KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

ARTŪRAS KAUNAS

OWL 2 ONTOLOGIJŲ TRANSFORMAVIMO Į SBVR VEIKLOS ŽODYNĄ IR VEIKLOS TAISYKLES METODIKA

Baigiamasis magistro projektas

Vadovė
lekt. mag.Gintarė Kriščiūnienė

Konsultantė
prof. L. Nemuraitė

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

OWL 2 ONTOLOGIJŲ TRANSFORMAVIMO Į SBVR VEIKLOS ŽODYNĄ IR VEIKLOS TAISYKLES METODIKA

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

Vadovė
lekt. mag.Gintarė Kriščiūnienė
2016-05-23

Recenzentė
doc. dr. Rita Butkienė
2016-05-23

Projektą atliko
Artūras Kaunas
2015-05-23

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

(Fakultetas)

Artūras Kaunas

(Studento vardas, pavardė)

Informacinių sistemų inžinerijos studijų programa, 621E15001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „OWL 2 ONTOLOGIJŲ TRANSFORMAVIMO Į SBVR VEIKLOS ŽODYNĄ IR VEIKLOS TAISYKLES METODIKA“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 16 m. Gegužės 23 d.

Kaunas

Patvirtinu, kad mano, **Artūro Kauno**, baigiamasis projektas tema „OWL 2 ONTOLOGIJŲ TRANSFORMAVIMO Į SBVR VEIKLOS ŽODYNĄ IR VEIKLOS TAISYKLES METODIKA“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Kaunas, Artūras. *Methodology For Transforming OWL 2 Ontologies Into SBVR*: Master's thesis in Information Systems Engineering / supervisor lekt. mag. Gintarė Kriščiūnienė. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Informatics Engineering, Technology Science

Key words: *ontology, business rules, vocabulary, transformation, OWL, SBVR.*

Kaunas, 2016. 65 p.

SUMMARY

Creating business vocabularies and rules is a difficult process, but ontologies is a different case with its strict structure policies and specialised tools with logic and syntax checkers. As a result of this there‘s a wider range of ontology examples created. So it‘s wise not to start creating business rules and vocabularies from scratch, but take an existing ontology and transform it to business vocabulary and rules. But there is no reliable solution how to achieve this transformation. This masters thesis examines the possibility of transforming a particular subject area ontology into SBVR business vocabularies and rules and describes the proposed methodology for transforming OWL2 described ontology into SBVR. The methodology allows the transformation of OWL syntax elements without individuals. Activity diagrams and functional equations were used for describing the methodology in formal manner. The created methodology was successfully implemented in a prototype tool. A conclusion was drawn from the experimental results, that if the ontology element set does not exceed the defined rules set and the ontology itself is not ambiguous, the created methodology can be held as correct and can be used in business and information system creation processes.

Kaunas, Artūras. OWL 2 ontologijų transformavimo į sbvr veiklos žodyną ir veiklos taisykles metodika. Magistro baigiamasis projektas / vadovė lekt. mag. Gintarė Kriščiūnienė; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Informatikos inžinerija, technologijos mokslai

Reikšminiai žodžiai: *ontologija, veiklos taisyklės, žodynai, transformacija, OWL, SBVR.*

Kaunas, 2016. 65 p.

SANTRAUKA

Veiklos žodyno ir taisyklių kūrimas – sudėtingas procesas. Tuo tarpu ontologijos turi daug griežčiau apibrėžtą struktūrą ir specializuotus įrankius logikos ir sintaksės tikrinimui, todėl egzistuoja daug išsamesnių ir platesnių sričių ontologijų pavyzdžiu. Dėl to tikslinga veiklos žodyną ir taisykles nepradėti kurti nuo nulio, o transformuoti jau esamą ontologiją. Tačiau dar nėra sukurtų sprendimų, kurie leistų gauti veiklos žodyną ir taisykles iš sukurtos ar jau egzistuojančios ontologijos. Magistro baigiamajame darbe iškeltas tikslas sukurti metodiką, kurią naudojant, būtų galima transformuoti OWL2 ontologijas į veiklos žodynus ir taisykles, bei sukurti eksperimentinį prototipą. Darbe analizuojamos tam tikros dalykinės srities ontologijų transformavimo galimybės į SBVR veiklos žodynus ir taisykles, aprašyta siūloma OWL į SBVR transformavimo metodika. Algoritmas leidžia transformuoti OWL schemą be individų elementų. Formaliam sudarytos metodikos veikimo principų aprašymui panaudotos veiklos diagramos ir funkcinės lygtys. Sukurta transformavimo metodika buvo panaudota prototipo programinėje įrangoje ir patikrintas jos veikimas. Iš eksperimento metu gautų rezultatų galima teigti, kad kai elementų aibė neviršija aprašytų taisyklių transformacijos aibės ir transformuojama ontologija yra korektiška, transformacijos metodika veikia teisingai ir ją galima naudoti verslo ir sistemų kūrimo procesuose.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Ontologija – tai esminių dalykinės srities esybių ir jų semantinių ryšių aprašymas.

OWL (*angl. Ontology web language*) OWL (*angl. Ontology web language*) – semantinio žymėjimo kalba, skirta ontologijoms vaizduoti internete.

SBVR (*angl. Business Rules Generation from Natural Language Specification*) – Veiklos žodyno ir veiklos taisyklių semantikos standartas.

XML (*angl. Extensible Markup Language*) yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba. Pagrindinė XML kalbos paskirtis yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingo tipo sistemų, dažniausiai sujungtų internetu.

RDF (*angl. Resource Description Framework Site Summary*)

OMG grupė (*angl. Object Management Group*) – atvira tarptautinė, ne pelno siekianti organizacija leidžianti standartus ir įvairius sprendimus verslui.

W3C (*angl. World Wide Web Consortium*) – saitynui skirtų programinės įrangos standartų leidimo konsorciumas.

XMI (*angl. XML Metadata interchange*) yra OMG standartas metaduomenų apsikeitimui naudojant XML.

XSD (*angl. XML sheme definition*) xml schemos aprašas.

ĮVADAS

Darbą rašė Kauno technologijos universiteto studentas Artūras Kaunas, studijuojantis magistratūros INFORMACINIŲ SISTEMŲ INŽINERIJOS studijų programą Informatikos fakultete.

Darbo problematika ir aktualumas

Veiklos žodyno ir taisyklių semantika (SBVR) suteikia lingvistinį būdą semantiškai apibūdinti verslo koncepcijas ir nurodyti veiklos taisykles [1]. Lingvistinis žinių aprašymo metodas leidžia verslo žinias ištransliuoti per teiginius, o ne diagramas. Žinome, kad diagramos ir schemos puikiai tinka aprašant organizacines struktūras ar elgsenas, tačiau jos yra nepraktiškos kaip pagrindinė priemonė nustatant žodynus ir išreiškiant veiklos taisykles.

Keletas mokslinės bendruomenės narių jau atpažino, kokią naudą gali duoti SBVR ir OWL konvertavimas ir atitikimo tikrinimas [2], [3], [4], [5]. Ontologijų struktūros tikrintojai gali automatiškai padėti nustatyti verslo modelio vientisumą. Ontologijos taip pat naudojamos analitinėje dalyje kuriant programinę įrangą verslui valdyti.

Šiame tyrime sprendžiama OWL 2 transformavimo į SBVR specifikaciją problema. Naudojantis dabartinėmis priemonėmis ir įrankiais, nėra galimybės OWL 2 kalba aprašytą ontologiją transformuoti į veiklos žodyną ir veiklos taisykles užtikrinant korektiškumą ir transformacijos pilnumą. Kadangi sukurtų ontologijų aibė žymiai didesnė, nei veiklos žodynų ir veiklos taisyklių, tikslinga pasiimti jau sukurtą ontologiją ir ją transformuoti į veiklos žodyną, tokiu atveju sutaupoma daug laiko ir žmogiškųjų išteklių. Transformacijai iš SBVR į OWL sukurtas įrankis S2O, tačiau norint transformuoti ontologijas įrankio nėra, todėl reikalinga taisyklių aibė leidžianti transformuoti į SBVR.

Darbo tikslas ir uždaviniai

Sudaryti galimybes įvertinti OWL 2 ir SBVR specifikacijų tarpusavio transformacijų korektiškumą ir išsamumą, sukuriant automatizuoto eksperimentinio tyrimo informacinės sistemos prototipą, kuris leistų transformuoti dalykinės srities ontologijas, užrašytas OWL 2 kalba, į tos srities SBVR veiklos žodynais bei taisyklėmis. Šiam tikslui pasiekti iškelti uždaviniai.

Darbo uždaviniai:

1. Išanalizuoti:
   1.1. Ontologijų OWL 2 kalbos sąvokas;
   1.2. SBVR veiklos žodynų ir taisyklių sąvokas;
   1.3. OWL 2 ir SBVR elementų tarpusavio atitikimo ir transformavimo taisykles, aprašytas mokslinėse publikacijose;

1.4. Galimas realizacijos technologijas.

2. Sudaryti ontologijų į veiklos taisykles ir žodynus transformavimo metodiką, suprojektuoti ją realizuojantį prototipinį įrankį;
3. Realizuoti prototipą, leidžiantį atlikti ontologijos transformavimą į veiklos žodyną ir taisykles;
4. Atlikti eksperimentą prototipo tinkamumui įvertinti;
5. Apibendrinti tyrimo rezultatus.

Darbo rezultatai ir jų svarba

Problemai spręsti sukurta automatizuoto OWL 2 į SBVR modelių transformavimo metodiką, bei eksperimentinis prototipas kuris suteikia vartotojams galimybę transformuoti ontologiją į veiklos žodyną ir taisykles be didelių laiko išteklių, užtikrinant modelių korektiškumą ir tarpusavio atitikimą.

Darbo struktūra

Darbą sudaro keturi skyriai. Pirmame skyriuje aprašoma tyrimo tikslas, sritis, objektas, uždaviniai, esami sprendimai ir analizės išvados. Antrame skyriuje aprašomas transformacijos algoritmas, dalykinės srities modelis. Trečiame skyriuje aprašomas realizacija, naudotos technologijos, testavimo modelis. Ketvirtame skyriuje aprašomas eksperimentas ir jo rezultatas.

1. OWL 2 IR SBVR TRANSFORMAVIMO PROBLEMINĖS SRITIES ANALIZĖ

1.1. Analizės tikslas

Analizės metu siekiama išanalizuoti ontologijų OWL 2 kalbos sąvokas, taip pat lygiagrečiai analizuojamos ir SBVR veiklos žodynai ir taisyklės, siekiant sukurti tarpusavio atitikimo ir transformacijos taisykles, kurios naudojamos, kuriant įranki skirtą šių metamodelių transformacijai.

1.2. Tyrimo objektas, sritis ir problema

1.2.1. Tyrimo objektas

Tyrimo pagrindinis objektas yra ontologijų OWL 2 transformacijos į SBVR veiklos žodynus ir taisykles atlikimo procesas.

1.2.2. Tyrimo sritis

Norint įvertinti šį procesą reikia išnagrinėti ontologijų kalbą OWL 2, SBVR metamodelį, OWL 2 ontologijų ir SBVR veiklos žodynų tarpusavio transformacijas, šių transformacijų korektiškumą ir išsamumą.

1.2.3. Problema

Šiame tyrime sprendžiama OWL 2 transformavimo į SBVR specifikaciją problema. Naudojantis dabartinėmis priemonėmis ir įrankiais, sunku palyginti dviem skirtingais metamodeliais (OWL 2 ir SBVR) grindžiamas dalykinės srities specifikacijas ir užtikrinti informacijos nepraradimą pereinant nuo vieno aprašymo būdo prie kito. Dalykinės srities aprašymą kartais tikslinga pradėti nuo vieno arba kito vaizdavimo būdo – kartais tikslinga kurti ontologijas, kartais veiklos žodynus ir taisykles, o vėliau tuos aprašymus transformuoti arba į veiklos žodynus, arba į ontologijas. Šioms transformacijoms yra sukurtas įrankis SBVRToOWL2. Tačiau ontologijoms transformuoti tikslios metodikos nėra, todėl reikalingas taisyklių bei rekomendacijų rinkinys, kuris apibrėžtų kaip transformuoti OWL2 modelius į veiklos žodynus ir taisykles SBVR.

1.3. OWL 2 ir SBVR sandarų analizė

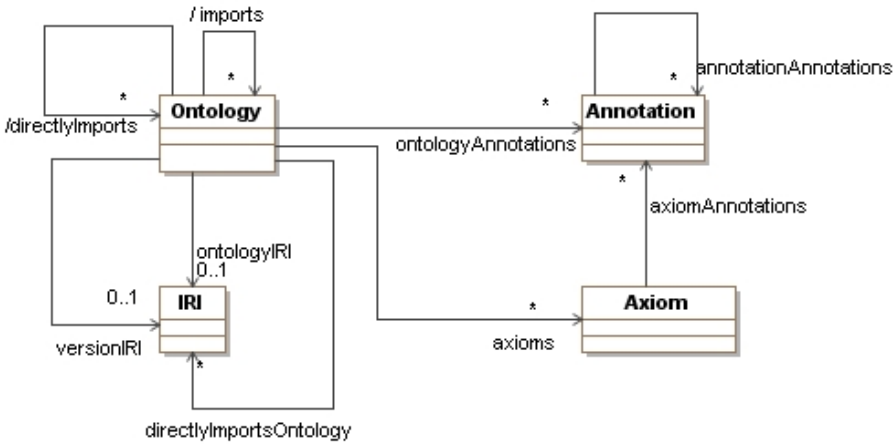
1.3.1. OWL 2

Ontologijos – tai esminių dalykinės srities esybių ir jų semantinių ryšių aprašymas. Ontologija turi daug apibrėžimų. Ontologija (didžioji „O“) yra filosofinė disciplina; ontologija (mažoji "o") yra specifinis artefaktas, sukurtas norint apibrėžti numatytą žodyno prasmę [6]. Pasak Borst, ontologija yra visuotinai pripažinta ir aiškiai išreikšta formali konceptualizacijos specifikacija (angl. *explicit and formal specification of shared conceptualization*) [7].

OWL – semantiniu žymėjimu pagrįsta tinklo ontologijų kalba, sukurta programoms kurios apdoroja informaciją, naudoti [8], o ne tik ją pateikia žmonėms. Pagrindinė paskirtis – ontologijas vaizduoti saityne (WWW). *OWL* buvo sukurta *RDF* pagrindu siekiant išplėsti semantines *RDF* galimybes, todėl pasižymi geresne technine interpretacija, nei kitos kalbos tokios kaip: XML, RDF ir RDF schemos [9]. Tai pasiekiama su papildomu žodynu ir formaliąja semantika. OWL turi tris sudėtingėjančias kalbas: OWL lite, OWL DL ir OWL full. [10]

OWL 2 – tai 2009 m. konsorciumo *WC3* patvirtintas ontologijų kalbos aprašymo standartas (šiuo metu naudojama 2012 m. versija).

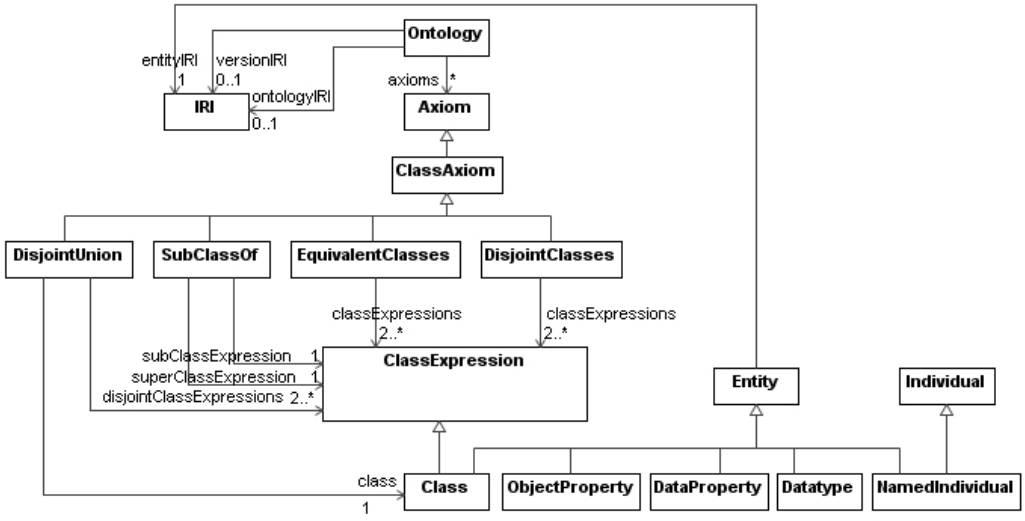
OWL 2 ontologijos struktūra pateikta 1.1 pav.. Pagrindinis *OWL2* ontologijos komponentas yra aksioma. Be aksiomų, ontologija turi anotacijas ir gali importuoti kitas ontologijas.



1.1 pav. *OWL 2* ontologijos struktūra [10]

PAGRINDINIAI *OWL2* ONTOLOGIJOS KONCEPTAI

Pagrindiniai *OWL2* konceptai, atitinkantys dalį *SBVR* konceptų, yra pavaizduoti 1.2 pav. *OWL 2* ontologija susideda iš aksiomų. Pagrindinis *OWL2* ontologijos metamodelio konceptas yra *OWL2* klasė (angl. *Class*), kuri yra *ClassExpression* poklasis.



1.2 pav. *Axioms, Entities and Individuals* poklasiai *OWL 2* metamodelyje [10]

Ontologijos ir *OWL 2* esybės (*Classes*, *ObjectProperties*, *DataProperties*, *Datatypes* ir *NamedIndividuals*) yra identifikuojamos naudojant *IRI* (angl. *Internationalized Resource Identifiers*). *IRI* skiriasi nuo *URI* (interneto adreso) tuo, kad *IRI* gali turėti bet kokius simbolius.

Kad būtų lengviau suprantama nagrinėdami ontologijas naudosimės „Kompiuterių ontologija“, aprašančia kompiuterius ir jų komponentus, bei sąryšius tarp jų. Ontologijų esybės:

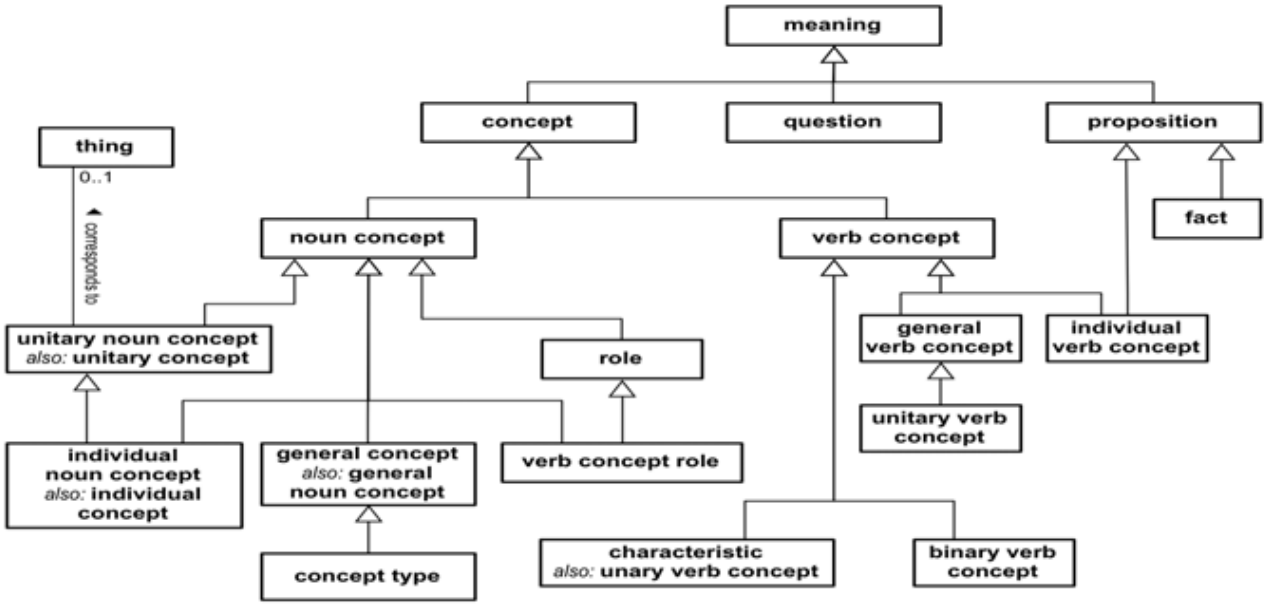
- klasės (angl. *Classes*) apibrėžiamos kaip individų aibės. Kompiuterių ontologijoje klases atitinką kompiuterių tipų klasifikacija: *stalinis*, *nešiojamasis*, *delninis*.
- individai (angl. *Individual*) vaizduoja tam tikros srities objektus, pvz: *Mac* ir *MacBook*, priklausantys atitinkamoms klasėms *stalinis* ir *nešiojamas*.
- savybės (angl. *Properties*) yra binariniai individų ryšiai, t. y., jos sujungia du individus tarpusavyje . *ObjectProperties* jungia klasių egzempliorius (individus), *DataProperties* jungia klasių egzempliorius su duomenų egzemplioriais (atributais), kurie turi tam tikrus duomenų tipus (*DataType*).

1.3.2. SBVR

Veiklos žodyno ir veiklos taisyklių semantikos standartas *SBVR* (angl. *Semantics of Business Vocabulary and Business Rules*) yra *OMG* grupės sudarytas standartas veiklos taisyklėms specifikuoti ribota natūralia kalba [1]. *SBVR* standartas apibrėžia metamodelį, kuris formaliai aprašo veiklos žodynus ir veiklos taisykles. Pirmas žingsnis, pradedant kurti informacinę sistemą, yra apsibrėžti kompiuterizuojamos veiklos žodyną ir taisykles. *SBVR* taikymas neapsiriboja vien tik verslo sprendimais, jo taikymo sritys apima bet kokią veiklą (valstybės valdymą, sveikatos apsaugą, teisę, ir t. t.).

SBVR specifikacija apibrėžia veiklos žodyno (angl. *business vocabulary*), veiklos taisyklių (angl. *business rules*) ir veiklos faktų (angl. *business facts*) metamodelį, skirtą aprašyti veiklos modeliams. Veiklos žodynai ir veiklos taisyklės užrašomi tekstu, tačiau turi ir formalų aprašymą, kuris *XMI* schemos formatu leidžia veiklos žodynus ir taisykles apdoroti kompiuteriu.

Prasmę vaizduojanti *SBVR* metamodelio dalis pateikta 1.3 paveiksle.



1.3 pav. SBVR prasmės metamodelis [1]

Veiklos žinias *SBVR* [11] susistemina į veiklos žodynus, kurie apima visus konceptus, kuriuos organizacija ar bendruomenė naudoja veikloje. Žodynai yra sudaryti iš įrašų – vienai sąvokai skiriamas vienas įrašas. Įrašas prasideda sąvokos pavadinimu, be to, sąvokai apibūdinti gali būti panaudoti laukai, aprašyti 1.2 lentelėje.

