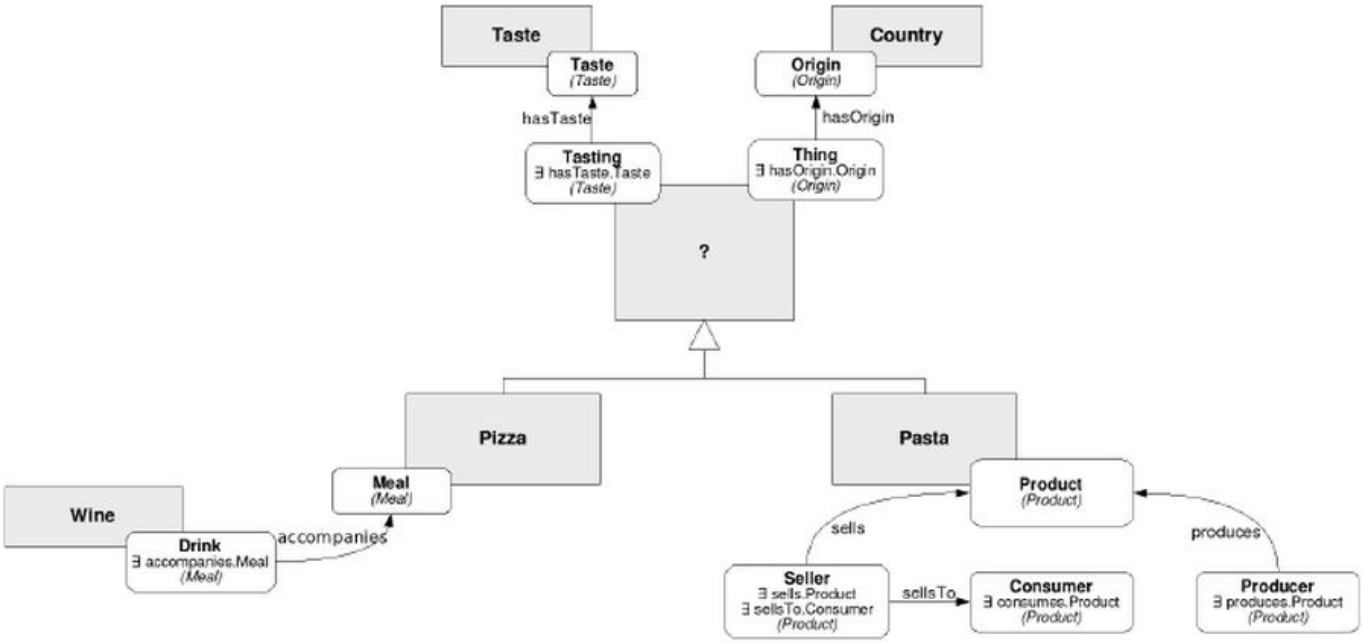
Paveiksliukas NR. 15



18 pvksl. Picos ir Pastos ontologijos su vaidmenų modeliais[41]

[41] Šaltinis: http://mp.binaervarianz.de/ontology_composition_gb_slides.pdf

Paveiksliukas NR. 16



19 pvksl. Picos ir Pastos ontologijų sujungimas panaudojant vaidmenis[42]

Lentelė NR. 1

J. Sowa pasiūlytos aukščiausio antologijų lygio kategorijos:

Fizinis			**Abstraktus**	
	Ištęstinis	**Vykstantis**	**Ištęstinis**	**Vykstantis**
Nepriklausomas	Objektas	Procesas	Schema	Skriptas
Sabtykinis	Sąjunga	Dalyvavimas	Aprašymas	Istorija
Netiesioginis	Struktūra	Situacija	Priežastis	Tikslas

1 lentelė. J. Sowa pasiūlytos aukščiausio antologijų lygio kategorijos

[42] Šaltinis: http://mp.binaervarianz.de/ontology_composition_gb_slides.pdf