1.1 lentelė. *SBVR* žodyno elementus apibūdinantys laukai

Laukas	Aprašymas
SBVR žodyno elementas	Pirminė koncepto forma: terminas (daiktavardinis konceptas) arba veiksmažodinė forma (veiksmažodinis konceptas)
Definition	Sąvokos (koncepto) apibrėžimas formalia arba laisva žmogui suprantama kalba
Source	Koncepto šaltinis – žodynas arba dokumentas
General Concept	Bendresnis konceptas, leidžiantis sudaryti elementų apibendrinimo hierarchijas
Concept Type	Koncepto tipas – nurodoma, kai norima patikslinti koncepto tipą
Note	Pastabos
Synonym	Koncepto sinonimai
Synonymous Form	Sinoniminė forma (taikoma veiksmažodiniams konceptams)
See	Jei žodyno elementas yra sinonimas arba sinoniminė forma, čia nurodoma pirminė forma

PAGRINDINIAI *SBVR* KONCEPTAI

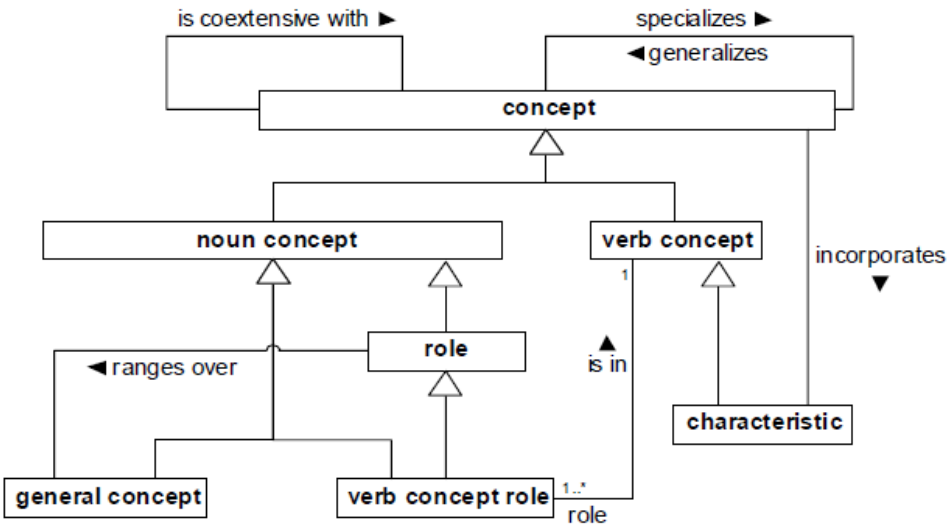
SBVR konceptai yra suskirstyti į dvi pagrindines grupes: veiklos žodynų konceptus ir veiklos taisyklių konceptus.

Veiklos žodyno konceptai (1.4 pav.):

- bendriniai konceptai (angl. *General concept*), išreiškiami bendriniais daiktavardžiais, klasifikuojančiais objektus pagal jų bendras savybes.
- Individualūs konceptai (angl. *Individual concept*) išreiškiami tikriniais daiktavardžiais.
- Veiksmažodiniai konceptai (angl. *Verb concept*) išreiškiami veiksmažodinėmis formomis, apibūdinančiomis ryšį tarp dviejų daiktavardinių konceptų. Veiksmažodiniai konceptai yra skirstomi:
  - Asociacija – tai bendriausias veiksmažodinis konceptas, kuris turi du vaidmenis (tas pats veiksmažodis gali būti naudojamas su įvairiais vaidmenimis ir kiekvienu atveju turės skirtingą prasmę (pvz., „darbuotojas dirba organizacijoje“ ir „darbuotojas dirba mokytoju“.

- Dalies-visumos (angl. *partitive* veiksmažodinis konceptas) – tai binarinis veiksmažodinis konceptas, kuris nusako, kad vienas objektas yra kito objekto dalis.
- Savybės asociacija, kuri nusako bendrinio koncepto savybę.
- Kategorizavimo fakto tipas nusako, kad vienas objekto tipas yra specifinis kito, bendresnio objekto tipo atžvilgiu.

- Vaidmenys (angl. *Role*) išreiškiami daiktavardžiais, t. y., daiktavardiniais konceptais atitinkančiais bendrinių konceptų atliekamas funkcijas, dalyvavimą ar panaudojimą tam tikroje situacijoje.
- Veiksmažodinio koncepto vaidmuo (angl. *verb concept role*) reiškia vaidmenį, kurį bendrinis konceptas atlieka ryšyje.



1.4 pav. SBVR metamodelio konceptų fragmentas [1]

1.3.3. Lyginamoji esamų sprendimų analizė

Norint sukurti korektiškas ir pilnavertes transformavimo taisykles reikalinga išnagrinėti dabartinius transformacijos sprendimus. Darbo vykdymo metu transformacijos, viešai prieinamų, programinių sprendimų nėra, todėl nagrinėjami kiti sprendimai aprašantys galimus transformacijos būdus ir taisykles.

Ne vienas mokslinės bendruomenės narys jau pastebėjo, kad ontologijos ar veiklos žodynai reikalauja papildomo funkcionalumo ir kiek vienas modelis atskirai nepatenkina poreikių, tačiau galimybė juos abu naudoti kartu pašalina šią problemą ir suteikia naujas galimybes. Todėl pradėtos kurti metodikos ir taisyklės transformuoti ontologijoms į veiklos žodynus [12], [13] ir priešingos transformacijos – veiklos žodynai ir taisyklės į ontologijas [3], [14]. Nors

transformacijų autorių tikslai skirtingi, nuo galimybės patogiau rašyti SPARQL užklausas ontologijoms [3] ar tiesiog sudaryti galimus elementų transformavimo variantus [12], [13] iki patikimo būdo atvaizduoti lietuvių kalbai semantiniame tinkle [3], [14]. Visi aprašyti transformavimo metodai pateikia tikslų taisyklių rinkinį kuriuo reikėtų vadovautis. J. Karpovič [12] savo darbe pateikia detaliausią transformacijos metodiką su 31 transformacijos taisykle, apimančia visą elementų aibę iš žodynų ir taisyklių pusės, tačiau šios taisyklės aprašo priešingos krypties transformaciją nei kuriama metodika šiame darbe. Tuo tarpu E. Reynares [13], pateikia septyniolika transformavimo taisyklių su papildomomis vidinėmis sąlygomis, tačiau darbe veiklos žodynų ir taisyklių taisyklingumui skiriama mažiau dėmesio, o tik stengiamasi padengti kuo daugiau pavienių elementų, transformacija taip pat vykdoma į priešingą pusė, nei kuriama šiame darbe. G. Kriščiūnienė (Bernotaitytė) [3] savo metodikoje pateikia keturiolika transformacijos taisyklių, kurios autorės teigimu pakankamai padengia OWL 2 schemą, kad būtų atvaizduota visa struktūra ir būtų galima kurti SPARQL užklausas remiantis suformuotu veiklos žodynu ir taisyklėmis, todėl visiškai neskiriama dėmesio teiginiams ir objektų kardinalumams. Tačiau autorė toliau plėtoja taisykles kitoje publikacijoje [14] ir taisyklių rinkinį praplečia iki dvidešimt šešių, kuriose jau labiau gilinasi, ne tik į pavienius elementus, bet ir į jų grupių galimas transformacijas. Kaip jau buvo minėta autorės pateikta metodika išsiskiria tuo, kad akcentuojamos lietuvių kalbos transformavimo galimybės. Visos publikacijos aptaria, kad metamodeliai turėtų būti sudaryti naudojant gerąsias praktikas ir turi būti taisyklingi tačiau, dėl lietuvių kalbos sudėtingumo akcentuojama, kad būtinos papildomos anotacijos nurodančios bendrines žodžių formas, tam kad būtų tiksliausiai atlikta transformacija.

1.2 lentelė. Esamų sprendimų palyginimo lentelė

	Taisyklių skaičius	Abipusė tranformacija	Reikalingas specialiai paruošta ontologija/žodynas
G. Kriščiūnienė [14]	26	Ne	Taip
J. Karpovič [12]	31	Taip	Taip
E. Reynares [13]	17	Ne	Ne

Visi analizuoti sprendimai pabrėžia, kokie svarbūs veiklos žodynai ir taisyklės bei ontologijos semantiniame tinkle. Ontologijos ir žodynai teikia daug privalumų ir abu modeliai papildo vienas kitą, suteikdami papildomų galimybių ir funkcionalumo, todėl reikalingas sprendimas realizuoti OWL schemos transformacijai. Kaip ir pastebėjo G. Kriščiūnienė [14] savo publikacijoje, J. Karpovič [12] sprendimas plačiausiai aprašo, ne tik transformaciją, tačiau ir patį

atitikimą tarp metamodelių. Bet dėl to, kad OWL 2 kalba ne tokia ekspresyvi, lyginant su SBVR, atvirkštinė transformacija nereikalauja tokio kiekio taisyklių, tačiau kaip transformacijos metodikos pagrindu remiamasi J. Karpovič taisyklių rinkiniu ir naudojamos rekomendacijos kaip turėtų būti transformuojami pavieniai elementai.

1.3.4. OWL 2 ir SBVR elementų tarpusavio atitikimo ir transformavimo taisykles

Ontologijos ir veiklos žodynai, dvi skirtingos, bet vienoda logika pagrįstos kalbos turi savo atitikmenis viena kitoje. Norint transformuoti ontologijas į veiklos žodynus su taisyklėmis, būtina atlikti palyginimo analizę suvedant atitinkančius elementus, bei atitikimo taisykles. [3]

Didžiausi iššūkiai tikrinant ontologijos atitikimą veiklos žodynui ir taisyklėms yra kalbų neatitikimas ir sinoniminės formos [5]. SBVR aprašomas naudojant vietinę kalbą, todėl šiuo atveju, naudojant lietuvių kalbą transformacija yra itin sudėtingas procesas. Norint transformuoti ontologijas parašytas lietuvių būtina naudoti papildomus elementus komentarus (angl *owl:label*), kuriuose nurodoma lietuviška elementų reikšmė.

Šios modeliavimo kalbos taip pat išsiskiria savo palaikomų kintamųjų tipų aibe, nes SBVR palaiko tik sveikąsias reikšmes, trupmeninius skaičius ir tekstines vertes. Tuo tarpu OWL 2 pasižymi plačia aibe kintamųjų tipu, įskaitant ir XSD.

1.2 lentelėje aprašomi esybių atitinkančių tarp SBVR ir OWL 2 metamodelių. Elementai nėra vienareikšmiai atitikmenys, jie labiau atspindi atitikimo galimybę, nes SBVR gali lanksčiau apibūdinti bet kokią veiklą ar daiktą, nei OWL, nes nėra taip griežtai apibrėžta.

1.3 lentelė. SBVR ir OWL2 atitikmenų matrica **[15]**

OWL2	**SBVR**
Ontology, entity IRI, AnnotationLanguage	vocabulary, namespace URI, language
Class	general concept
NamedIndividual, ClassAssertion	individual concept, classification *individas, klasifikacija*
ObjectProperty	Association *asociacija*
DataProperty	property association *savybių asociacija*
SubDataProperty	property association hierarchy *savybių asociacijos hierarchija*
DataProperty	Characteristic *charakteristika*

OWL2	**SBVR**
ObjectPropertyAssertion	Fact *faktas*
SubClassOf	Categorization *kategorizacija*
SubClassOf, ObjectAllValuesFrom	necessity statement *būtinumo pareiškimas*
SubClassOf, ObjectSomeValuesFrom	necessity statement *būtinumo pareiškimas*
SubClassOf, ObjectMinCardinality	necessity statement with at-least-n quantification *būtinumo pareiškimas su bent-jau-n kiekybės nurodymu*
SubClassOf, ObjectSomeValuesFrom	necessity statement with existential quantification *būtinumo pareiškimas su egzistenciniu kiekybės nurodymu*
SubClassOf, ObjectMaxCardinality	necessity statement with at-most-n quantification *būtinumo pareiškimas su daugiausia-n kiekybės nurodymu*
SubClassOf, FunctionalObjectProperty	necessity statement with at-most-one quantification *būtinumo pareiškimas su vienu kiekybės nurodymu*
SubClassOf, ObjectExactCardinality	necessity statement with exactly-n quantification *būtinumo pareiškimas su tiksliai-n kiekybės nurodymu*
SubClassOf, ObjectMinCardinality, ObjectMaxCardinality	necessity statement with numeric range quant *būtinumo pareiškimas su kiekybiniais rėžiais*
SubClassOf, DataExactCardinality	concept incorporates characteristic *konceptas turi savybes*
EquivalentClasses	association 'concept is_coextensive_with concept' *asociacija, konceptas lygiavertis su*
EquivalentClasses, SubClassOf, DisjointUnion	Segmentation *kategorizacija*
ObjectProperty, InverseObjectProperty	verb concept, inverse verb concept *veiksmažodinis konceptas, atvirkščias veiksmažodinis konceptas*
SubObjectPropertyOf	verb concept categorization

OWL2	SBVR
	veiksmažodinio koncepto paveldėjimas
ObjectHasSelf	purely reflexive verb concept
TransitiveObjectProperty	transitive verb concept
ObjectProperty, SubObjectPropertyOf(ObjectProperty partitive_object_property)	partitive verb concept
Class, SubclassOf, EquivalentClasses	verb concept role, verb concept

1.4. Realizacijos technologijų analizė

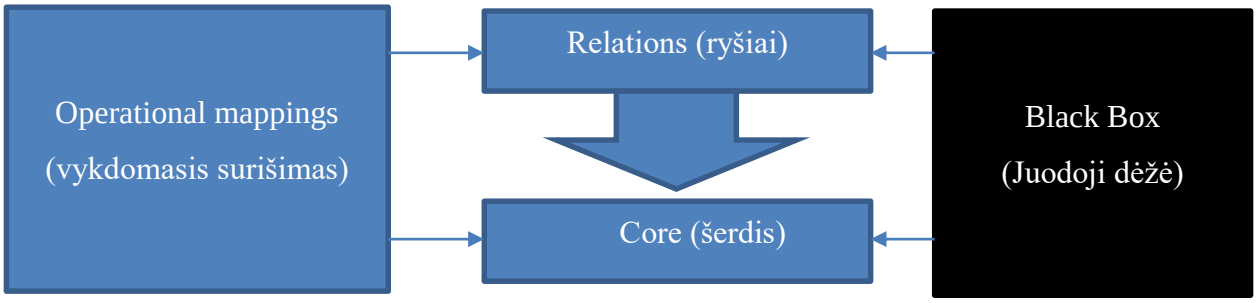
Norint transformuoti šiuos du meta modelius reikalinga automatizuota transformacijų kalba galinti, padengti sukurtas taisykles. Palyginimui paimtos dvi rinkoje plačiai naudojamos būtent transformacijoms sukurtos kalbos:

- QVT (angl Query/View/Transfromation)
- ATL (angl ATLAS transformation language)

1.4.1. QVT transformacijų kalba

Modelių transformacija yra pagrindinis metodas, naudojamas modeliais grįstoje architektūroje [16]. Kaip matyti iš pavadinimo QVT (*angl. query/view/transformation*), OMG standartas apima transformacijas, rodinius ir užklausas kartu. Modelio užklausos ir modelio rodiniai(views) gali būti vertinami kaip specialios rūšies modelio transformacijos. Modelio transformacija yra programa, kuri veikia naudodama kitus modelius.

Qvt standartas apibrėžia tris modelių transformacijos elementus. Visi jie veikia Meta-Object Facility (MOF) 2,0 meta modeliu; transformacijos šerdis apibūdina, kokie meta modeliai naudojami transformacijoje. Qvt standartas integruoja OCL 2.0 standartą ir išplečia jį su būtinomis



funkcijų.

1.5. pav QVT struktūrinė schema

Qvt-vykdomoji(angl *QVT-Operational*) kalba yra kalba skirta rašyti vienakryptes transformacijas tarp skirtingų meta modelių.

Qvt-Santykių(angl *QVT-Relations*) sluoksnis yra deklaratyvi kalba, skirta leisti tiek vienkryptes tiek dvikryptes transformacijas. Transformacija įkūnija nuoseklų ryšį tarp dviejų meta modelių. Šios transformacijos gali būti vykdomos tikrinimo rėžimu (angl *checkonly*), tada tikrinami abu modeliai ir grąžinama informacija tik ar jie atitinka vienas kitą (true/false).

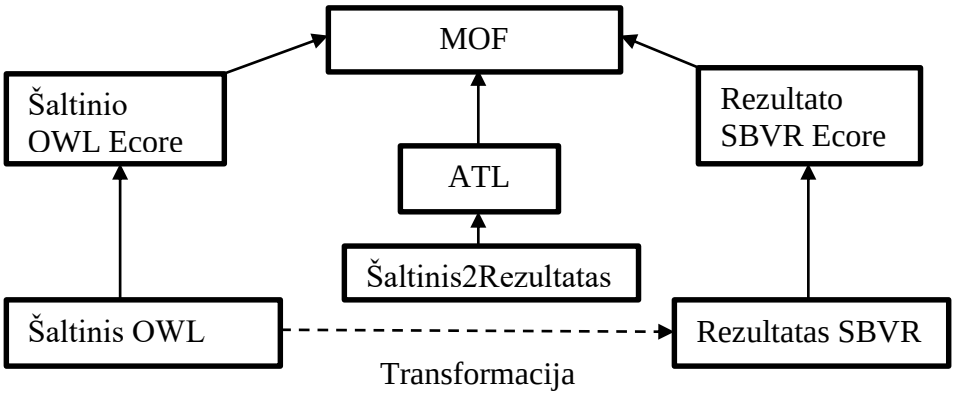
Qvt šerdis (angl *Core*) aprašo transformuojamų meta modelių struktūrines savybes elementus ir sąryšius tarp jų. Transformacijai būtini du šerdies elementai aprašantys abu meta modelius. Šiuo atveju naudojama SBVR.core ir OWL.core aprašai, kuriuos suteikia OMG korporacija.

Galiausiai yra mechanizmas vadinamas Qvt-juodoji dėžė (angl BlackBox), kuri vaizduoja kitus elementus už transformacijos ribų išreikštus kitomis kalbomis (pavyzdžiui XSLT arba XQuery), kurie naudojami kaip pagalbiniai įrankiai transformacijoms atlikti.

Nors QVT ir aprėpia daug transformavimo galimybių, ji neaprėpia visko, kas laikoma transformacijomis. Pavyzdžiui QVT nepalaiko jokios transformacijos kai verčiama iš arbą į tekstinį modelį, nes kiek vienas QVT transformuojamas modelis turi būti aprašomas MOF 2.0 modeliu.

1.4.2. ATL transformacijų kalba

ATL - ATLAS transformavimo kalba (angl. *ATLAS Transformation Language*) yra modelių transformavimo kalba ir įrankių rinkinys, kuris sukurtas ir naudojamas Eclipse aplinkoje. Ši transformavimo kalba buvo sukurta kaip atsakas į QVT stengiantis praplėsti jau esamos kalbos funkcionalumą ir panaudojamumą. Taip pat kaip ir QVT, ATL naudoja MOF ir Ecore meta modeliai transformuojamų modelių apibūdinimams. Todėl tiek šaltinis, tiek rezultatas privalo turėti Ecore aprašymus, ko pasekoje ATL taip pat negali transformuoti tekstinių modelių.



1.6 pav. ATL transformacijos schema

Dėl to kad ATL yra bandymas praplėsti QVT kalbą, joje transformacijų sluoksnis taip pat aprašomas deklaratyvia kalba, skirta leisti tiek vienkryptes tiek dvikryptes transformacijas ir aprašo ryšį tarp dviejų meta modelių. Tačiau siekiant padaryti transformacijų kalbą universalesne ir įgyvendinti moduliškumą, ALT atsisakė transformacijos validavimo ir tie patys elementai gali būti transformuojami n kartų.

1.4.3. ATL ir QVT palyginimas

Norint palyginti transformacijų kalbas reikia išsikelti tam tikrus kriterijus pagal kuriuos lyginamos kalbos: darbinė aplinka, naudojimo paprastumas, dokumentacija.

Darbinė aplinka. Abi transformacijų kalbos vykdomos Eclipse aplinkoje. Tiek QVT, tiek ATL turi savo įskiepius programinio kodo redagavimui, tačiau QVT programinio kodo atpažinimas turi platesnį funkcionalumą ir atpažysta netik vardines sritis, bet ir išorinį naudojamą programinį kodą.

Naudojimo paprastumas. QVT kalba transformuojant kiek vieną elementą vykdo tikrinimą ar elementas dar netransformuotas ir apsaugo nuo perteklinio elementų kiekio, kuris gali padaryti galutinį rezultatą nepanaudojamą. ATL transformacijų kalba šio tikrinimo neturi ir esant neefektyviam panaudojimui kuria perteklinius elementus.

Dokumentacija. QVT kalba pasižymi panašumu į Java programavimo kalbą, todėl jai patogiau ir lengviau galima rasti pavyzdžių. Abi kalbos turi dokumentacijas, tačiau QVT dokumentacija pateikiama oficialioje specifikacijoje, kai tuo tarpu ATL naudoja Wiki puslapį, kuriame lyginant su QVT specifikacija yra sudėtingiau naviguoti, nes informacija iškaidyta per aibę šaltinių.

Po realizacijos technologijų analizės nuspręsta, kad prototipinis įrankis bus realizuotas QVT transformacijų kalba, dėl jo geresnės darbinės aplinkos, lengvesnio panaudojimo ir platesnės patogiau pateiktos dokumentacijos.

1.5. Tyrimo objekto naudotojų analizė

Tyrimo rezultato, OWL 2 į SBVR transformavimo informacinės sistemos, naudotojų aibę sudaro aukštos kvalifikacijos informacinių technologijų, analitinių sričių specialistai gebantys naudotis sudėtinga programine įranga ir be didesnių nesklandumų naudotis PĮ net ir be grafinės sąsajos, jeigu ji pakankamai aiškiai dokumentuota. Pagrindinė naudotojų patiriama problema, tai kad transformuoti OWL į SBVR galima tik žmogiškųjų resursų pagalba, o tai užima daug laiko ir kaip bet koks žmogaus atliekamas darbas yra neefektyvus ir neatsparus klaidų įsivėlimui.

1.6. Darbo tikslas, uždaviniai, planas ir siekiami privalumai

1.6.1. Darbo tikslas

Aprašyti metodiką skirtą transformuoti iš OWL 2 į SBVR užtikrinant specifikacijų tarpusavio transformacijų korektiškumą ir išsamumą, sukuriant eksperimentinio tyrimo prototipą, kuris leistų transformuoti dalykinės srities ontologijas, aprašytas OWL 2 kalba, į tos srities SBVR veiklos žodyną su taisyklėmis. Šiam tikslui pasiekti iškelti uždaviniai.

1.6.2. Darbo uždaviniai

1. Išanalizuoti:
   1.1. Ontologijų OWL 2 kalbos sąvokas;
   1.2. SBVR veiklos žodynų ir taisyklių sąvokas;
   1.3. OWL 2 ir SBVR elementų tarpusavio atitikimo ir transformavimo taisykles, aprašytas mokslinėse publikacijose;
   1.4. Galimas realizacijos technologijas.
2. Sudaryti ontologijų į veiklos taisykles ir žodynus transformavimo metodiką, suprojektuoti ją realizuojantį prototipinį įrankį;
3. Realizuoti prototipą, leidžiantį atlikti ontologijos transformavimą į veiklos žodyną ir taisykles;
4. Atlikti eksperimentą prototipo tinkamumui įvertinti;
5. Apibendrinti tyrimo rezultatus.

1.6.3. Siekiami privalumai

Tyrimo metu sukurtas transformavimo įrankis suteiks vartotojams galimybę transformuoti ontologiją į veiklos žodyną be didelių laiko išteklių, užtikrinant modelių tarpusavio atitikimą.

1.7. Siekiamo sprendimo apibrėžimas

Problemai spręsti sukurta metodika apibrėžianti taisyklių rinkinį kuriuo reikia vadovautis transformuojant ontologijas į veiklos žodynus ir taisykles. Metodikos taisyklės panaudotos eksperimentiniame įrankyje vykdančiame transformaciją.

1.8. Analizės išvados

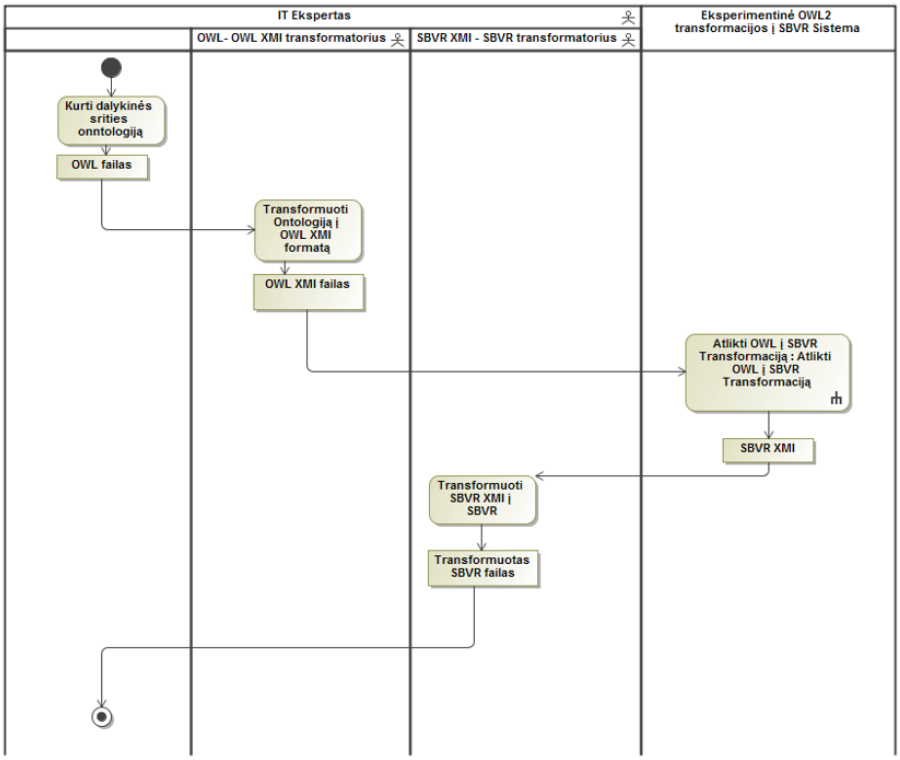
Darbe buvo apžvelgtos problemos, susijusios su dalykinės srities ontologijų, veiklos žodynų ir taisyklių naudojimu, detaliau nagrinėjant jų tarpusavio atitikimą ir transformacijos galimybes. Analizės metu padarytos tokios išvados.

1. Ontologijos yra perspektyvus ir įgyvendinamas būdas patobulinti ir supaprastinti žinių modelių kūrimą ir naudojimą informacinėse sistemose, tačiau verslo pasaulyje žodynai ir veiklos taisyklės yra daug lengviau suprantami nenaudojant papildomų įrankių, todėl reikalinga transformacijos galimybė.
2. Remiantis analize sudarytas algoritmas leidžiantis transformuoti OWL metamodelį į veiklos žodynus ir taisykles.
3. Išnagrinėjus literatūros šaltinius, galima daryti išvadą, kad šiuo metu nėra standartų, kaip turėtų vykti ontologijų transformacija į veiklos žodynus su taisyklėmis, tačiau yra pavyzdžių kurių pagrindu galima sukurti savo ontologijų transformacijos metodiką.
4. Realizacijos technologijų analizė pardodė, kad QVT kalba yra pranašesnė, nei ATL, todėl prototipo realizacijai ir pasirinkta ši QVT kalba.
5. Analizės metu nustatyta, kad šiuo metu nėra egzistuojančių sprendimų galinčių transformuoti ontologijas į veiklos žodynus ir taisyklė. Įvertinus situaciją, nuspręsta, kad toks įrankis yra reikalingas veiklos analitikams ir ekspertams norint aprašyti tą pačią sritį dviem meta modeliais.

2. Ontologijų transformacijos į veiklos žodynus ir taisykles prototipo reikalavimai

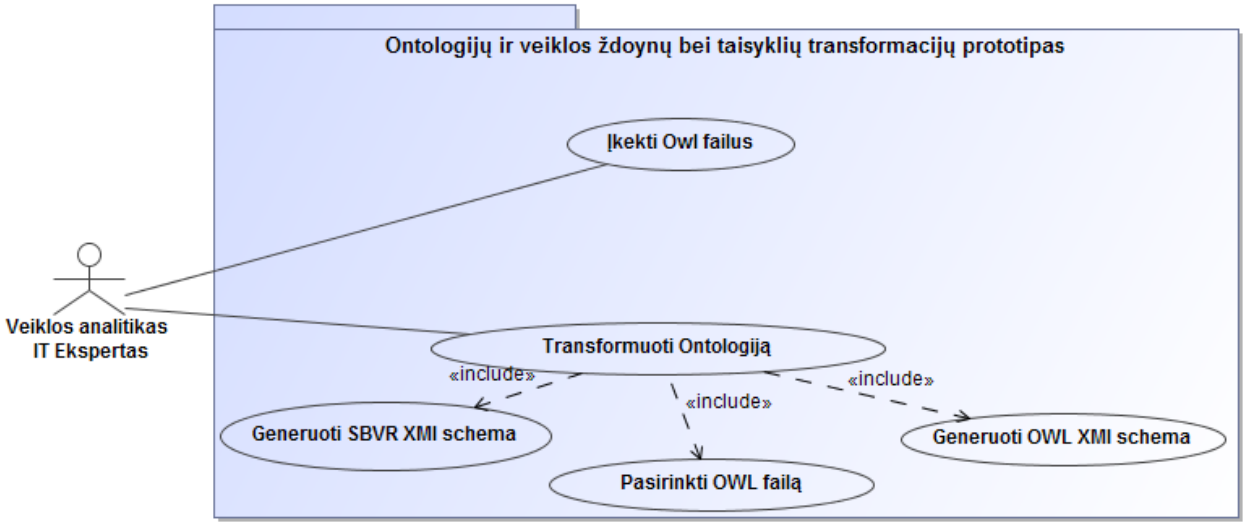
2.1. Transformacijos įrankio panaudojimo atvejai

Prototipo paskirčiai geresniam supratimui pateikta bendros veiklos diagrama 2.1 paveiksle. Sistemos paskirtis transformuoti OWL meta modeliu aprašytas dalykines sritis transformuoti į SBVR modelio aprašus. Šiuo metu atliekant transformacijas išeikvojama per daug žmogiškųjų išteklių ir darbas neefektyvus.



2.1 pav. Bendras veiklos procesas

Detaliau numatomos ontologijų ir veiklos žodynų eksperimentinio tyrimo prototipo vartotojų aibės savybės ir funkcijos apibrėžtos panaudojimo atvejų diagramoje. Sistemos vartotojai gali į sistemą kelti savo Ontologijų failus, kurie vėliau naudojami transformacijoms. Generuojami veiklos žodynų ir taisyklių failai kurie gaunami transformavus parinktą įkeltą failą. Panaudojimo atvejų diagrama pateikta 2.2 paveiksle.



2.2 pav. Kompiuterizuojamų panaudojimo atvejų diagrama

Įkelti OWL failus. Ši funkcija numato kliento OWL failų kėlimą į transformacijo įrankį. Panaudojimo atvejo *„Įkelti OWL failus*“ specifikacija pateikiama 2.1 lentelėje.

2.1 lentelė PA „Įkelti OWL failus“ specifikacija

Panaudojimo atvejis	Įkelti OWL failus
Numeris	PA1
Aktorius	Klientas
Sistema	Ontologijų transformacijos į veiklos žodynus ir taisykles įrankis
Sužadinimo sąlyga:	Klientas pasirenka įrankio funkciją Įkelti OWL failą
Prieš sąlyga	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1.Klientas pasirenką norimą įkelti OWL failą.	1.1. Jei kliento failo dydis neviršija apribojimų ir atitinką reikiamą formatą, tai failas sėkmingai nuskaitomas ir įkeliamas transformacijai.
Po sąlyga	Naujas kliento failas paruoštas transformacijai
Alternatyvos (nesėkmės atvejai)	Išvedamas informacija apie nesėkmingą nuskaitymą
Vykdymo variantai	Klientas pasirenką norimą įkelti OWL failą.
Veiklos taisyklės	1. Failo dydis nedidesnis nei 10MB 2. Failo formatas .OWL.xmi

Transformuoti ontologiją. Ši funkcija numato kliento OWL failų pateiktų xmi formatu transformaciją į SBVR xmi failą. Panaudojimo atvejo *„Įkelti OWL failus*“ specifikacija pateikiama 2.1 lentelėje.

2.2 lentelė PA „Įkelti OWL failus“ specifikacija

Panaudojimo atvejis	Įkelti OWL failus
Numeris	PA1
Aktorius	Klientas
Sistema	Ontologijų transformacijos į veiklos žodynus ir taisykles įrankis
Sužadinimo sąlyga:	Klientas pasirenka įrankio funkciją Transformuoti ontologiją
Prieš sąlyga	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1.Klientas transformuoja pasirinktą failą.	1.1. Įvyksta failo transformacija ir naujas suformuotas žodynas išsaugomas darbinės aplinkos šakninėje direktorijoje
Po sąlyga	Suformuotas transformuotas failas
Alternatyvos (nesėkmės atvejai)	Išvedamas klaidos pranešimas
Vykdymo variantai	Klientas pasirenką norimą transformuoti OWL.xmi failą ir spaudžia paleidžia funkciją.
Veiklos taisyklės	1. Vartotojas privalo pasirinkti tinkamą owl.xmi fialo tipą. Jei vartotojas pasirenka kito tipo failą, sistema nevykdo transformacijos

2.2. Transformacijos įrankio nefunkciniai reikalavimai

Transformacijos įrankio nefunkciniai reikalavimai Transformacijos įrankiui buvo iškelti šie nefunkciniai reikalavimai:

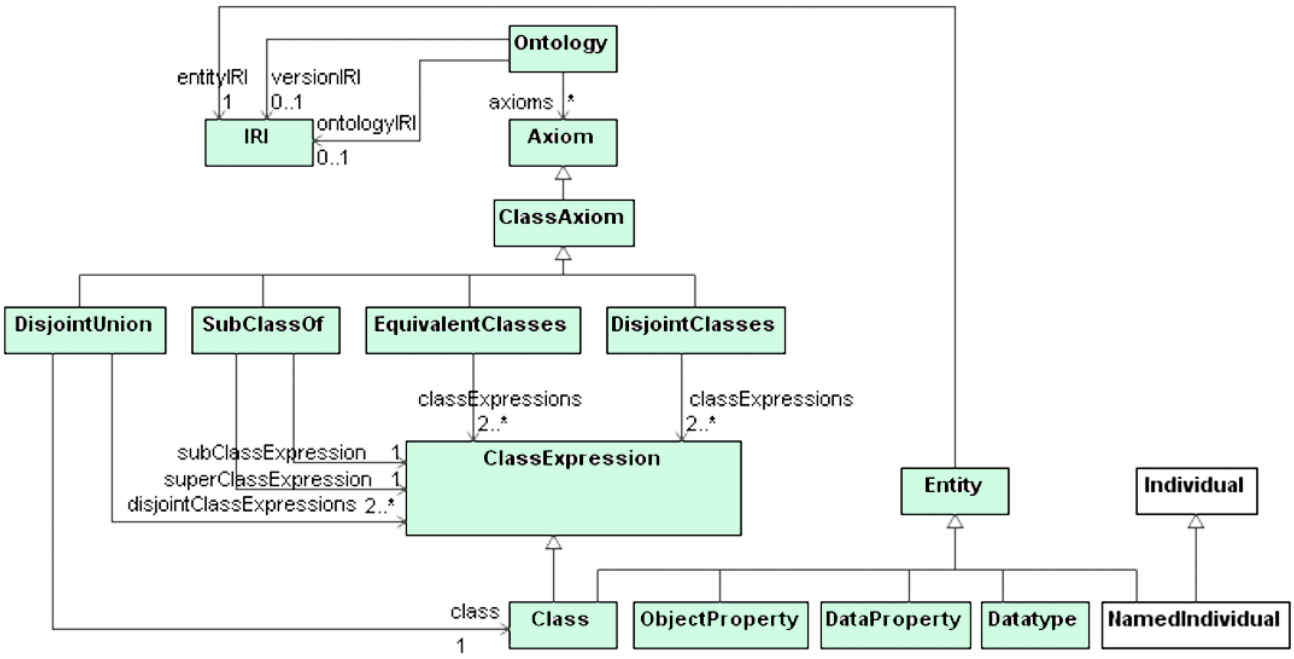
- Sistema turi veikti patikimai, t.y. jos darbas neturi būti nutraukiamas ir duomenys negali būti sugadinami dėl ontologinio aprašo sintaksės klaidų.
- Sistama privalo veikti nepriklausomai nuo naudojamos platformos, t.y. turi būti galimybė ja naudotis įvairiose operacinėse sistemose.

3. Ontologijų transformacijos į veiklos žodynus ir taisykles algoritmo formalus aprašas

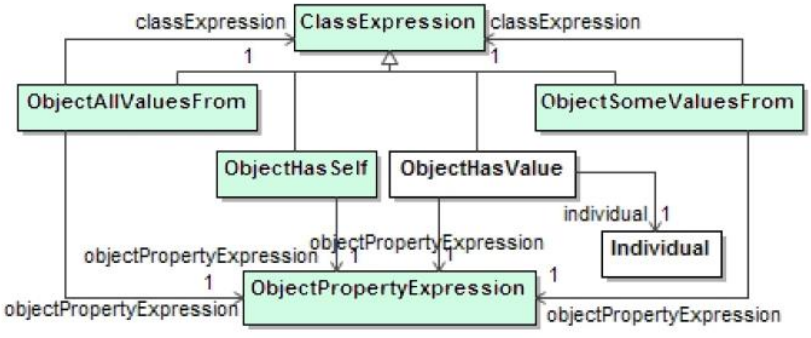
3.1. Transformacijos metodika

Esminė darbo dalis tai visuma taisyklių ontologijų transformacijai į veiklos žodynus ir taisykles. Kuriant metodiką stengtasi apimti kuo daugiau OWL2 ontologijų elementų ir surasti jiems visiems atitinkamas reikšmes veiklos žodynuose ir taisyklėse.

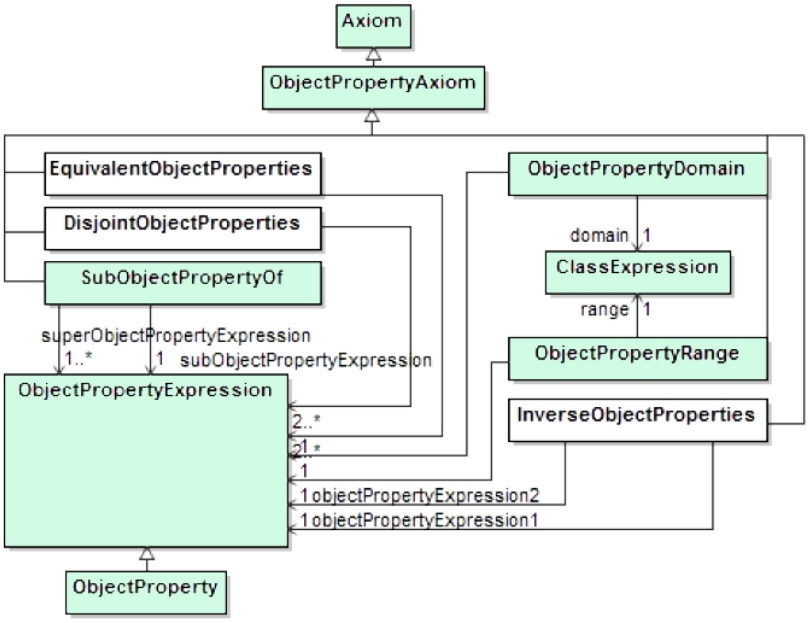
Pagal 3.1 paveikslą matome paryškintus ontologijos schemos diagramos elementų tipus, kurie apimami transformacijos metodikos. Paveiksluose **3.2** ir 3.3 detalizuojamos objektų savybės. Kaip matome iš ontologijos pusės apimama didžioji dalis bazinių elementų, visiškai netransformuojami tik individų elementai.



3.1 pav. Metodikos transformuojami elementai bendri



3.2 **pav. Metodikos transformuojamų objektų savybių detalizacija**



3.3 pav. Metodikos transformuojamų objektų savybių detalizacija 2

Metodikos transformacijos taisyklės aprašytos priklausomybės lygtimis. Kaip pavyzdį paimame poklasių transformacijos aprašą, taisyklę T_2 ir ją išnagrinėsime.

Aprašas: suformuoti SBVR veiksmo konceptą klasių sąryšį, atitinkantį sąryšį tarp klasių.

<precond>: egzistuoja sukurti du bendrieji konceptai, suformuoti pagal taisyklę T_1 ir abu elementai klasės tipo.

T_2: *transform(class$_1$: SubClass, class$_2$: SuperClass, SubClassOf (class$_1$, class$_2$))* → SBVR *konceptų hierachinė asociacija*

Pvz.: *transform('lorry' ,'vehicle', SubClassOf('lorry','vehicle')) →*

lorry

General_concept: vehicle

Visos lygtys susideda iš:

- aprašo – trumpai pateikiamas transformacijos taisyklės veiksmas ir tikslas
- precond – reikalinga sąlyga taisyklės vykdymui, aprašo transformacijos sąlygą arba kokia taisyklė jau turi būti įvykdita prieš vykdant šią.
- transformacijos narių – pateikiami elementai kurie transformuojami iš ontologijos ir jų tipai, pavyzdinėje taisyklėje turime klases atpindinčius elementus (*class$_1$: SubClass, class$_2$: SuperClass),* tačiau šie parametrai nėra apriboti vien klasėmis, jie gali būti bet kokio tipo OWL elementai, pvz.: duomenų savybės, objektų savybės, kardinalumai ir t.t.

- sąryšio tarp elementų – jeigu egzistuoja ryšys tarp elementų, tai jo išraiška pateikiama OWL elementų asociacijos elementu (*SubClassOf (class$_1$, class$_2$)*). Asociaciją išreiškiantiems elementams, kaip parametrai gali būti paduodamos ir kitos asociacijos, jeigu tai reikalinga elementų komplektui transformuoti,
- gaunamas transformuotas SBVR rezultatas – aprašo sakinio tipą kuris gaunamas įvykdžius transformacijos taisyklę(SBVR *konceptų hierachinė asociacija*).

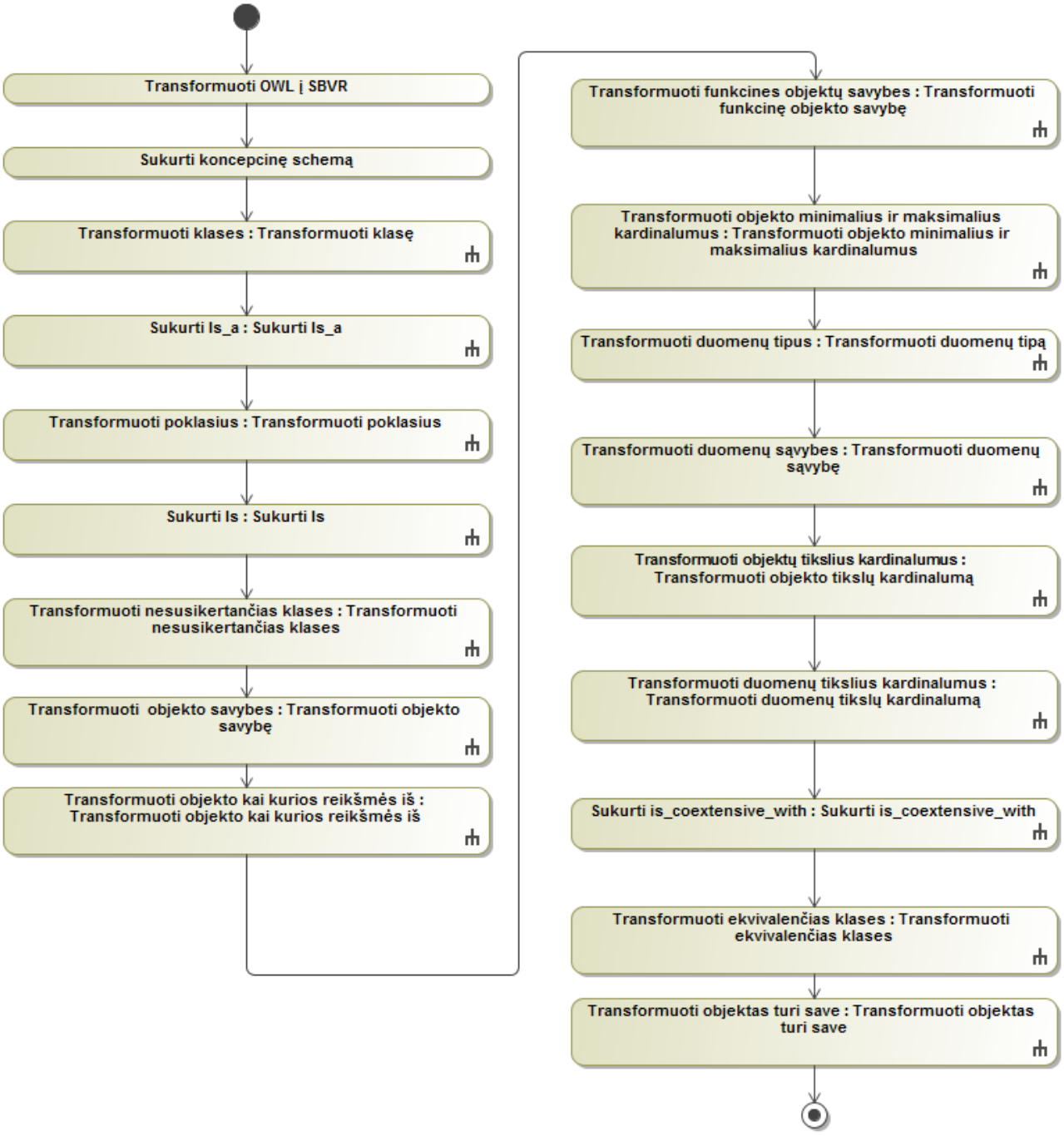
3.2. Transformuojamų ontologijų kokybiniai reikalavimai

Norit, kad sukurta metodika transformuotos ontologijos rezultatas būtų patikimiausias, keliami tam tikri reikalavimai jos aprašui. Visų pirma ontologijos aprašas turėtų būti validus ir be loginių klaidų ir prieštaravimų. Duomenų ir objektų savybės turėtų turėti nustatytas tiek srities, tiek diapazono reikšmes. Rekomenduojama anglų kalba, dėl kuriamų sąryšio elementų, kad būtų teisingai išreikštos pilnosios tekstinės savybių formos žodynuose. Taip pat patartina elementams naudoti etikečių (angl. *label*) anotacijas, nurodant bendrinę elemento tekstinę formą tam, kad būtų išgauta lengviausiai suprantama ir teisingiausia žodyno išraiška, tačiau jų nesant, naudojamos tekstinės išraiškos aprašytos pačiame elemente. Norint transformuoti ontologiją aprašytą lietuvių kalba reikalingas papildomas lemavimo servisas, kuris nėra naudojamas šiame darbte, todėl ontologijų aprašytų lietuvių nerekomenduojama transformuoti, nes veiklos žodynas ir taisyklės būs sunkiai skaitomos ir logiškai nesuprantamos.

3.3. Sprendimo specifikacija

OWL2 transformacijos į SBVR realizacijos algoritmas matomas 3.4 paveiksle. Veikla pradedama nuo Ontologijos transformacijos į žodyną. Pirmiausia transformuojamos ontologijos klasės. Tikrinama ar egzistuoja vaikinės klasės, jei taip, tai sukuriamas „Is_a“ elementas, kuris naudojamas tekstinėse išraiškose atvaizduojant elementų hierarchiją (pvz: sunkvežimis Is_a transporto priemonė.) su transformuotais poklasiais. Transformavus poklasius vėl analogiškai tikrinama ar yra nesusikertančių klasių elementų ir jei jie rasti sukuriama tekstinė išraiška „Is“, kuri naudojama nesusikertančių klasių transformacijos metu, tekstinei išraiškai atvaizduoti. Sekančiuose žingsniuose transformuojami objektų savybių elementai, objekto kai kurios reikšmės iš (angl *ObjectSomeValuesFrom*), funkcinės objektų savybės, po jų seka minimalių, maksimalių ir tiksliųjų kardinalumų apribojimai. Prieš transformuojant duomenų savybių elementus, konvertuojami visi duomenų tipai. Analogiškai kaip ir su vaikinėmis klasėmis prieš konvertuojant ekvivalenčias klases tikrinama ar tokių elementų kiekis didesnis už nulį ir sukuriamas „is_coexistive_with“ elementas, kuris vėliau naudojamas ekvivalenčių klasių tekstinei išraiškai.

Visa detalesnė kiek vieno elemento transformacija pateikiama veiklos diagramose, kurios atvaizduotos paveikslais nuo 3.5 paveikslo iki 3.16.

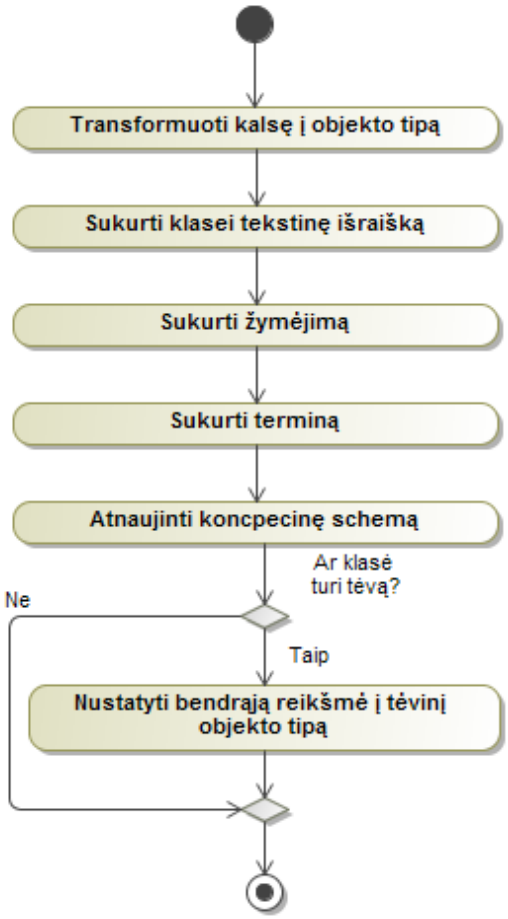


3.4 pav. Bendrinė transformacijos veiklos diagrama

3.5 paveiksle pavaizduota klasių transformacijos veiklos diagrama. Kiek vienai klasei sukuriamas objekto tipas ir tekstinė išraišką, tada kuriami žymėjimo ir termino elementai, kurie sujungia objekto tipą su tekstine išraiška. Sekančiame žingsnyje tikrinama ar klasė turi tėvinį elementą, jeigu tėvinis elementas rastas, tai objekto tipui papildomai uždedamas požymis rodantis į tėvinę klasę.

Aprašas: suformuoti bendrąjį konceptą, atitinkantį klasę.

T_1*: transform(class:Class)* → SBVR *bendrasis konceptas*
Pvz.: *transform('vehicle')* → vehicle



3.5 pav. Klasių transformavimo veiklos diagrama

3.6 paveiksle matoma poklasių transformavimo veiklos diagrama. Kadangi ne visais atvejai kuriami vienodi elementai, tai prieš pradedant transformaciją, tikrinama ar poklasio tėvinis elementas yra klasės tipo.

Kai tėvinis elementas klasės tipo(šaka „Taip“), tai sukuriamas kategorizacijos fakto tipas (angl *categorizationFactType*), kuris nurodo sąryšį tarp elementų, abi klases paverčia į roles. Po

to kai atnaujinama koncepcinė schema, sukuriami žymėjimo (angl *designation*) ir termino (angl *term*) elementai rišantys tekstines išraiškas su rolėmis, tada sukuriamas elementas su pilnu tekstu kuris atspindi tėvinės klasės ryšį su vaikine klase ir sukuriame vietos išskyrimo (angl *PlaceHolder*) elementus, kurie nurodo kuris teksto dali atspindi kurią rolę.

Kai tėvinis elementas objekto visos reikšmės iš (angl *ObjectAllValueFrom*) tipo (šaka „Ne“), kuriamas asociatyvus fakto tipas (angl *AssociativeFactType*), nurodantis kitokį sąryšį, tarp elementų. Kuriamos rolės abiem transformacijos elementams. Kadangi objekto visos reikšmės iš (angl *ObjectAllValueFrom*) atspindi apribojimo veiksmą, kuriami atominės formuluotės (angl *AtomicFormulation*) elementai, su savo kintamaisiais (angl *Variable*), rolių dalyvavimą apribojime apibrėžiame rolių susiejimo (angl *RoleBinding*) elementais, tada kuriamas egzistencinis kiekybiškumas (angl *ExistensialQantification*) ir būtinumo formuluotė (angl *NecessityFormulation*) apibrėžianti rolės egzistavimui reikalingas sąlygas. Po to kuriami atitinkami fakto simbolio (angl *FactSymbol*), sakinio formuluotės (angl *SententialForm*), teiginio (angl *Statement*) ir pasiūlymo (angl *Proposition*) elementai.

Aprašas: suformuoti SBVR veiksmo konceptą klasių sąryšį, atitinkantį sąryšį tarp klasių.

<precond>: egzistuoja sukurti du bendrieji konceptai, suformuoti pagal taisyklę T_1 ir abu elementai klasės tipo.

T_2: *transform*($class_1$: *SubClass*, $class_2$: *SuperClass*, *SubClassOf* ($class_1$, $class_2$)) → SBVR *konceptų hierachinė asociacija*

Pvz.: *transform(‘lorry’ ,‘vehicle’, SubClassOf(‘lorry’, ‘vehicle’))* →

lorry

General_concept: vehicle

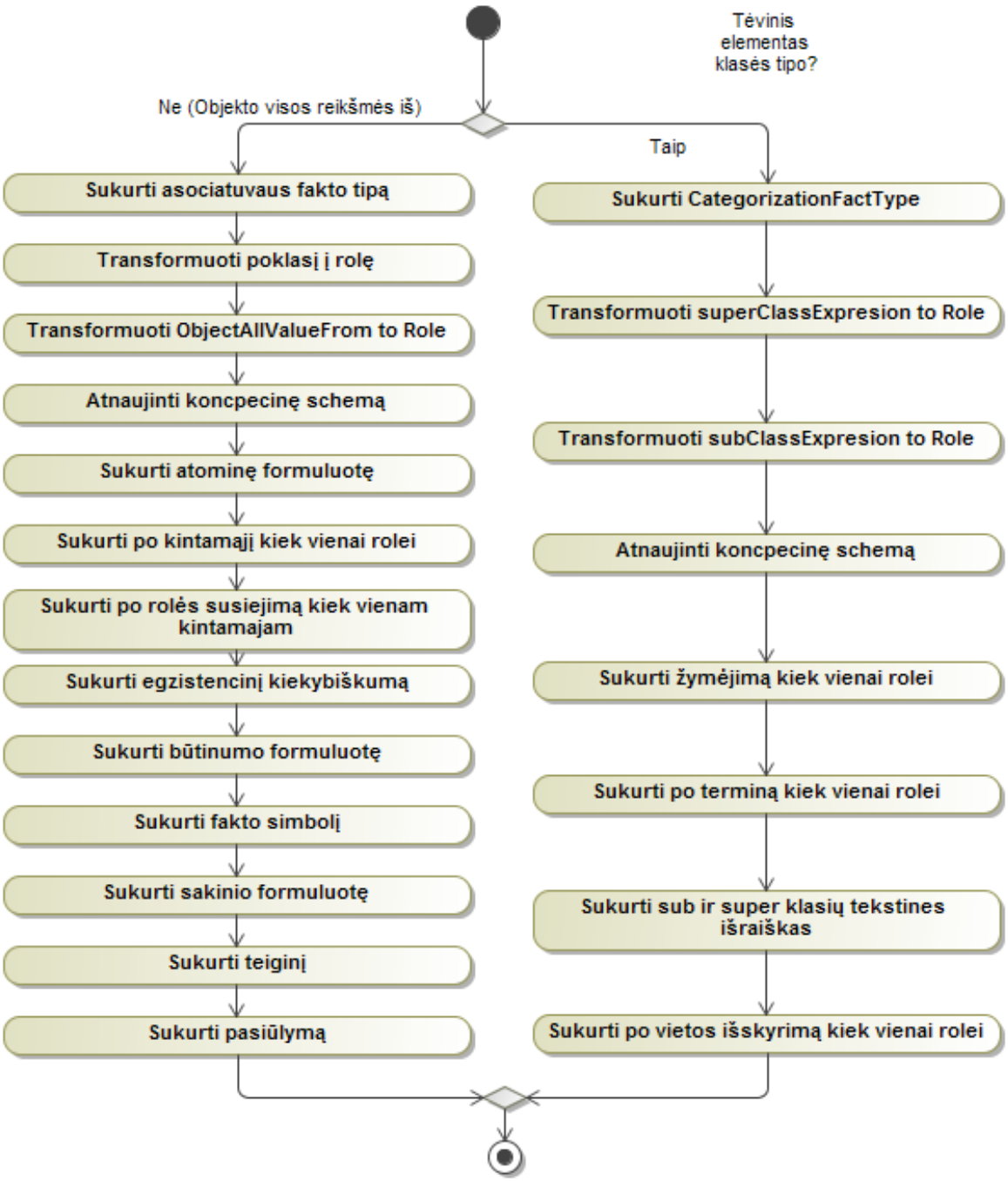
Tikslas: suformuoti rinkinį veiklos taisyklių, atitinkančių asociacijos ryšį.

<precond>: SuperClass elementas *ObjectAllValuesFrom* tipo.

T_3: *transform*($class_1$: *Class*, $class_2$: *Class*, $class_3$: *Class*, *ObjectAllValuesFrom*($class_1$, *quantifier*$_1$, $class_2$, $class_3$)) → SBVR *veiklos taisyklė aprašanti koncepto sandarą*

Pvz.: *transform(’lorry’,‘ engine’, ‘diesel_Engine’ , ObjectAllValuesFrom(’Lorry’,1, ‘ Engine’, ‘diesel_Engine’))* →

It is necessary that lorry *contains* engine *that is* diesel_engine



3.6 pav. Poklasių transformavimo veiklos diagrama

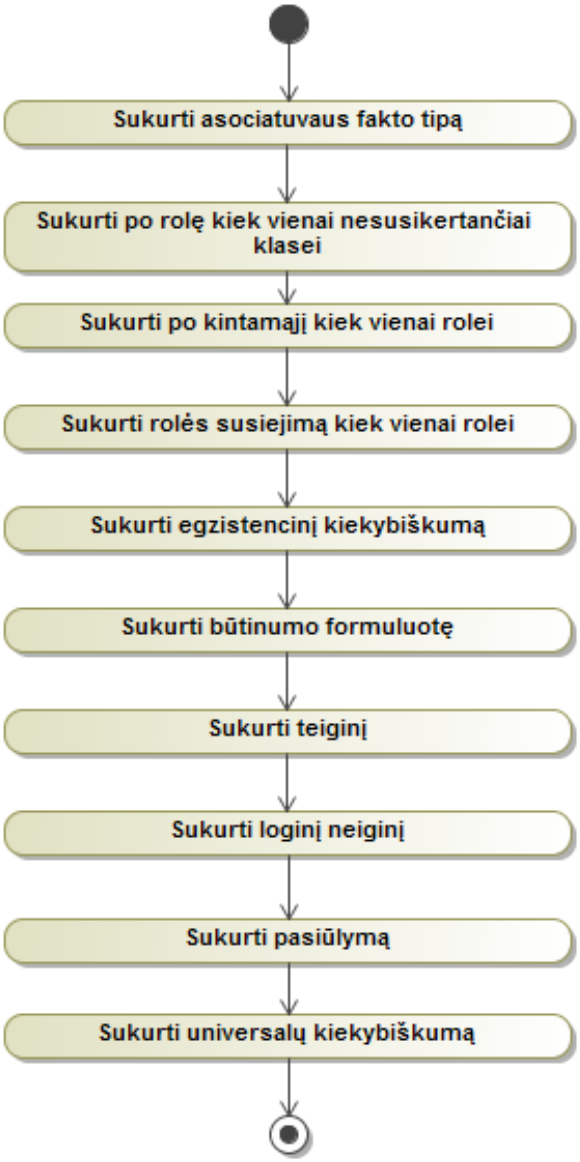
3.7 paveiksle pavaizduotas nesusikertančių klasių (angl *DisjointClass*) transformavimo algoritmas. Sąryšio išreiškimui tarp klasių kuriamas asociatyvaus fakto tipo (angl *AssociativeFactType*) elementas, abi klasės tada keičiamos į rolės elementus, abejoms rolėms kuriama po kintamąjį (angl *Variable*) ir rolės susiejimo (angl *RoleBinding*) Elementą. Antrai rolei kuriamas egzistencinio kiekybiškumo (angl *ExistentialQuantification*) elementas, o pirmai universalaus kiekybiškumo (angl *UniversalQuantification*), po to kuriami loginis neiginys (angl

LogicalNegation), pasiūlymas (angl *Proposition*), būtinumo formuluotė (angl *NecessityFormulation*) ir teiginys (angl *Statement*).

Tikslas: suformuoti rinkinį veiklos taisyklių, atitinkančių nesusikertančių klasių ryšį.

<precond>: egzistuoja sukurti du bendrieji konceptai, suformuoti pagal taisyklę T_1 ir abu elementai klasės tipo.

T_4: *transform(class$_1$: Class, class$_2$: Class , Disjoint(class$_1$, class$_2$))* → SBVR *veiklos taisyklė aprašanti neįmanomumo salyga*
Pvz.: *transform('lorry','motorcycle' , Disjoint('lorry','motorcycle'))* →
It is impossible that lorry *is* motorcycle



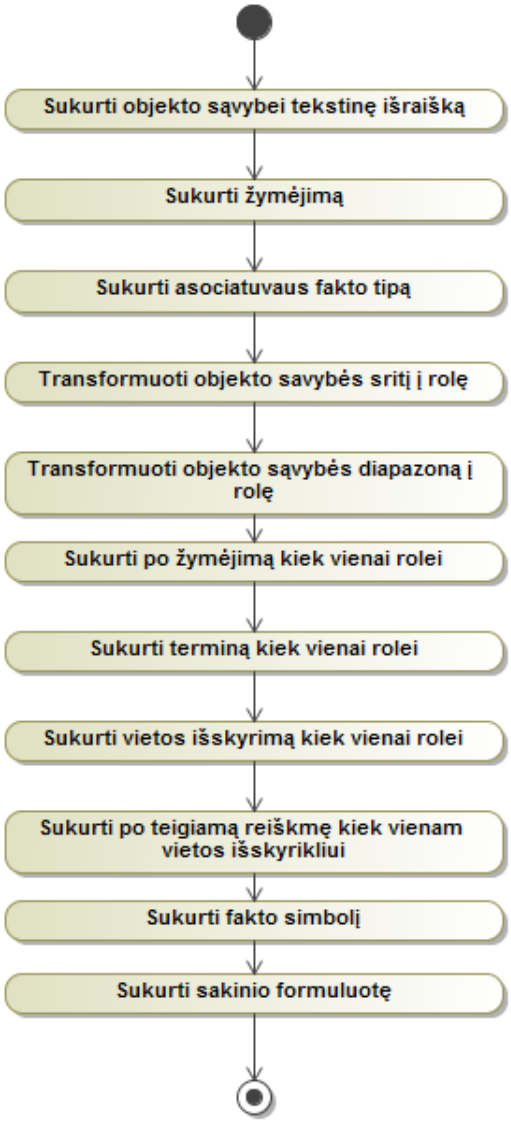
3.7 pav. Nesusikertančių klasių transformavimo veiklos diagrama

3.8 paveiksle pavaizduota objekto sąvybės (angl *ObjectProperty*) transformacijos veiklos diagrama.Veiksmai pradedami sukuriant objekto savybės tekstinę išraišką ir tikslą. . Sąryšio išreiškimui tarp klasių kuriamas asociatyvaus fakto tipo (angl *AssociativeFactType*) elementas. Objekto savybės sritis ir diapazonas (angl *ObjectProperty Domain, Range)* elementai kečiami į rolės elementus. Rolėms sukuriama po tikslą, tekstinę išraišką, sąkinio formuluotę (angl *SententialForm*) su vietos išskyrimo (angl *placeHolder*) elementais ir jų teigiamomis sveikosiomis (angl *positiveInteger*) reikšmėmis.

Tikslas: suformuoti veiksmažodinį konceptą, atitinkantį priklausomybės ryšį tarp klasių.

<precond>: egzistuoja sukurti du bendrieji konceptai, suformuoti pagal taisyklę T_1.

T_5: *transform(class*$_1$*: Class, class*$_2$*: Class, objProperty: ObjectProperty, Association(objProperty,class*$_1$*, class*$_2$*))* → SBVR *veiksmažodinio koncepto asociacija*
Pvz.: *transform('lorry', 'professional_driver', 'is_driven_by', Association('is_driven_by', 'lorry', 'professional_driver'))* → lorry *is_driven_by* professional_driver

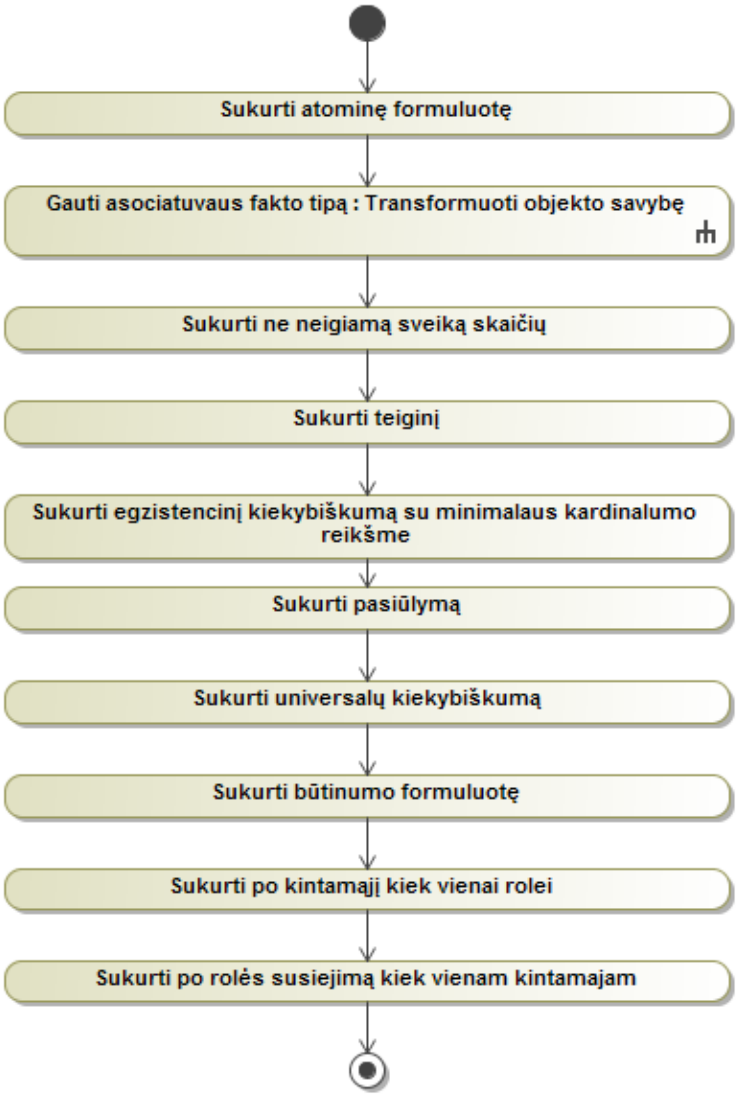


3.8pav. Objekto savybių transformavimo veiklos diagrama

3.9 paveiksle pavaizduota objekto kai kurios reikšmės iš (angl *ObjectSomeValueFrom)* transformavimo veiklos diagrama. Veikla pradedama nuo atominės formuluotės (angl *AtomicFormulation*) elemento sukūrimo, kuris rodo į taikomus apribojimus, kadangi objekto savybės (angl *ObjectProperty*) jau transformuotos, tai paimama transformacijos reikšmė, kuri naudojama tolesnių elementų sąryšiuose. Sukuriame ne neigiamą sveiką skaičių (angl *NonNegativeInteger*), kuris nurodo, skaitinę kardinalumo reikšmė. Atitinkamai sukuriame universalaus kiekybiškumo ir teiginio (angl *Statement*, *UniversalQuantification*), kintamuosius ir rolės susiejimą (angl *RoleBinding*). Kardinalumas aprašomas egzistencinio keikybiškumo (angl *ExistentialQauntification*) elemente su atributu minimalus kardinalumas (angl *minimumCardinality*), kuris rodo į aukščiau sukurtą ne neigiamą sveiką skaičių.

Tikslas: suformuoti rinkinį veiklos taisyklių, atitinkančių asociacijos ryšį su egzistenciniu reikalavimu.

T_6: *transform(class$_1$: Class, class$_2$: Clas, ObjectSomeValuesFrom(class$_1$, class$_2$))* → SBVR *veiklos taisyklė aprašanti koncepto sandarą*
Pvz.: *transform('vehicle', 'engine', ObjectSomeValuesFrom('vehicle', 'engine'))*→ It is necessary that vehicle *contains* engine

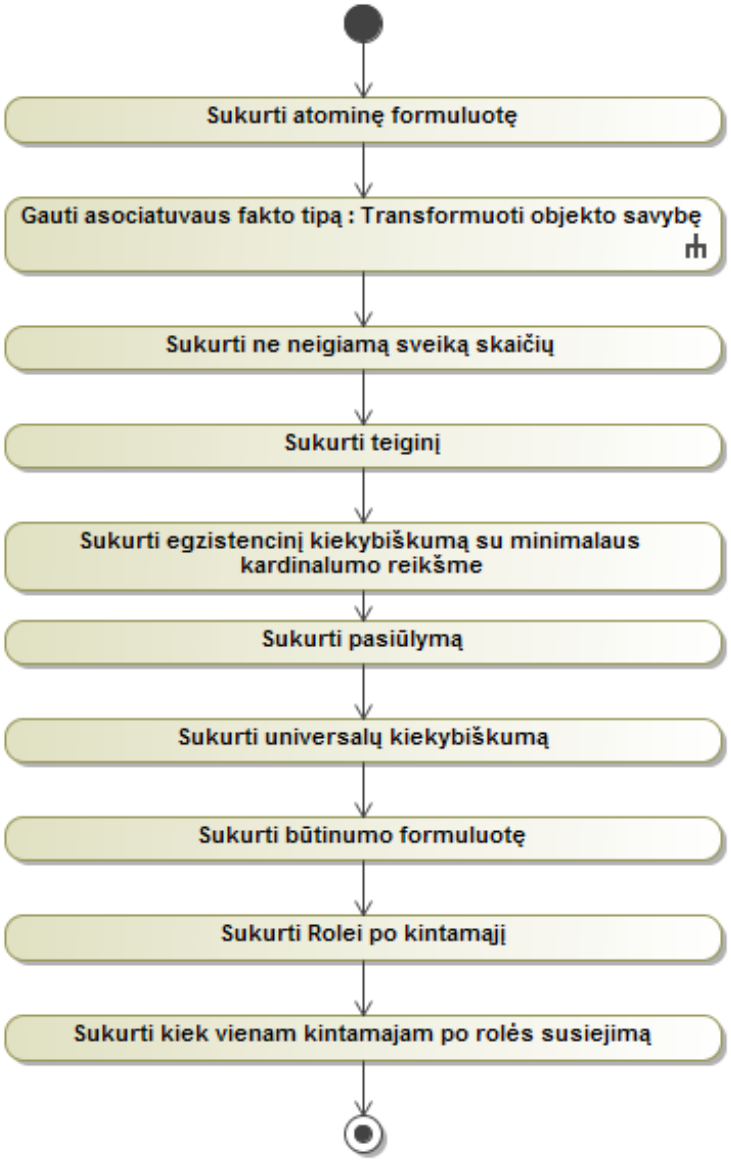


3.9 pav. Objekto kai kurios reikšmės iš transformavimo veiklos diagrama

3.10 paveiksle, kuriame atvaizduota funkcinės objekto savybės (angl *FunctionalObjectProperty*) transformavimo veiklos diagrama, veiksmų seka atliekama tokia pat, kaip ir paveiksle 3.9, tik vietoje egzistencinio kiekybiškumo kuriamas maksimalaus kiekybiškumo (angl *AtMosNQuantification*) elementas su maksimalaus kardinalumo (angl *maximumCardinality*) atributu.

Tikslas: suformuoti rinkinį veiklos taisyklių, atitinkančių asociacijos ryšį egzistenciniu reikalavimu su maksimaliu kardinalumu.

T_7: transform($class_1$: Class, $class_2$: Class, objProperty: ObjectProperty, FunctionalObjectProperty(objProperty, $class_1$,, $class_2$)) → SBVR veiklos taisyklė aprašanti veiksmažodinio koncepto asociaciją su kardinalumo nurodymu
Pvz.: *transform('lorry', 'professional_driver', 'is_driven_by', FunctionalObjectProperty('is_driven_by', 'lorry', 'professional_driver'))*→ It is necessary that lorry *is_driven_by at_most 1* professional_driver



3.10 pav. Funkcinių objekto savybių transformavimo veiklos diagrama

3.11 paveiksle atvaizduotas minimalaus ir maksimalaus kardinalumų transformavimo algoritmo veiklos diagrama. Kadangi minimalus ir maksimalus kardinalumai yra objekto savybės (angl *objectProperty*) dalis, kuris jau yra transformuotas, tai paimame, gautą rezultatą ir kiek vienai rolei sukuriame kintamąjį (angl *Variable*) ir rolės susiejimo (angl *RoleBindig*) elementus.

Sekančiame žingsnyje kuriame universalaus kiekybiškumo, pasiūlymo ir teiginio (angl *UniversalQuiantification, Proposition Statement)* elementus. Pereiname prie sąlygos elemento, kuris tikrina ar klasė turi maksimalų ir minimalų apribojimą:

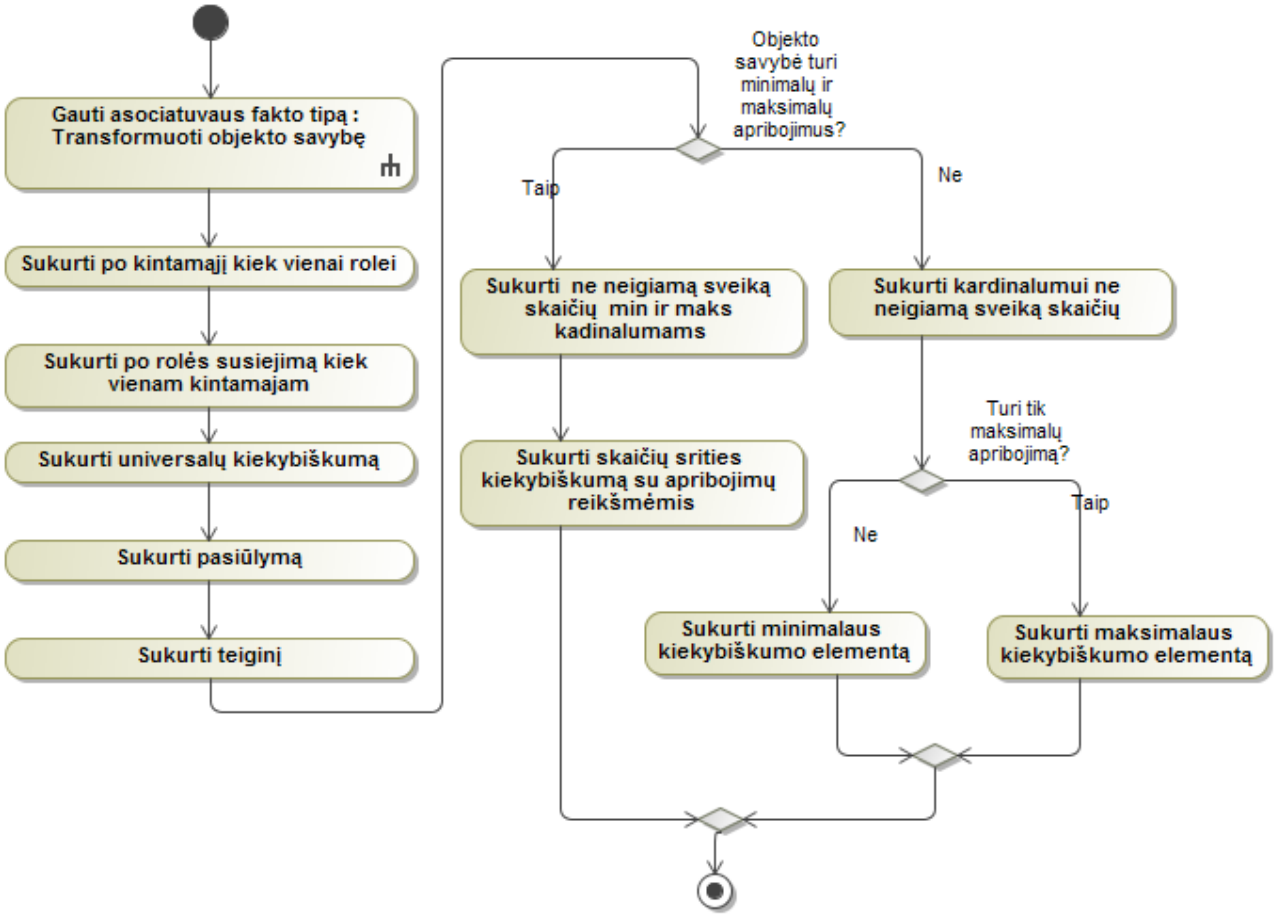
Kai klasė apribota minimaliu ir maksimaliu kardinalumu tokiu atveju patenkame į šaką „Yes“, kurioje kuriami du ne neigiamo sveikojo skaičiaus (angl *NonNegativeInteger*) elementai su atitinkamomis minimalia ir maksimalia reikšmėmis ir po to kuriamas skaičių srities kiekybiškumas (angl *NumericRangeQuantification*), kuris ir nurodo klasės kiekinius apribojimus.

Kai klasė apribota tik vienu kardinalumu patenkame į „No“ šaką, kurioje sukuriamas vienas neigiamo sveikojo skaičiaus (angl *NonNegativeInteger*) elementas su kardinalumo reikšme. Tada tikriname ar maksimalus kardinalumo apribojimas. Jeigu maksimalus - kuriame maksimalaus kiekybiškumo (angl *AtMostNQuantification*) elementą. Jeigu minimalus - kuriame minimalaus kiekybiškumo (angl *AtLeastNQuantification*) elementą

Tikslas: suformuoti rinkinį veiklos taisyklių, atitinkančių minimalius ir maksimalius kardinalumus

.

T8: transform(class1: Class, class2: Class, minQuantifier1:MinCardinality, maxQuantifier1:MaxCardinality CardinalityMinMax(class1, minQuantifier1, class2, maxQuantifier1)) → SBVR *veiklos taisyklė aprašanti elementų sandarą su minimaliais ir maksimaliais kardinalumais*
Pvz.: *transform('Lorry', 'Side_Mirror',4,null, CardinalityMinMax ('Lorry',4, 'Side_Mirror',null)) →*
It is necessary that Lorry *has* at_least 4 Side_Mirror
Pvz.: *transform('Car', 'Wheel',null,4, CardinalityMinMax ('Car',null, 'Wheel',4)) →*
It is necessary that Car *contains* at_most 4 Wheel
Pvz.: *transform('high_utility_vehicle', 'axle_lock', 2, 9, CardinalityMinMax('high_utility_vehicle',2, 'axle_lock', 9)) →*
It is necessary that high_utility_vehicle *has* at_least 2 and at_most 9 axle_lock

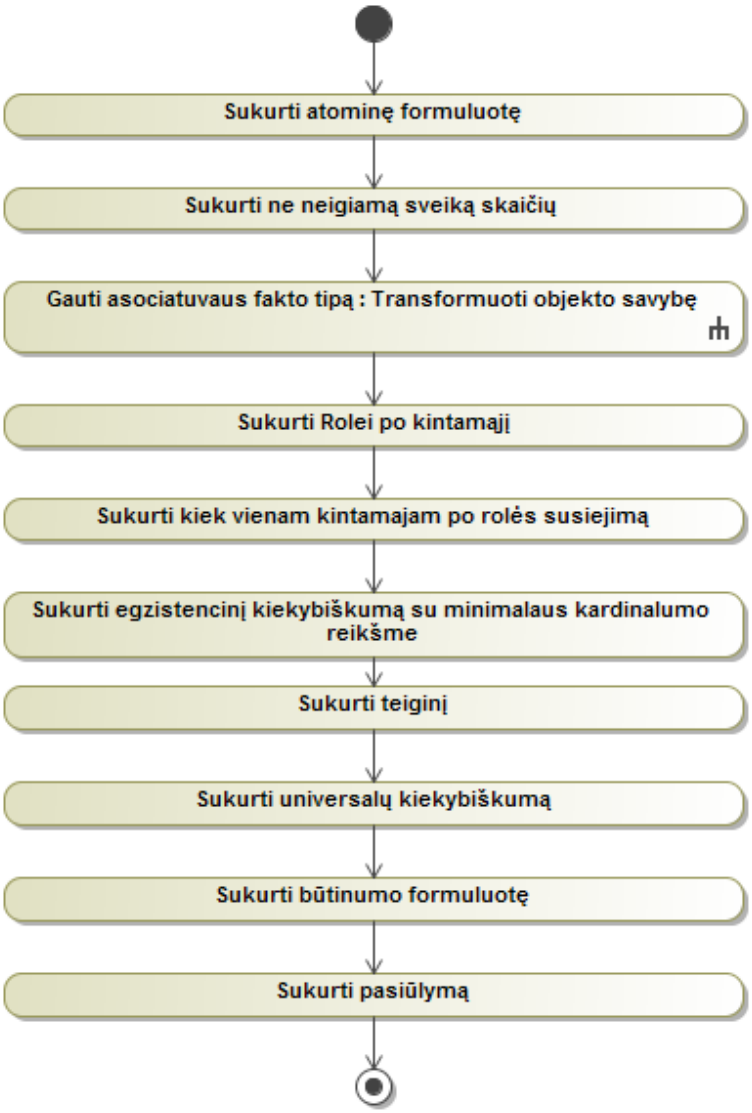


3.11 pav. minimalaus ir maksimalaus kardinalumų transformavimo veiklos diagrama

3.12 paveiksle, kuriame atvaizduota objekto tikslaus kardinalumo (angl *ObjectExactCardinality*) transformavimo veiklos diagrama, veiksmų seka atliekama tokia pat, kaip ir paveiksle 3.9, tik vietoje egzistencinio kiekybiškumo (angl *ExistentialQuantification*) kuriamas egzistencinio kiekybiškumo elementas su minimalaus kardinalumo (angl *minimumCardinality*) atributu.

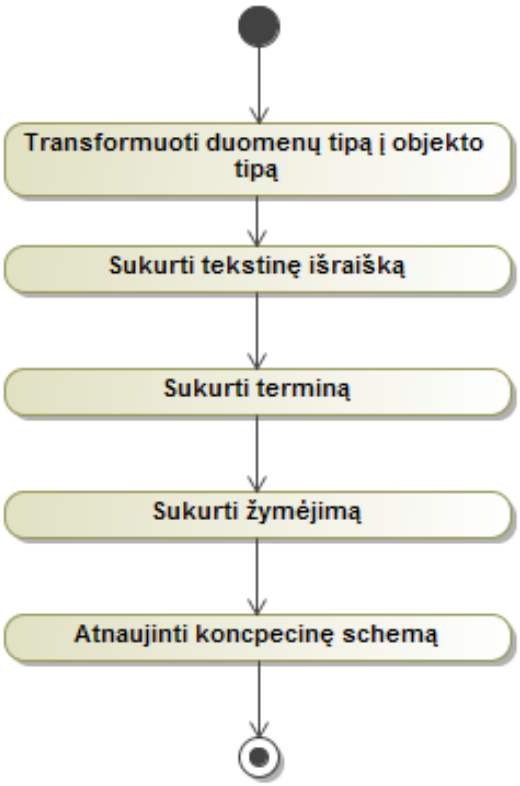
Tikslas: suformuoti rinkinį veiklos taisyklių, atitinkančių tikslius kardinalumus .

T_9: *transform(class*$_1$*: Class, class*$_2$*: Class, quantifier*$_1$*:Cardinality, ExactCardinality(class*$_1$*, class*$_2$*, quantifier*$_1$*))* → SBVR *veiklos taisyklė aprašanti elementų sandarą tiksliu kardinalumu*
Pvz.: *transform('lorry','tachograph', 1, ExactCardinality('lorry', 'tachograph',1))* →
It is necessary that lorry *contains* exactly 1 tachograph



3.12pav. ObjectExactCardinality transformavimo veiklos diagrama

3.13 paveiksle pavaizduota duomenų tipo (angl *DataType*) transformavimo veiklos diagrama. Duomenų tipai naudojami duomenų sąvybių (angl *DataProperty*) transformacijose, visi duomenų tipai (*integer, boolean, double* ir t.t.) esantys ontologijoje yra transformuojami į objekto tipo (angl *objectType*) elementus su terminais ir tikslo elementais, bei tekstinėmis išraiškomis.



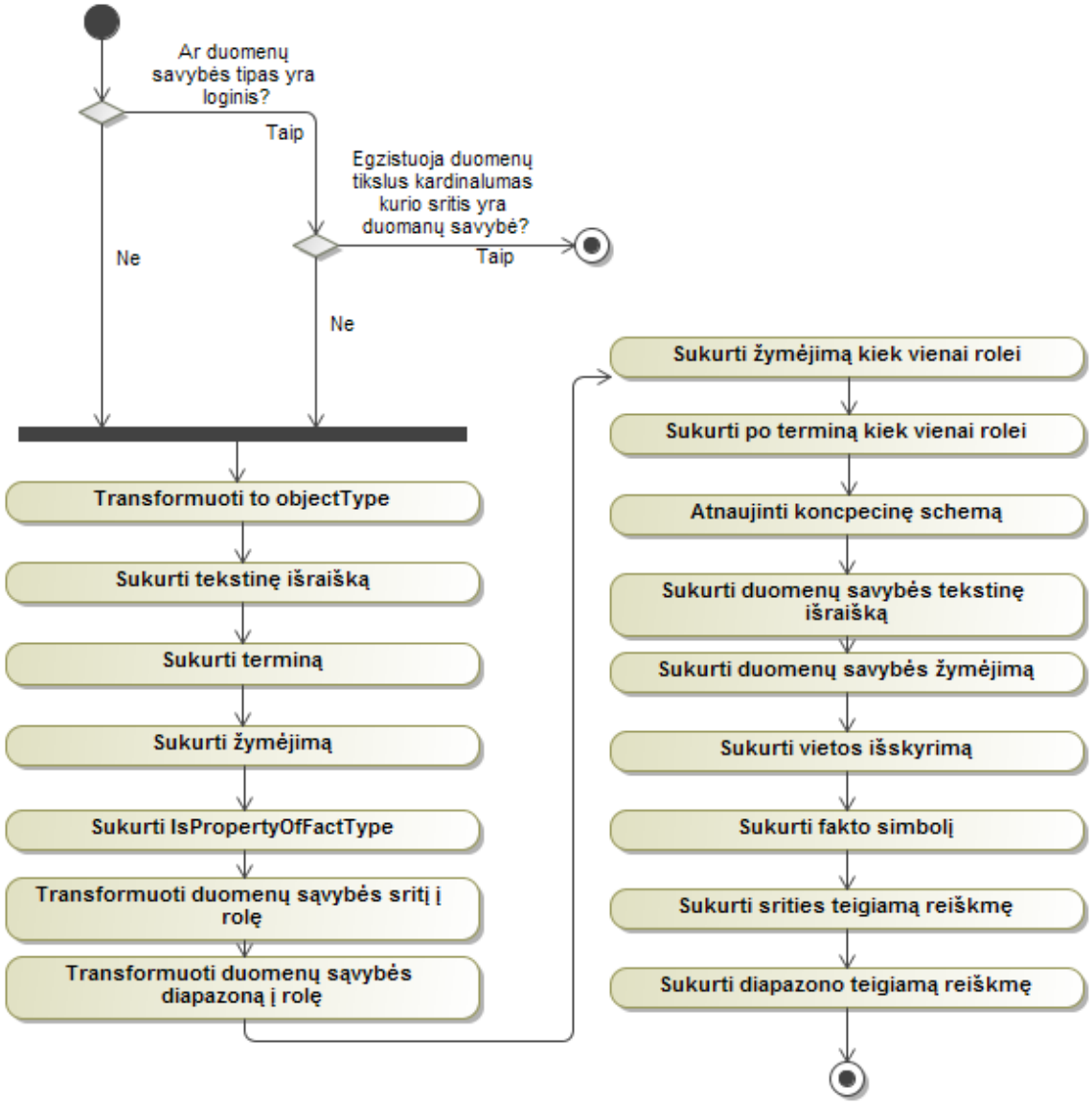
3.13 pav. DataType transformavimo veiklos diagrama

3.14 paveiksle pavaizduota duomenų sąvyvės (angl *DataProperty*) transformavimo veiklos diagrama. Prieš pradedant transformaciją, tikrinama ar duomenų savybė yra loginio tipo savybė, jeigu taip, tai tikrinama ar yra toks tikslaus duomenų kardinalumo (angl *DataExactCardinality*) elementas, kurio range yra transformuojamas duomenų savybės elementas, jeigu yra, tai transformacija nutraukiama ir elementas transformuojamas vėliau. Jeigu viena iš šių sąlygų grąžina neigiamą rezultatą, tada transformacija vykdoma ir duomenų savybė transformuojama į objekto tipą (angl *objectType*), kuriam sukuriamas terminas, žymėjimas ir tekstinė išraiška. Toliau kuriamas yra savybė fakto tipo (angl *IsPropertyFactType*) elementas nurodantis sąryšį su klase. Duomenų savybės sritis ir diapazonas (angl *DataPropertyDomain, DataPropertyRange*) konvertuojami į roles, kurioms sukuriamos po termino ir tikslo elementą. Atnaujinus koncepcinę schemą, kuriama pilna duomenų sąvybės tekstinė išraiška ir fakto simbolis (angl *factSymbol*). Sukuriamos teigiamo sveikojo skaičiaus (angl *positiveInteger*) elementai

nurodantys rolių vietą tekstinėje išraiškoje ir įrašomos vietos išskyriklio (angl *PlaceHolder*) elemento atributuose.

Tikslas: suformuoti charakteristiką, atitinkančią duomenų savybę

T_{10}: transform(class:Class, dataProperty$_1$: DataProperty) → SBVR *konceprto charakteristika*
Pvz.: *transform('engine', 'is_forced_induction')* → engine *is_forced_induction*



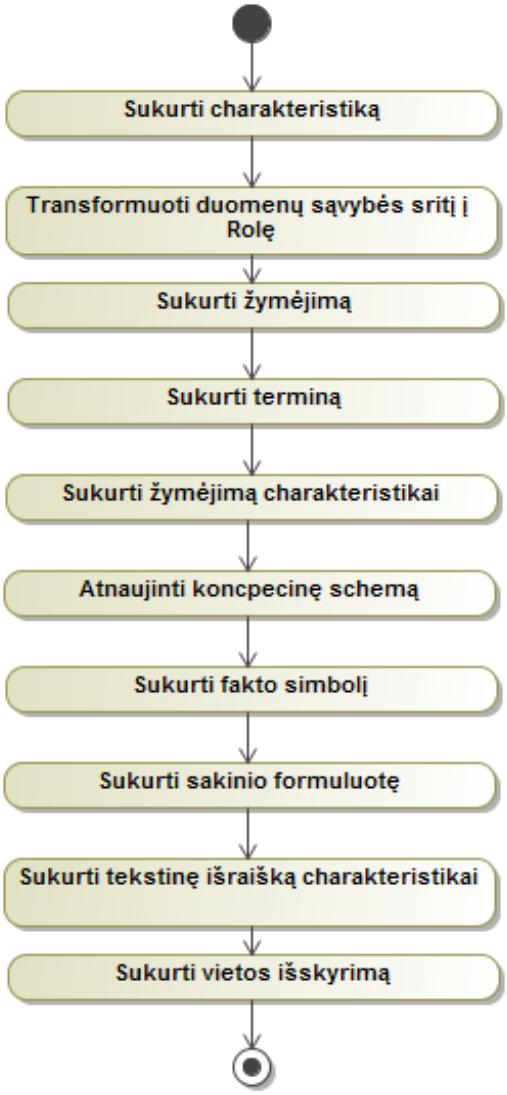
3.14 pav. Duomenų savybių transformavimo veiklos diagrama

3.15 paveiksle pavaizduota tikslaus duomenų kardinalumo (angl *DataExactCardinality*) transformavimo veiklos diagrama. Transformuojant duomenų savybę (angl *DataProperty*), kai savybė yra loginio tipo, kuriamas charakteristikos (angl. *Charasteristic*) elelemtas, turintis tik vieną rolę transformuotą iš duomenų savybės. Rolei ir klasei sukuriami tikslo ir termino elementai. Atnaujinus koncepcinę sukuriama tekstinė išraiška su atitinkama sakinio formuluote (angl

SententialForm) vietos išskyriklio (angl *PlaceHolder*) ir fakto simbolio (angl *FactSymbol*) elementais.

Tikslas: suformuoti apimančią charakteristiką, atitinkančią duomenų savybę

T_{11}: transform(class:Class, dataProperty$_1$: DataProperty, Incorporates(class$_1$,dataProperty$_1$)→ SBVR *įtrauktoji charakteristika*
Pvz.: *transform('engine', 'is_ForcedInduction',Incorporates('engine', 'is_ForcedInduction'))* → concept `engine ` incorporates characteristic `engine *is_ForceInduction*`



3.15 pav. Duomenų tikslaus kardinalumo transformavimo veiklos diagrama

3.16 paveiksle pavaizduota Ekvivalenčių klasių transformavimo veiklos diagrama. Transformuojant ekvivalenčias klases galimi du variantai, kai yra tik ekvivalenčios klasės arba kai ekvivalenčios klasės yra narės nesusikertančių klasių sąjungoje (angl *DisjointUnion*).

Transformuojant tik ekvivalenčias klases sukuriamas asociacijos fakto tipo (angl *AssociativeFactType*) elementas, rolės kuriamos iš atitinkamų klasių. Sukuriamas žymėjimas (angl

Designation) su nuoroda į „*is_coexistive_with*“ tekstą. Ekvivalenčios klasės išreiškiamos pilna tekstine išraiška ir sukuriama sakinio formuluotė (angl *SententialForm*) su vietos išskyrimo (angl *placeHolder*) elementais ir jų teigiamomis sveikosiomis (angl *positiveInteger*) reikšmėmis nurodančiomis rolių pozicijas tekstinėje išraiškoje.

Kai ekvivalenčios klasės yra narės nesusikertančių klasių sąjungoje, tada kuriamas segmentacijos elementas. Klasės transformuojamos į segmentacijos kategorijų tipus (angl *Segmentation Category*), šios klasės priskiriamos segmentacijai. Segmentacijai sukuriamas individualus konceptas (angl *IndividualConcept*). Klasėms kuriama po tekstinę išraišką, terminą ir tikslą. Segmentacijos ir individualaus koncepto elementams sukuriamas vardo elementas. Atlikus visus veiksmus atnaujinama koncepcinė schema.

Tikslas: suformuoti sinoniminius konceptus, atitinkančius ekvivalenčias klases.

<*precond*>*:* egzistuoja sukurti du bendrieji konceptai, suformuoti pagal taisyklę T_1.

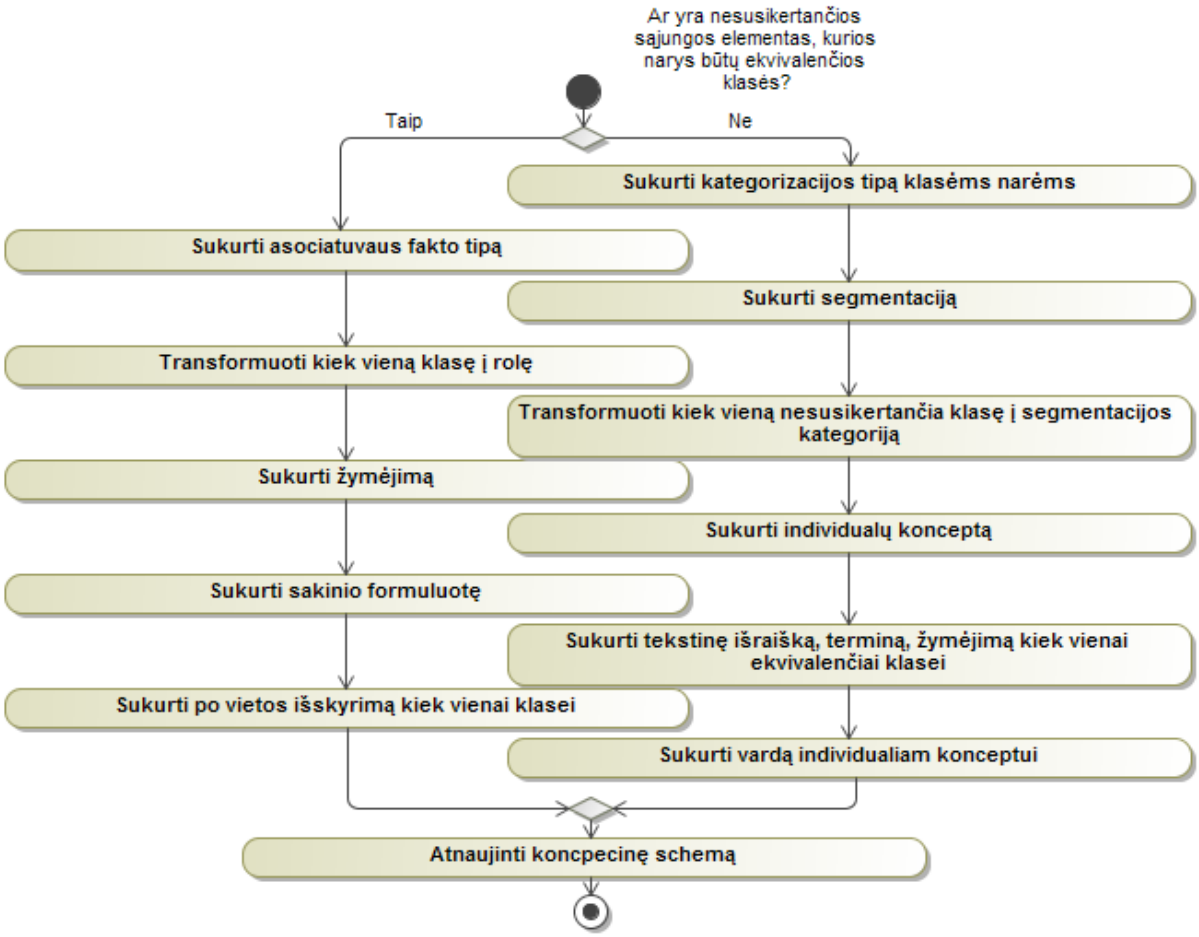
T_{12}*: transform(class*$_1$*: Class, class*$_2$*: Class, Equivalent (class*$_{1,}$ *class*$_2$*))* → SBVR *koegzistuojančių konceptų asociacija*
Pvz.: *transform(‘fuel’, ‘gas’, Equivalent (‘fuel’, ‘gas’))* →
fuel *is_coextensive_with* gas

Tikslas: suformuoti konceptus, atitinkančius klasių segmentacija.

T_{13}*: transform(SegmentClass*$_1$*: Class,class*$_1$*: Class, class*$_2$*: Class,Equivalent(SegmentClass*$_1$ *DisjointUnion (class*$_{1,}$ *class*$_2$*))*→ SBVR *konceptų segmentavimo asociacija*
Pvz.: *transform(‘vehicle_drive_train’, ‘front_wheel_drive’,‘ front_wheel_drive‘ Equivalent(‘vehicle_drive_train’ DisjointUnion (‘front_wheel_drive’,‘ front_wheel_drive‘))*

```
Vehicle_by_drive_train
 Necessity: segmentation for general_concept vehicle that subdivides vehicle
by vehicle_drive_train

front_wheel_drive
 General_concept: vehicle
 Necessity: is_included_in Vehicle_by_drive_train
rear_wheel_drive
 General_concept: vehicle
 Necessity: is_included_in Vehicle_by_drive_train
```



3.16 pav. Ekvivalenčių klasių transformavimo veiklos diagrama

3.17 paveiksle pavaizduota objektas turi save (angl. ObjectHasSelf) savybės transformavimo veiklos diagrama. Transformuojant savybę sukuriamas objekto tipas ir jam priskiriamas objekto savybės instancija. Sukuriama tekstinė išraiška su reikšme „purely_reflexive_verb_concept“. Objekto tipui sukuriamas terminas ir žymėjimas. Atlikus visus veiksmus atnaujinama koncepcinė schema.

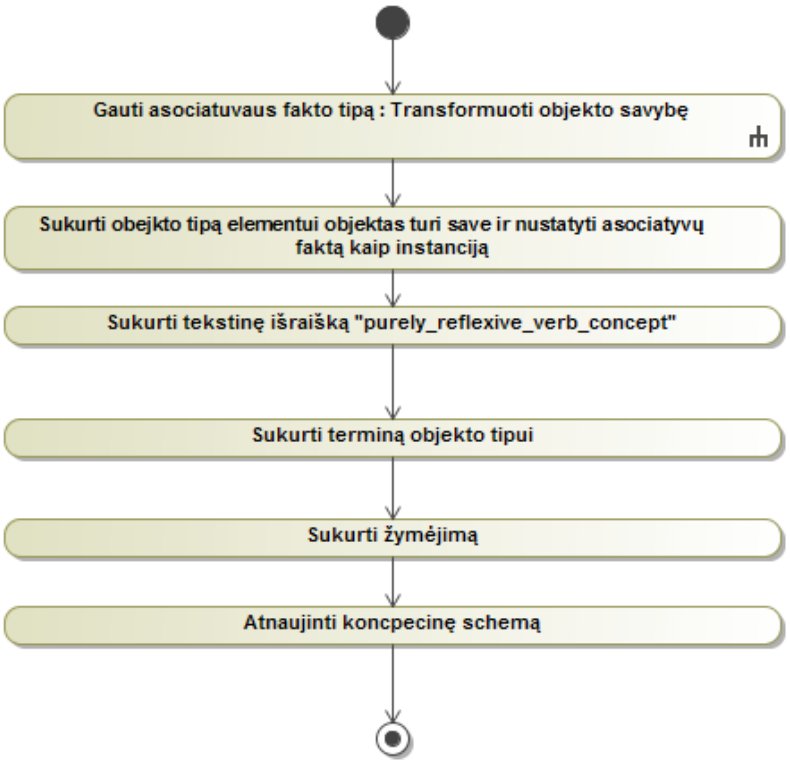
Tikslas: suformuoti konceptus, atitinkančius savęs turėjimo savybę.

<precond>: egzistuoja sukurti du bendrieji konceptai, suformuoti pagal taisyklę T_1.

T_{14}: *transform(class$_1$: Class, class$_2$: Class, ObjectHasSelf (class$_1$, class$_2$))* → SBVR *reflektyvus veiksmažodinis konceptas*

Pvz.: *transform(‘replaceable_part’, ‘replacing_part’, ObjectHasSelf(‘replaceable_part’, ‘replacing_part’)* →

```
replaceable_part
 General_concept: vehicle_part
replacing_part
 General_concept: vehicle_part
replaceable_part has replacing_part
 Concept_type: purely_reflexive_verb_concept
```



3.17 pav. Objektas turi save savybės transformavimo veiklos diagrama

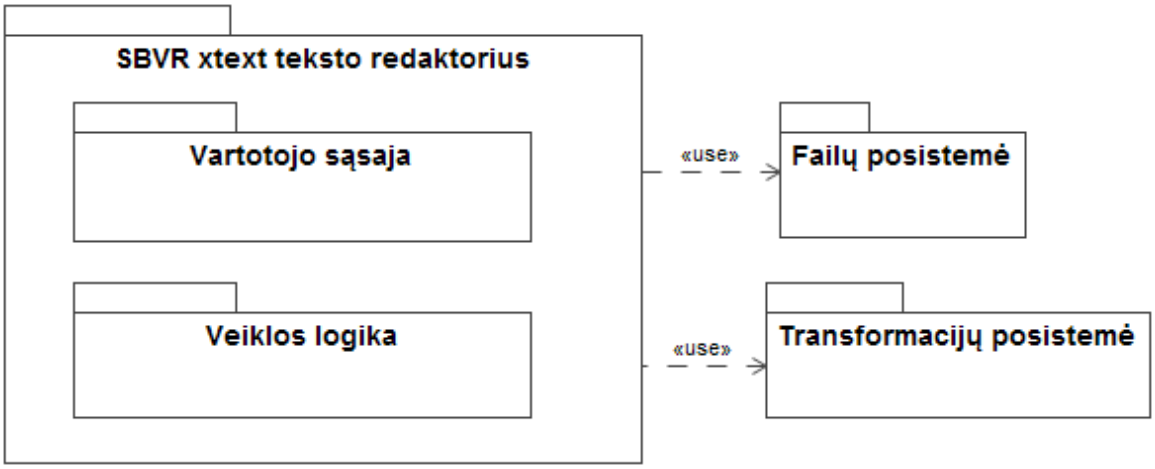
4. ONTOLOGIJŲ TRANSFORMACIJOS Į VEIKLOS ŽODYNUS IR TAISYKLES PROTOTIPO EKSPERIMENTINĖS REALIZACIJOS PROJEKTAS

4.1. Sprendimo architektūra

Kadangi transformacijos įrankis, yra siauro funkcionalumo, jį kurti kaip atskirą įrankio nėra praktiška, todėl pasirinktas sprendimas jį integruoti kaip įskiepį į SBVR teksto redaktorių.

4.1.1. Loginė sprendimo architektūra

Sistemos loginė architektūra susideda iš SBVR teksto redaktoriaus, kuris naudoja failų posistemę failams nuskaityti ir saugoti ir transformacijų posistemę, kuri atlieka transformacijos algoritmą. Ontologijų transformacijos į veiklos žodynus ir taisykles loginė architektūra su ja sudarančiais elementais pateikiama sistemos architektūros 4.1 paveiksle. Šiame darbe realizuojama transformacijų posistemė, kuri vėliau naudojama teksto redaktoriuje.



4.1 pav. Sistemos loginė architektūra

4.2. Projekto klasių modelis

Sukurtos realizacijos klasių diagrama pavaizduota 4.2 paveiksle. Transformacinis prototipas realizuotas naudojant vieną klasę. QVT kalboje metodai neturi privatumo apibrėžimo, todėl visi metodai, pavaizduoti be privatumo simbolių (+, -, #)

4.2. pav. Realizacijos kalsių diagrama

5. SPRENDIMO REALIZACIJA IR TESTAVIMAS

5.1. Sprendimo realizacijos ir veikimo aprašas

Norint atliki Ontologijų transformavimą į veiklos žodynus ir taisykles reikalinga speciali OMG grupės sukurta programavimo kalba skirta specialiai transformacijoms vykdyti. Pagal transformuojamus modelius palankiausia programavimo kalba QVT, kuri pagrįsta Java kalbos pagrindu ir realizuojama naudojant „Eclipse“ kūrimo aplinką (IDE). Projektinis sprendimas realizuotas naudojant QVTo transformacijų kalbą 3.5 versiją.

5.2. Testavimo modelis, duomenys, rezultatai

Testavimas atliekamas kuriamam eksperimentiniam prototipui suteikiant OWL XMI failus, jie tada yra transformuojami ir lyginami su Veiklos žodynus ir taisykles į ontologijas transformuojančio įrankio įvesties failais. Lyginamas elementų kiekis ir jų tarpusavio sąryšis leidžia nustatyti transformacijos korektiškumą ir pilnumą.

Testavimo įvesties failai ir gauti rezultatai pateikiami 9.5 priede.Analizuojant rezultatus matome, kad gauti failai yra identiški, nes abiejuose XMI failuose gautas vienodas atitinkamų elementų kiekis su tokiais pat sąryšiais, todėl galima teigti, kad su šiais testinais duomenimis eksperimentinio tyrimo IS transformavimo algoritmas veikia sėkmingai.

5.1 lentelė. Testavimo rezultatai

Elemento tipas	**Kiekis iš S2O transformatoriaus**	**Kiekis iš prototipo**
AtomicFormulation	1	1
ConceptualSchema	1	1
Concept	3	3
Designation	5	5
FactSymbol	1	1
NecessityFormulation	1	1
NonNegativeInteger	1	1
NumericRangeQuantification	1	1
Placeholder	1	1
PositiveInteger	2	2
Proposition	1	1
RoleBinding	2	2
SententialForm	1	1
Statement	1	1
Term	4	4

Text	5	5
UniversalQuantification	1	1
Variable	2	2

6. EKSPERIMENTINIS OWL TRANSFORMACIJOS Į SBVR TYRIMAS

6.1. Eksperimento planas

Eksperimentinio tyrimo metu tiriami, pagal šiuo darbu aprašytą transformacijos metodiką sukurto, transformatoriaus išvesties rezultatai. Norit užtikrinti transformacijų vientisumą, informacijos korektiškumą ir duomenų išsaugojimą, iš ontologijų sukurti žodynai ir taisyklės, naudojantis s2o transformatoriumi, yra transformuojami atgal į ontologijas ir tikrinama ar buvo prarasta duomenų, jei taip tai kiek ir kokių. Eksperimentui naudojama transporto priemonių ontologija, kuri buvo naudojama aprašant transformavimo taisykles taip pat agentų ir foto įrangos žodynas su atitinkančiomis ontologijomis paimtomis iš S2O transformacijos pavyzdžių, kad būtų galima įvertinti transformacijos pilnumą.

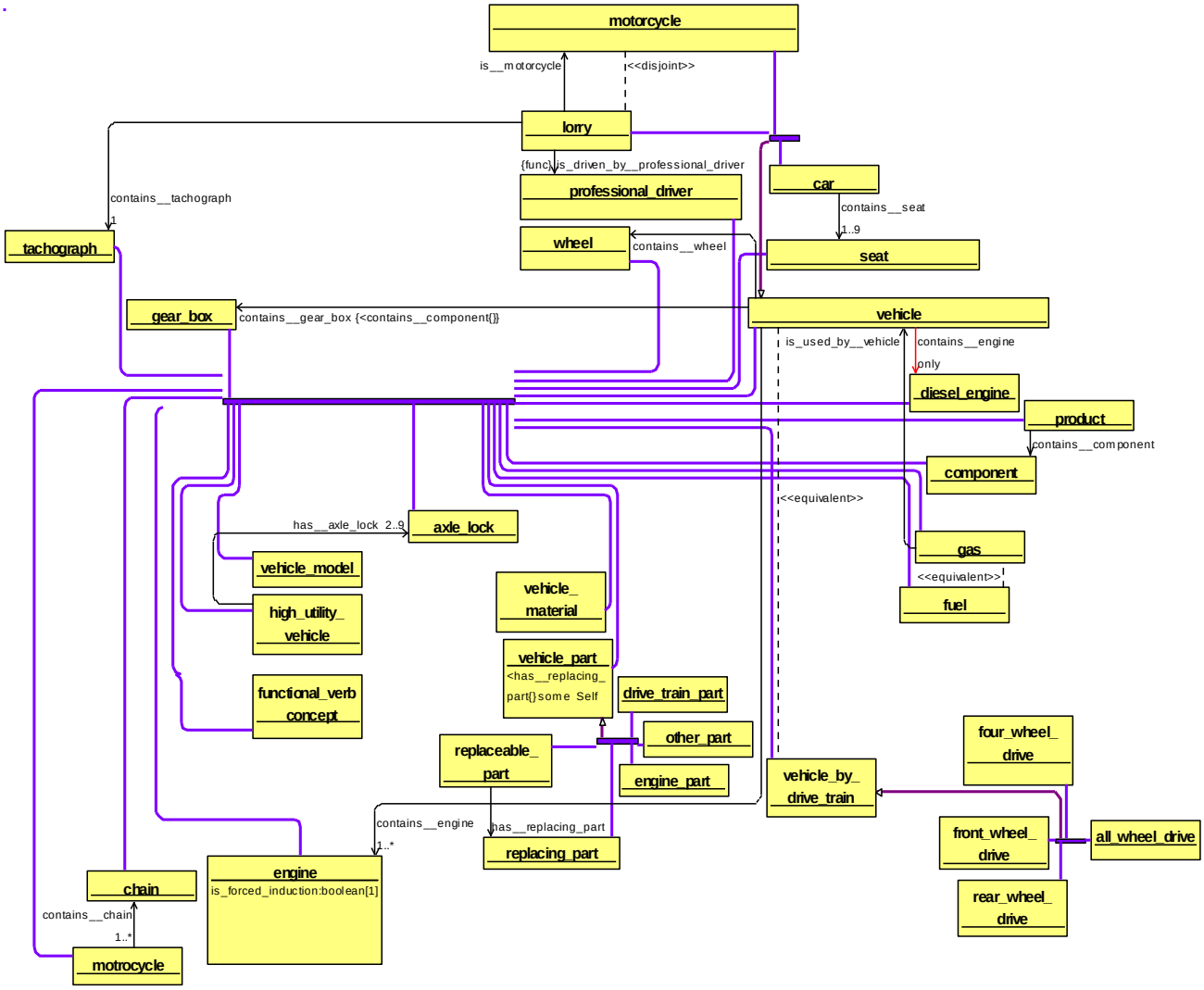
6.2. Eksperimento rezultatai

Pirma eksperimento dalis pradedama naudojantis sukurta transporto priemonių ontologija. Eksperimento tikslas parodyti taisyklių transformacijų rezultatus, pateikiant 6.2 lentelę su taisyklės nuoroda, įvesties ontologija, interpretuota veiklos žodyno ir taisyklės dalimi ir s2o transformatoriumi gautą atgalinės transformacijos elementą.

Transporto priemonių ontologijos parametrai pateikti 6.1 lentelėje. Klasių struktūrą atspindinti schema pateikta 6.1 paveiksle.

6.1 lentelė. Transporto priemonių ontologijos metrikos

Metrika	**Kiekis**
Aksiomos	252
Loginės aksiomos	61
Klasės	31
Objektų savybės	11
Duomenų savybės	5
Individai	0



6.1 pav. Transporto priemonių ontologijos klasių schema

6.2 lentelė. Transporto priemonių ontologijos tranformacijos rezultatai

Taisyklė	**Įvesties ontologijos elementas**	**Gauto XMI interpretuotas SBVR teiginys**	**Atgalinės transformacijos rezultato elementas**
T1	Class(<ns:s2o#vehicle>)	vehicle	Class(<ns:s2o#vehicle>)
T2	SubClassOf(<ns:s2o#lorry> <ns:s2o#vehicle>)	lorry General_concept: vehicle	SubClassOf(<ns:s2o#lorry> <ns:s2o#vehicle>)
T3	SubClassOf(<ns:s2o#lorry> ObjectAllValuesFrom(<ns:s2o#contains__engine> <ns:s2o#diesel_engine>))	It is necessary that lorry *contains* engine *that is* diesel_engine	SubClassOf(<ns:s2o#lorry> ObjectAllValuesFrom(<ns:s2o#contains__engine> <ns:s2o#diesel_engine>))

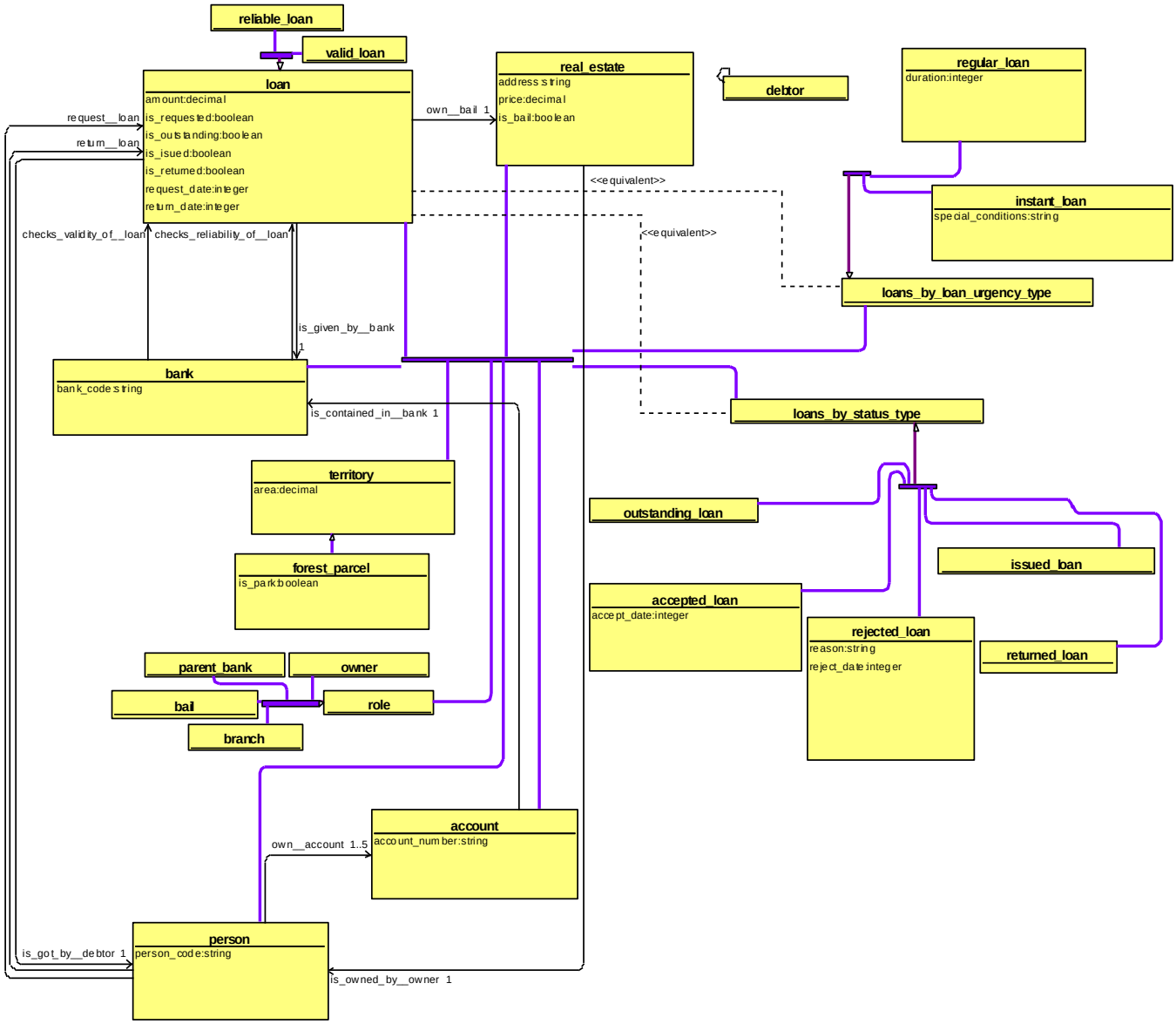
Taisyklė	Įvesties ontologijos elementas	Gauto XMI interpretuotas SBVR teiginys	Atgalinės transformacijos rezultato elementas
T4	DisjointClasses(<ns:s2o#lorry> <ns:s2o#motorcycle>)	It is impossible that lorry *is* motorcycle	DisjointClasses(<ns:s2o#motorcycle> <ns:s2o#lorry>)
T5	ObjectProperty(<ns:s2o#is_driven_by_ _professional_driver>)	lorry *is_driven_by* professional_driver	ObjectProperty(<ns:s2o#is_driven_by_ _professional_driver>)
T6	SubClassOf(<ns:s2o#vehicle> ObjectSomeValuesFrom(<ns:s2o#contains__engine> <ns:s2o#engine>))	It is necessary that vehicle *contains* engine	SubClassOf(<ns:s2o#vehicle> ObjectSomeValuesFrom(<ns:s2o#contains__engine> <ns:s2o#engine>))
T7	FunctionalObjectProperty(<ns:s2o#is_driven_by_ _professional_driver>)	It is necessary that lorry *is_driven_by at_most 1* professional_driver	FunctionalObjectProperty(<ns:s2o#is_driven_by_ _professional_driver>)
T8	SubClassOf(<ns:s2o#high_utility_vehicle > ObjectMinCardinality(2 <ns:s2o#has__axle_lock> <ns:s2o#axle_lock>)) SubClassOf(<ns:s2o#high_utility_vehicle > ObjectMaxCardinality(9 <ns:s2o#has__axle_lock> <ns:s2o#axle_lock>))	It is necessary that high_utility_vehicle *has* at_least 2 and at_most 9 axle_lock	SubClassOf(<ns:s2o#high_utility_vehicle> ObjectMinCardinality(2 <ns:s2o#has__axle_lock> <ns:s2o#axle_lock>))SubClassOf(<ns:s2o#high_utility_vehicle> ObjectMaxCardinality(9 <ns:s2o#has__axle_lock> <ns:s2o#axle_lock>))
T9	SubClassOf(<ns:s2o#lorry> ObjectExactCardinality(1 <ns:s2o#contains_ _tachograph> <ns:s2o#tachograph>))	It is necessary that lorry *contains* exactly 1 tachograph	SubClassOf(<ns:s2o#lorry> ObjectExactCardinality(1 <ns:s2o#contains_ _tachograph> <ns:s2o#tachograph>))
T10	Declaration(DataProperty(<ns:s2o# is_forced_induction >))	engine *is_forced_induction*	Declaration(DataProperty(<ns:s2o#is_forced_induction>))

Taisyklė	Įvesties ontologijos elementas	Gauto XMI interpretuotas SBVR teiginys	Atgalinės transformacijos rezultato elementas
T11	SubClassOf(<ns:s2o#engine> DataExactCardinality(1 <ns:s2o#is_forced_induction > < /XMLSchema#boolean>))	concept `engine` *incorporates* characteristic `engine` *is_forced_induction*`	SubClassOf(<ns:s2o#engine> DataExactCardinality(1 <ns:s2o#is_forced_induction> < XMLSchema#boolean>))
T12	EquivalentClasses(<ns:s2o#gas> <ns:s2o#fuel>)	fuel *is coextensive with* gas	EquivalentClasses(<ns:s2o#fuel> <ns:s2o#gas>)
T13	DisjointUnion(<ns:s2o#vehicle_by_drive_train> <ns:s2o#front_wheel_drive> <ns:s2o#rear_wheel_drive>)	Vehicle_by_drive_train Necessity: segmentation *for* general_concept vehicle that *subdivides* vehicle by vehicle_drive_train front_wheel_drive General_concept: vehicle Necessity: *is_included_in* Vehicle_by_drive_train rear_wheel_drive General_concept: vehicle Necessity: *is_included_in* Vehicle_by_drive_train	DisjointUnion(<ns:s2o#vehicle_by_drive_train> <ns:s2o#front_wheel_drive> <ns:s2o#rear_wheel_drive>)
T14	SubClassOf(<ns:s2o#vehicle_part> ObjectHasSelf(<ns:s2o#has_ _replacing_part>))	replaceable_part General_concept: vehicle_part replacing_part General_concept: vehicle_part replaceable_part *has* replacing_part Concept_type: purely_reflexive verb_concept	SubClassOf(<ns:s2o#vehicle_part> ObjectHasSelf(<ns:s2o#has_ _replacing_part>))

Antroje eksperimento dalyje paimta paskolų ontologija [15], kurios schema atvaizduotas 6.2 paveiksle transformuojama naudojantis sukurtu eksperimentiniu įrankiu ir transformuota S2O transformatoriumi atgal į ontologiją ir lyginama su pradine ontologija. Palyginimo rezultatai pateikti 6.3 lentelėje.

6.3 lentelė. Paskolų ontologijos transformacijų metrikos

Metrika	Agentų ontologija	Atgalinės transformacijos rezultatas
Aksiomos	303	303
Loginės aksiomos	85	85
Klasės	24	24
Objektų savybės	10	10
Duomenų savybės	20	20
Individai	0	0



6.2 pav. Paskolų ontologijos schema [15]

Paskutinėje eksperimento dalyje paimame ontologiją su elementais, kurie nėra transformuojami aprašytos metodikos. Šiam tikslui naudojama foto įrangos ontologija [15], kurios metrikos pateiktos 6.4 lentelėje.

6.4 lentelė Foto įrangos ontologijos charakteristikos

Metrika	**Kiekis**
Aksiomos	639
Loginės aksiomos	214
Klasės	45
Objektų savybės	35
Duomenų savybės	14
Individai	11

Transformavus ontologiją sukurtu prototipiniu įrankiu, gauname XMI failą, tačiau jo transformuoti atgal į ontologiją nepavyko, todėl 6.5 lentelėje lyginame gauto failo elementus, su foto įrangos ontologiją atitinkančio žodyno ir taisyklių XMI failu.

6.5 lentelė. Foto įrangos veiklos žodyno ir taisyklių XMI failų palyginimas

Elementas	**Kiekis iš ontologiją atitinkančio failo XMI**	**Kiekis iš prototipo įrankio**
AtLeastNQuantification	1	1
AtMostNQuantification	7	7
AtomicFormulation	53	33
CategorizationScheme	1	1
ClosedProjection	14	0
ConceptualSchema	1	1
concept	164	165
role	183	167
Conjunction	1	0
Designation	376	367
Disjunction	3	0
ExactlyNQuantification	6	6
ExistentialQuantification	19	17
FactSymbol	125	127

Elementas	**Kiekis iš ontologiją atitinkančio failo XMI**	**Kiekis iš prototipo įrankio**
IndividualConcept	2	0
InstantiationFormulation	12	0
LogicalNegation	10	7
Name	16	1
NecessityFormulation	28	32
NonNegativeInteger	19	25
NumericRangeQuantification	1	1
Placeholder	247	248
PositiveInteger	25	131
Proposition	28	32
RoleBinding	106	66
Segmentation	2	1
containsCategory	6	3
SententialForm	125	127
Statement	30	32
Term	235	240
Text	243	276
UniversalQuantification	36	32
Variable	94	35

Pagal gautus palyginmo rezultatus matome, kad elementų kiekis nėra vienodas abejuose XMI failuose. Dėl įrankių, galinčių apdoroti šiuos XMI failus ir transformuoti į žodynus, stokos palyginimas sudėtingas procesas ir gali ne iki galo išaiškinti problemas. Tačiau šis gautas rezultatas vis tiek padės atskleisti darbe aprašytos transformacijos neapimtus elementus.

Analizę pradedame nuo elementų kurie neturėjo visiškai jokio atitikmens gautame rezultate uždara projekcija (angl. *ClosedProjection*), konjukcija (angl *Conjunction*), disjunkcija (angl. *Disjunction*), indivualus konceptai (angl. *IndividualConcept*) ir momentinė formuluotė (angl. *InstantiationFormulation*) visi gaunami transformuojant konceptų aprašus (angl. *Definition*). Konceptai su tokiais aprašais dar vadinami individualiais konceptais. Pagal elementų trūkumą galime tegti, kad jie visiškai netransformuojami. Dėl šių netransformuojamų elementų gauname mažesnį ir atominių formuluočių skaičių, nes ontologijoje aprašyta net 13 individų. Rolės

elementai kaip ir rolių priskyrimas ir kintamieji tiesiogiai susiję su atominėmis formuluotėmis, dėl to jų gauti kiekiai atitinkamai mažesni.

Kaip jau buvo minėta ontologijos reikalavimų skyriuje, ontologijos savybės privalo turėti nustatytas tiek diapazono, tiek srities reikšmes, bet šioje ontologijoje yra savybė be srities reikšmės, todėl gaunamas nekorektiškas konceptas, dėl kurio, lyginant su pradiniu žodynu, gauname vienu konceptu daugiau, kuris rišasi su vietos išskyrimo, teigiamo skaičiaus ir kintamojo elementais ir padidina jų kieki.

Egzistencinio kiekybiškumo (angl *ExistentialQuantification*) elementai naudojami aprašant sandaros būtinumą arba nesusikertančius elementus ir atspindi taisykles. Peržiūrėjus taisyklių komplektą pateiktą priede 9.6, matome, kad egzistuoja dvi taisyklės aprašančios koegzistavimo ir nesusikirtimo saryšius tarp individų, dėl šios priežasties gauname dviem egzistencinio kiekybiškumo elementais mažiau nei pradiniame faile šie elementai kiek vienas siejami su pora būtinumo formuluočių ir jie atitinkamai su ne neigiama sveikąja reikšme, pasiulymu, sakinine forma ir teiginiu.

Pagal teigiamų skaičių kiekį matome, kad transformacijos prototipas veikia neefektyviai resursų požiūriu, nes generuoja perteklini teigiamų sveikūjų skaičių elementų aibe. Ši problema iškyla dėl to, kad QVTo priešingai nei kitus elemetus, kurių transformacijas tikrina, kad nebūtų transformuojamas elementas du kartus, sveikuosius skaičius ji kuria kiek vieno kreipimosi metu, todėl sukuriamas aibė elementų su vienodomis reikšmėmis.

Segmentacijos ir kategorizacijos elementų kiekis taip pat mažesnis, tai vyksta dėl to, kad ontologija segmentuoja elementus dvejomis grupėmis, tačiau vienoje iš segmentacijos grupių aprašoma conceptų segmentacija, o kitoje individų segmentacija.

6.3. Sprendimo veikimo ir savybių analizė

Darbo metu sukurta metodika ir prototipinis įrankis, kuris leidžia atlikti ontologijos aprašytos OWL2 kalba XMI formatu, transformaciją į veiklos žodynus ir taisykles. Eksperimento metu naudojant prototipinį įrankį buvo transformuotos trys ontologijos, viena sukurta taisyklių teisingumui parodyti ir dvi iš interneto paimtos ontologijos. Transformacijos metu transformuojant pirmas dvi ontologijas ir įvykdžius gauto rezultato atgalinę transformaciją matome, kad ontologija liko struktūriškai nepakitusi ir metodika panaudota prototipiniame įrankyje suveikė kaip ir planuota. Tačiau vykdant trečiosios ontologijos transformaciją gautas failas nebuvo korektiškas, todėl jis buvo lyginamas XMI formatu su veiklos žodynu ir taisyklėmis atitinkančiais transformuotą ontologiją. Iš gautų rezultatų galime teigti, kad ontologijoje esant elementų, kurie

nėra transformuojami apibrėžta metodika, susidaro situacija, kad kai kurie elementai yra interpretuojami ir netransformuojami arba transformuojami į kito tipo elementus. Iš eksperimento nustatyta, kad taisyklės visiškai nedengia konceptų aprašų (angl. *Definition*) ir iššaukiamos kirtinės situacijos kai ontologijoje egzistuoje bet kokie individų elementai. Taip pat nustatyta, kad reikalinga papildoma realizacijos optimizacija galinti tikrinti jau sukurtus elementus, kurie nėra automatiškai tikrinami pačios QVT kalbos derintuvo. Norint sukurti metodiką, kuri atliktų visų galimų ontologijos savybių transformaciją į veiklos žodynus ir taisykles reikėtų laikytis tų pačių principų kaip aprašomos šio darbo taisyklėse.

Pagal gautus transformacijų rezultatus, galima teigti, kad transformavimo taisyklės veikia korektiškai ir transformaciją naudojant S2O ir šio darbo metu sukurtą transformatorių, galima vykdyti į abi puses, kai ontologijose nėra elementų kurie netransformuojami šios metodikos aprašytomis taisyklėmis. Tačiau norit pilnai ištirti sukurtą sprendimą reikalingas papildomas įrankis gebantis transformuoti ontologijas į XMI formatą, nes dabartinėmis sąlygomis sudėtinga kurti įvesties duomenis. Taip pat norint patikimai naudoti transformatorių būtina praplėsti individų ir aprašų transformacijos galimybes.

7. IŠVADOS

Ontologijų panaudojimas įvairių sričių informacinių sistemų kūrime tampa vis populiaresnis, ypatingai sudėtingesnėse, tačiau jos sunkiau suprantamos ir nėra tokios intuityvios kaip veiklos žodynai ir taisyklės. Šiame darbe buvo analizuotas procesas, kaip tam tikros dalykinės srities ontologijos aprašas OWL2 galėtų būti transformuotas į lengviau suprantamą veiklos žodyną ir taisykles. Apibendrinant visus darbo rezultatus suformuotos tokios išvados:

1. Atlikus analizę galima teigti, kad ontologijos ir veiklos žodynai ir taisyklės inovatyvios technologijos kurių naudojimas tik augs, tačiau dėl lengvesnio veiklos žodynų ir taisyklių suprantamumo, reikalinga transformacijos iš OWL2 į SBVR metodika
2. Literatūros šaltinių analizė parodė, kad nėra standarto kaip turėtų būti vykdomas ontologijų transformavimas į veiklos žodynus ir taisykles, tačiau pateikiami patikimi algoritmai, kuriais galima remtis.
3. Sukurta metodika pranašesnė už kitus nagrinėtus sprendimus, nes nereikalauja jokių papildomų elementų kūrimo ir veikia be rankinio įsikišimo, kai transformuojama ontologija yra korektiška.
4. Sukūrus transformacijos metodiką naudojantį transformatorių matome, kad efektyviai galima transformuoti, pagrindines OWL2 schemos konstrukcijas, bet metodiką dar reikia plėsti.
5. Eksperimento metu transformuojant tiek transporto, tiek paskolų ontologijas, gauti puikūs transformacijos rezultatai ir nulinis duomenų nutekėjimas transformacijos metu, bet metodiką galima laikyti patikima tik jeigu ontologijos elementų aibė atitinka taisyklių aibę.
6. Sukurtą metodiką galima efektyviai taikyti verslo ir sistemų kūrimo procesuose, tačiau, kad pilnai būtų palaikoma visa OWL2 elementų aibė metodą dar reikia tobulinti, praplėsti duomenų savybių transformavimo galimybes, pridėti lemavimą norint naudoti lietuvių kalbą.

8. LITERATŪRA

[1] OMG , „Semantics of Business Vocabulary and Business Rules (SBVR) Version 1.2. OMG document number: formal/2013-11-04,“ 4, lapkričio, 2013.

[2] J. Karpovic and L. Nemuraite, „Transforming SBVR Business semantics into Web Ontology Language OWL2: Main Concepts,“ įtraukta *17th International Conference on Information and Software Technologies (IT 2011)*, Kaunas, 2011.

[3] Gintare Bernotaitytė, Lina Nemūraitė, Rita Butkienė, Bronius Paradauskas, „Developing SBVR Vocabularies and Business Rules from OWL2 Ontologies“.

[4] Emiliano Reynares, Ma. Laura Caliusco, Ma. Rosa Galli, „SVBR to OWL 2 mappings: An Automatable and Structural-Rooted Aproach,“ *CLEI Eletronic Journal*, t. 17, nr. 2, 2014 gruodis.

[5] Elisa Kendall, Mark H. Linehan IMB, „Maping SBVR To OWL,“ 2013. [Tinkle]. Available: http://domino.watson.ibm.com/library/CyberDig.nsf/1e4115aea78b6e7c85256b360066f0d4/a9777f4edb2552ae85257b34004c4eb3!OpenDocument. [Kreiptasi 8 sausio 2015].

[6] N. Guarino, C. Welty, „Tutorial on Conceptual Modeling and Ontological Analysis,“ 31 liepos 2000. [Tinkle]. Available: http://www.cs.vassar.edu/~weltyc/aaai-2000/. [Kreiptasi 8 gruodžio 2015].

[7] R. Studer, R. Benjamins, D. Fensel, „Data & Knowledge Engineering,“ įtraukta *Knowledge engineering: Principles and methods*, 1998, pp. 161-198.

[8] H. Knublauch, D. Oberle, P. Tetlow ir E. (.-0.-0. Wallace, „ A Semantic Web Primer for Object-Oriented Software Developers,“ 09 kovo 2006. [Tinkle]. Available: http://www.w3.org/2001/sw/BestPractices/SE/ODSD/. [Kreiptasi 15 gruodžio 2014].

[9] P. F. Patel-Schneider ir B. Motik, „ OWL 2 Web Ontology Language Mapping to RDF Graphs,“ 27 sausio 2009. [Tinkle]. Available: http://www.w3.org/TR/2009/REC-owl2-mapping-to-rdf-20091027/. [Kreiptasi 15 gruodžio 2014].

[10] W3C, „OWL Web Ontology Language Overview,“ 10 vasario 2004. [Tinkle]. Available: http://www.w3.org/TR/owl-features/. [Kreiptasi 14 lapkričio 2014].

[11] R. v. Haarst, „BRCommunity,“ rugpjūtis 2014. [Tinkle]. Available: http://www.brcommunity.com/b770.php. [Kreiptasi 15 lapkričio 2014].

[12] J.Karpovič, G.Bernotaitytė, L.Ablonskis, L. Nemuraitė, „SUITABILITY OF SEMANTICS OF BUSINESS VOCABULARY AND BUSINESS RULES (SBVR) TO REPRESENT OWL 2 ONTOLOGIES,“ Kaunas University of Technology, Department of Information Systems, Kaunas.

[13] Emiliano Reynares, Ma. Laura Caliusco, Ma. Rosa Galli, „An Automatable Approach for SBVR to OWL 2,“ įtraukta *CIbSE 2013 | XVI Conferencia Iberoamericana de Ingeniería de Software*, Montevidėjas, Urugvajus, 2013.

[14] Gintare Krisciuniene, Lina Nemuraite, Rita Butkiene, Bronius Paradauskas, „Rules for Transforming OWL 2 Ontology into SBVR“.

[15] Jaroslav Karpovic, Algirdas Sukys, prof. dr. Lina Nemuraite., „s2o: SBVR to OWL 2 Converter,“ Department of Information systems, [Tinkle]. Available: http://s2o.isd.ktu.lt/about.php. [Kreiptasi 10 gruodžio 2014].

[16] Bast, Wim; Murphree, Michael; Lawley, Michael; Duddy, Keith; Belaunde, Mariano; Griffin, Catherine; Sendall, Shane; Vojtisek, Didier; Steel, Jim; Helsen, Simon; Tratt, Laurence; Reddy, Sreedhar; Venkatesh, R.; Blanc, Xavier; Dvorak, Radek; Willink, Ed, „"Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT)",“ vasaris 2015. [Tinkle]. Available: http://www.omg.org/spec/QVT/1.2. [Kreiptasi 15 gegužės 2016].

[17] A. Šukys, L. Nemuraitė, E. Šinkevičius, B. Paradauskas, „Querying ontologies on the base of semantics of business vocabularies and business rules.,“ įtraukta *Information Technologies' 2011 : proceedings of the 17th international conference on Information and Software Technologies*, Kaunas, Lithuania, 2011.

9. PRIEDAI

9.1. priedas. Transporto priemonių žodynas ir taisyklės

```
vehicle
lorry
 General_concept: vehicle
motorcycle
 General_concept: vehicle
car
 General_concept: vehicle

vehicle_drive_train
 Concept_type: categorization_type
 Necessity: is_for general_concept vehicle

Vehicle_by_drive_train
 Necessity: segmentation for general_concept vehicle that subdivides vehicle
by vehicle_drive_train
front_wheel_drive
 General_concept: vehicle
 Necessity: is_included_in Vehicle_by_drive_train
rear_wheel_drive
 General_concept: vehicle
 Necessity: is_included_in Vehicle_by_drive_train
four_wheel_drive
 General_concept: vehicle
 Necessity: is_included_in Vehicle_by_drive_train
all_wheel_drive
 General_concept: vehicle
 Necessity: is_included_in Vehicle_by_drive_train

fuel
gas
fuel is_coextensive_with gas

gas is_used_by vehicle

vehicle_weight
 General_concept: integer

vehicle has vehicle_weight
 Concept_type: property_association

vehicle_material
vehicle_material is_light_weight
vehicle_model
name
model_name
vehicle_model has name
 Concept_type: property_association
vehicle_model has model_name
 Concept_type: property_association
 General_concept: vehicle_model has name


high_utility_vehicle
axle_lock
high_utility_vehicle has axle_lock
```

```
engine
diesel_engine
vehicle contains engine

seat
car contains seat


product
component
product contains component
gear_box
vehicle contains gear_box
 General_concept: product contains component

engine is_forced_induction
concept `engine` incorporates characteristic `engine is_forced_induction`

vehicle_part
engine_part
           General_concept: vehicle_part
drive_train_part
           General_concept: vehicle_part
other_part
           General_concept: vehicle_part
replaceable_part
 General_concept: vehicle_part
replacing_part
 General_concept: vehicle_part
replaceable_part has replacing_part
 Concept_type: purely_reflexive_verb_concept

professional_driver
lorry is_driven_by professional_driver

wheel
vehicle contains wheel

tachograph
lorry contains tachograph


motrocycle
chain
motrocycle contains chain
 Concept_type: partitive_verb_concept
 Concept_type: transitive_verb_concept
 Concept_type: asymmetric_verb_concept

It is necessary that lorry contains engine that is diesel_engine
It is necessary that vehicle contains engine
It is impossible that lorry is motorcycle
It is necessary that motrocycle contains at_least 1 chain
It is necessary that lorry is_driven_by at_most 1 professional_driver
It is necessary that car contains at_least 1 and at_most 9 seat
It is necessary that lorry contains exactly 1 tachograph
It is necessary that high_utility_vehicle has at_least 2 and at_most 9
axle_lock
It is necessary that vehicle contains at_most 16 wheel
```

9.2. priedas. Transporto priemonių ontologija

Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(ns:=<http://isd.ktu.lt/semantika/>)
Ontology(<http://isd.ktu.lt/semantika/>
Declaration(AnnotationProperty(<ns:#label_sbvr>))
Declaration(AnnotationProperty(<ns:#label>))
Declaration(Class(<ns:#vehicle>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#vehicle> "vehicle"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#vehicle> "vehicle"@en)
AnnotationAssertion(<ns:#label> <ns:#vehicle> "vehicle")
Declaration(Class(<ns:#lorry>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#lorry> "lorry"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#lorry> "lorry"@en)
AnnotationAssertion(<ns:#label> <ns:#lorry> "lorry")
Declaration(Class(<ns:#motorcycle>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#motorcycle> "motorcycle"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#motorcycle> "motorcycle"@en)
AnnotationAssertion(<ns:#label> <ns:#motorcycle> "motorcycle")
Declaration(Class(<ns:#car>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#car> "car"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#car> "car"@en)
AnnotationAssertion(<ns:#label> <ns:#car> "car")
Declaration(Class(<ns:#fuel>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#fuel> "fuel"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#fuel> "fuel"@en)
AnnotationAssertion(<ns:#label> <ns:#fuel> "fuel")
Declaration(Class(<ns:#gas>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#gas> "gas"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#gas> "gas"@en)
AnnotationAssertion(<ns:#label> <ns:#gas> "gas")
Declaration(Class(<ns:#vehicle_material>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#vehicle_material> "vehicle_material"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#vehicle_material> "vehicle material"@en)
AnnotationAssertion(<ns:#label> <ns:#vehicle_material> "vehicle material")
Declaration(Class(<ns:#vehicle_model>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#vehicle_model> "vehicle_model"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#vehicle_model> "vehicle model"@en)
AnnotationAssertion(<ns:#label> <ns:#vehicle_model> "vehicle model")
Declaration(Class(<ns:#high_utility_vehicle>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#high_utility_vehicle> "high_utility_vehicle"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#high_utility_vehicle> "high utility vehicle"@en)
AnnotationAssertion(<ns:#label> <ns:#high_utility_vehicle> "high utility vehicle")
Declaration(Class(<ns:#axle_lock>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#axle_lock> "axle_lock"@en)

AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#axle_lock> "axle lock"@en)
AnnotationAssertion(<ns:#label> <ns:#axle_lock> "axle lock")
Declaration(Class(<ns:#engine>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#engine> "engine"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#engine> "engine"@en)
AnnotationAssertion(<ns:#label> <ns:#engine> "engine")
SubClassOf(<ns:#engine> DataExactCardinality(1 <ns:#is_forced_induction> <http://www.w3.org/2001/XMLSchema#boolean>))
Declaration(Class(<ns:#diesel_engine>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#diesel_engine> "diesel_engine"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#diesel_engine> "diesel engine"@en)
AnnotationAssertion(<ns:#label> <ns:#diesel_engine> "diesel engine")
Declaration(Class(<ns:#seat>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#seat> "seat"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#seat> "seat"@en)
AnnotationAssertion(<ns:#label> <ns:#seat> "seat")
Declaration(Class(<ns:#product>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#product> "product"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#product> "product"@en)
AnnotationAssertion(<ns:#label> <ns:#product> "product")
Declaration(Class(<ns:#component>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#component> "component"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#component> "component"@en)
AnnotationAssertion(<ns:#label> <ns:#component> "component")
Declaration(Class(<ns:#gear_box>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#gear_box> "gear_box"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#gear_box> "gear box"@en)
AnnotationAssertion(<ns:#label> <ns:#gear_box> "gear box")
Declaration(Class(<ns:#vehicle_part>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#vehicle_part> "vehicle_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#vehicle_part> "vehicle part"@en)
AnnotationAssertion(<ns:#label> <ns:#vehicle_part> "vehicle part")
Declaration(Class(<ns:#engine_part>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#engine_part> "engine_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#engine_part> "engine part"@en)
AnnotationAssertion(<ns:#label> <ns:#engine_part> "engine part")
Declaration(Class(<ns:#drive_train_part>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#drive_train_part> "drive_train_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#drive_train_part> "drive train part"@en)
AnnotationAssertion(<ns:#label> <ns:#drive_train_part> "drive train part")
Declaration(Class(<ns:#other_part>))

AnnotationAssertion(<ns:#label_sbvr> <ns:#other_part> "other_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#other_part> "other part"@en)
AnnotationAssertion(<ns:#label> <ns:#other_part> "other part")
Declaration(Class(<ns:#replaceable_part>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#replaceable_part> "replaceable_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#replaceable_part> "replaceable part"@en)
AnnotationAssertion(<ns:#label> <ns:#replaceable_part> "replaceable part")
Declaration(Class(<ns:#replacing_part>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#replacing_part> "replacing_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#replacing_part> "replacing part"@en)
AnnotationAssertion(<ns:#label> <ns:#replacing_part> "replacing part")
Declaration(Class(<ns:#professional_driver>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#professional_driver> "professional_driver"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#professional_driver> "professional driver"@en)
AnnotationAssertion(<ns:#label> <ns:#professional_driver> "professional driver")
Declaration(Class(<ns:#wheel>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#wheel> "wheel"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#wheel> "wheel"@en)
AnnotationAssertion(<ns:#label> <ns:#wheel> "wheel")
Declaration(Class(<ns:#tachograph>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#tachograph> "tachograph"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#tachograph> "tachograph"@en)
AnnotationAssertion(<ns:#label> <ns:#tachograph> "tachograph")
Declaration(Class(<ns:#motrocycle>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#motrocycle> "motrocycle"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#motrocycle> "motrocycle"@en)
AnnotationAssertion(<ns:#label> <ns:#motrocycle> "motrocycle")
Declaration(Class(<ns:#chain>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#chain> "chain"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#chain> "chain"@en)
AnnotationAssertion(<ns:#label> <ns:#chain> "chain")
Declaration(Class(<ns:#vehicle_by_drive_train>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#vehicle_by_drive_train> "vehicle_by_drive_train"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#vehicle_by_drive_train> "vehicle by drive train"@en)
AnnotationAssertion(<ns:#label> <ns:#vehicle_by_drive_train> "vehicle by drive train")
Declaration(Class(<ns:#functional_verb_concept>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#functional_verb_concept> "functional_verb_concept"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#functional_verb_concept> "functional verb concept"@en)

AnnotationAssertion(<ns:#label> <ns:#functional_verb_concept> "functional verb concept")
Declaration(Class(<ns:#front_wheel_drive>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#front_wheel_drive> "front_wheel_drive"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#front_wheel_drive> "front wheel drive"@en)
AnnotationAssertion(<ns:#label> <ns:#front_wheel_drive> "front wheel drive")
Declaration(Class(<ns:#rear_wheel_drive>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#rear_wheel_drive> "rear_wheel_drive"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#rear_wheel_drive> "rear wheel drive"@en)
AnnotationAssertion(<ns:#label> <ns:#rear_wheel_drive> "rear wheel drive")
Declaration(Class(<ns:#four_wheel_drive>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#four_wheel_drive> "four_wheel_drive"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#four_wheel_drive> "four wheel drive"@en)
AnnotationAssertion(<ns:#label> <ns:#four_wheel_drive> "four wheel drive")
Declaration(Class(<ns:#all_wheel_drive>))
AnnotationAssertion(<ns:#label_sbvr> <ns:#all_wheel_drive> "all_wheel_drive"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#all_wheel_drive> "all wheel drive"@en)
AnnotationAssertion(<ns:#label> <ns:#all_wheel_drive> "all wheel drive")
Declaration(DataProperty(<ns:#is_forced_induction>))
DataPropertyDomain(<ns:#is_forced_induction> <ns:#engine>)
DataPropertyRange(<ns:#is_forced_induction> <http://www.w3.org/2001/XMLSchema#boolean>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#is_forced_induction> "is_forced_induction"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#is_forced_induction> "is forced induction"@en)
AnnotationAssertion(<ns:#label> <ns:#is_forced_induction> "is forced induction")
Declaration(ObjectProperty(<ns:#contains__engine>))
ObjectPropertyDomain(<ns:#contains__engine> <ns:#vehicle>)
ObjectPropertyRange(<ns:#contains__engine> <ns:#engine>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__engine> "vehicle contains engine"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__engine> "vehicle contains engine"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__engine> "vehicle contains engine")
Declaration(ObjectProperty(<ns:#is__motorcycle>))
ObjectPropertyDomain(<ns:#is__motorcycle> <ns:#lorry>)
ObjectPropertyRange(<ns:#is__motorcycle> <ns:#motorcycle>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#is__motorcycle> "lorry is motorcycle"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#is__motorcycle> "lorry is motorcycle"@en)
AnnotationAssertion(<ns:#label> <ns:#is__motorcycle> "lorry is motorcycle")
Declaration(ObjectProperty(<ns:#is_used_by__vehicle>))
ObjectPropertyDomain(<ns:#is_used_by__vehicle> <ns:#gas>)
ObjectPropertyRange(<ns:#is_used_by__vehicle> <ns:#vehicle>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#is_used_by__vehicle> "gas is_used_by vehicle"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#is_used_by__vehicle> "gas is used by vehicle"@en)

AnnotationAssertion(<ns:#label> <ns:#is_used_by__vehicle> "gas is used by vehicle")
Declaration(ObjectProperty(<ns:#has__axle_lock>))
ObjectPropertyDomain(<ns:#has__axle_lock> <ns:#high_utility_vehicle>)
ObjectPropertyRange(<ns:#has__axle_lock> <ns:#axle_lock>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#has__axle_lock> "high_utility_vehicle has axle_lock"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#has__axle_lock> "high utility vehicle has axle lock"@en)
AnnotationAssertion(<ns:#label> <ns:#has__axle_lock> "high utility vehicle has axle lock")
Declaration(ObjectProperty(<ns:#contains__seat>))
ObjectPropertyDomain(<ns:#contains__seat> <ns:#car>)
ObjectPropertyRange(<ns:#contains__seat> <ns:#seat>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__seat> "car contains seat"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__seat> "car contains seat"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__seat> "car contains seat")
Declaration(ObjectProperty(<ns:#contains__component>))
ObjectPropertyDomain(<ns:#contains__component> <ns:#product>)
ObjectPropertyRange(<ns:#contains__component> <ns:#component>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__component> "product contains component"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__component> "product contains component"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__component> "product contains component")
Declaration(ObjectProperty(<ns:#contains__gear_box>))
ObjectPropertyDomain(<ns:#contains__gear_box> <ns:#vehicle>)
ObjectPropertyRange(<ns:#contains__gear_box> <ns:#gear_box>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__gear_box> "vehicle contains gear_box"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__gear_box> "vehicle contains gear box"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__gear_box> "vehicle contains gear box")
SubObjectPropertyOf(<ns:#contains__gear_box> <ns:#contains__component>)
Declaration(ObjectProperty(<ns:#has__replacing_part>))
ObjectPropertyDomain(<ns:#has__replacing_part> <ns:#replaceable_part>)
ObjectPropertyRange(<ns:#has__replacing_part> <ns:#replacing_part>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#has__replacing_part> "replaceable_part has replacing_part"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#has__replacing_part> "replaceable part has replacing part"@en)
AnnotationAssertion(<ns:#label> <ns:#has__replacing_part> "replaceable part has replacing part")
Declaration(ObjectProperty(<ns:#is_driven_by__professional_driver>))
ObjectPropertyDomain(<ns:#is_driven_by__professional_driver> <ns:#lorry>)
ObjectPropertyRange(<ns:#is_driven_by__professional_driver> <ns:#professional_driver>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#is_driven_by__professional_driver> "lorry is_driven_by professional_driver"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#is_driven_by__professional_driver> "lorry is driven by professional driver"@en)
AnnotationAssertion(<ns:#label> <ns:#is_driven_by__professional_driver> "lorry is driven by

professional driver")
Declaration(ObjectProperty(<ns:#contains__wheel>))
ObjectPropertyDomain(<ns:#contains__wheel> <ns:#vehicle>)
ObjectPropertyRange(<ns:#contains__wheel> <ns:#wheel>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__wheel> "vehicle contains wheel"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__wheel> "vehicle contains wheel"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__wheel> "vehicle contains wheel")
Declaration(ObjectProperty(<ns:#contains__tachograph>))
ObjectPropertyDomain(<ns:#contains__tachograph> <ns:#lorry>)
ObjectPropertyRange(<ns:#contains__tachograph> <ns:#tachograph>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__tachograph> "lorry contains tachograph"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__tachograph> "lorry contains tachograph"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__tachograph> "lorry contains tachograph")
Declaration(ObjectProperty(<ns:#contains__chain>))
ObjectPropertyDomain(<ns:#contains__chain> <ns:#motrocycle>)
ObjectPropertyRange(<ns:#contains__chain> <ns:#chain>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#contains__chain> "motrocycle contains chain"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#contains__chain> "motrocycle contains chain"@en)
AnnotationAssertion(<ns:#label> <ns:#contains__chain> "motrocycle contains chain")
SubClassOf(<ns:#vehicle_part> ObjectHasSelf(<ns:#has__replacing_part>))
EquivalentClasses(<ns:#fuel> <ns:#gas>)
SubClassOf(<ns:#lorry> <ns:#vehicle>)
SubClassOf(<ns:#motorcycle> <ns:#vehicle>)
SubClassOf(<ns:#car> <ns:#vehicle>)
SubClassOf(<ns:#engine_part> <ns:#vehicle_part>)
SubClassOf(<ns:#drive_train_part> <ns:#vehicle_part>)
SubClassOf(<ns:#other_part> <ns:#vehicle_part>)
SubClassOf(<ns:#replaceable_part> <ns:#vehicle_part>)
SubClassOf(<ns:#replacing_part> <ns:#vehicle_part>)
EquivalentClasses(<ns:#vehicle> <ns:#vehicle_by_drive_train>)
Declaration(Class(<ns:#vehicle_by_drive_train>))
SubClassOf(<ns:#front_wheel_drive> <ns:#vehicle_by_drive_train>)
SubClassOf(<ns:#rear_wheel_drive> <ns:#vehicle_by_drive_train>)
SubClassOf(<ns:#four_wheel_drive> <ns:#vehicle_by_drive_train>)
SubClassOf(<ns:#all_wheel_drive> <ns:#vehicle_by_drive_train>)
DisjointUnion(<ns:#vehicle_by_drive_train> <ns:#front_wheel_drive> <ns:#rear_wheel_drive> <ns:#four_wheel_drive> <ns:#all_wheel_drive>)
AnnotationAssertion(<ns:#label_sbvr> <ns:#vehicle_by_drive_train> "vehicle_by_drive_train"@en)
AnnotationAssertion(<http://www.w3.org/2000/01/rdf-schema#label> <ns:#vehicle_by_drive_train> "vehicle by drive train"@en)
AnnotationAssertion(<ns:#label> <ns:#vehicle_by_drive_train> "vehicle by drive train")
SubClassOf(<ns:#vehicle> ObjectAllValuesFrom(<ns:#contains__engine> <ns:#diesel_engine>))
SubClassOf(<ns:#vehicle> ObjectSomeValuesFrom(<ns:#contains__engine> <ns:#engine>))

SubClassOf(<ns:#motrocycle> ObjectSomeValuesFrom(<ns:#contains__chain> <ns:#chain>))
FunctionalObjectProperty(<ns:#is_driven_by__professional_driver>)
SubClassOf(<ns:#lorry> ObjectExactCardinality(1 <ns:#contains__tachograph> <ns:#tachograph>))
SubClassOf(<ns:#high_utility_vehicle> ObjectMinCardinality(2 <ns:#has__axle_lock> <ns:#axle_lock>))
SubClassOf(<ns:#high_utility_vehicle> ObjectMaxCardinality(9 <ns:#has__axle_lock> <ns:#axle_lock>))
SubClassOf(<ns:#car> ObjectMinCardinality(1 <ns:#contains__seat> <ns:#seat>))
SubClassOf(<ns:#car> ObjectMaxCardinality(9 <ns:#contains__seat> <ns:#seat>))
DisjointClasses(<ns:#lorry> <ns:#motorcycle>)
)

9.3. priedas. Paskolų žodynas ir taisyklės

```
loan

loan_status_type
            Concept_type: categorization_type
            Necessity: is_for general_concept loan

Loans_by_status_type
            Necessity: categorization_scheme for general_concept loan that
subdivides loan by loan_status_type

loan_urgency_type
            Concept_type: categorization_type
            Necessity: is_for general_concept loan

Loans_by_loan_urgency_type
            Necessity: segmentation for general_concept loan that subdivides
loan by loan_urgency_type

reliable_loan
            General_concept: loan
valid_loan
            General_concept: loan

real_estate
            Synonym: asset
            Synonym: building

territory

person

bank

account

instant_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_loan_urgency_type

regular_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_loan_urgency_type
```

```
rejected_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_status_type

accepted_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_status_type

issued_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_status_type

returned_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_status_type

outstanding_loan
            General_concept: loan
            Necessity: is_included_in Loans_by_status_type

forest_parcel
            General_concept: territory

debtor
            Concept_type: verb_concept_role
            General_concept: person

bail
            Concept_type: verb_concept_role
            General_concept: real_estate

owner
            Concept_type: verb_concept_role
            General_concept: person

branch
            Concept_type: verb_concept_role
            General_concept: bank

parent_bank
            Concept_type: verb_concept_role
            General_concept: bank

amount
            General_concept: number
            Concept_type: role

duration
            General_concept: integer
            Concept_type: role

special_conditions
            General_concept: text

request_date
            General_concept: integer
            Concept_type: role

return_date
            General_concept: integer
            Concept_type: role
```

```
accept_date
           General_concept: integer
           Concept_type: role

address
           General_concept: text
           Concept_type: role

price
           General_concept: number
           Concept_type: role

person_code
           General_concept: text
           Concept_type: role

bank_code
           General_concept: text
           Concept_type: role

account_number
           General_concept: text

reject_date
           General_concept: integer
           Concept_type: role

reason
           General_concept: text
           Concept_type: role

area
           General_concept: number
           Concept_type: role

loan is_given_by bank
           Synonymous_form: bank gives loan

bank checks_validity_of loan

bank checks_reliability_of loan

loan is_got_by debtor

debtor request loan
debtor return loan

loan own bail
           Synonymous_form: bail is_owned_by loan

real_estate is_owned_by owner

person own account
           Synonymous_form: account is_owned_by person

account is_contained_in bank

loan is_requested
loan is_isued
loan is_returned
```

```
loan is_outstanding

forest_parcel is_park

real_estate is_bail

loan has amount
          Synonymous_form: amount of_the loan
          Concept_type: property_association
loan has request_date
          Concept_type: property_association
accepted_loan has accept_date
          Concept_type: property_association
rejected_loan has reject_date
          Concept_type: property_association
rejected_loan has reason
          Concept_type: property_association
regular_loan has duration
          Concept_type: property_association
instant_loan has special_conditions
          Concept_type: property_association
loan has return_date
          Concept_type: property_association
real_estate has address
          Concept_type: property_association
real_estate has price
          Concept_type: property_association
person has person_code
          Concept_type: property_association
bank has bank_code
          Concept_type: property_association

account has account_number
          Concept_type: property_association

territory has area
          Concept_type: property_association


It is necessary that loan is_got_by exactly 1 debtor

It is necessary that loan own exactly 1 bail

It is necessary that account is_contained_in exactly 1 bank

It is necessary that person own at_least 1 account

It is necessary that person own at_most 5 account

It is necessary that loan is_given_by exactly 1 bank

It is necessary that real_estate is_owned_by exactly 1 owner
```

9.4. priedas. Paskolų ontologija

Prefix(ns:=<http://isd.ktu.lt/semantika/>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)

Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)
Ontology(<http://isd.ktu.lt/semantika/s2o>
Declaration(Class(<ns:s2o#accepted_loan>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#accepted_loan> "accepted loan")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#accepted_loan> "accepted_loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#accepted_loan> "accepted loan"@en)
SubClassOf(<ns:s2o#accepted_loan> <ns:s2o#loans_by_status_type>)
Declaration(Class(<ns:s2o#account>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#account> "account"@en)
AnnotationAssertion(rdfs:label <ns:s2o#account> "account"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#account> "account")
SubClassOf(<ns:s2o#account> ObjectExactCardinality(1 <ns:s2o#is_contained_in__bank> <ns:s2o#bank>))
Declaration(Class(<ns:s2o#bail>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#bail> "bail"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#bail> "bail")
AnnotationAssertion(rdfs:label <ns:s2o#bail> "bail"@en)
SubClassOf(<ns:s2o#bail> <ns:s2o#role>)
Declaration(Class(<ns:s2o#bank>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#bank> "bank"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#bank> "bank")
AnnotationAssertion(rdfs:label <ns:s2o#bank> "bank"@en)
Declaration(Class(<ns:s2o#branch>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#branch> "branch")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#branch> "branch"@en)
AnnotationAssertion(rdfs:label <ns:s2o#branch> "branch"@en)
SubClassOf(<ns:s2o#branch> <ns:s2o#role>)
Declaration(Class(<ns:s2o#debtor>))
AnnotationAssertion(rdfs:label <ns:s2o#debtor> "debtor"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#debtor> "debtor"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#debtor> "debtor")
SubClassOf(<ns:s2o#debtor> <ns:s2o#role>)
Declaration(Class(<ns:s2o#forest_parcel>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#forest_parcel> "forest_parcel"@en)
AnnotationAssertion(rdfs:label <ns:s2o#forest_parcel> "forest parcel"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#forest_parcel> "forest parcel")
SubClassOf(<ns:s2o#forest_parcel> <ns:s2o#territory>)
Declaration(Class(<ns:s2o#instant_loan>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#instant_loan> "instant loan")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#instant_loan> "instant_loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#instant_loan> "instant loan"@en)
SubClassOf(<ns:s2o#instant_loan> <ns:s2o#loans_by_loan_urgency_type>)
Declaration(Class(<ns:s2o#issued_loan>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#issued_loan> "issued_loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#issued_loan> "issued loan")
AnnotationAssertion(rdfs:label <ns:s2o#issued_loan> "issued loan"@en)
SubClassOf(<ns:s2o#issued_loan> <ns:s2o#loans_by_status_type>)
Declaration(Class(<ns:s2o#loan>))

AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#loan> "loan")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#loan> "loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#loan> "loan"@en)
EquivalentClasses(<ns:s2o#loan> <ns:s2o#loans_by_loan_urgency_type>)
EquivalentClasses(<ns:s2o#loan> <ns:s2o#loans_by_status_type>)
SubClassOf(<ns:s2o#loan> ObjectExactCardinality(1 <ns:s2o#is_given_by__bank> <ns:s2o#bank>))
SubClassOf(<ns:s2o#loan> ObjectExactCardinality(1 <ns:s2o#is_got_by__debtor> <ns:s2o#person>))
SubClassOf(<ns:s2o#loan> ObjectExactCardinality(1 <ns:s2o#own__bail> <ns:s2o#real_estate>))
Declaration(Class(<ns:s2o#loans_by_loan_urgency_type>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#loans_by_loan_urgency_type> "Loans by loan urgency type")
AnnotationAssertion(rdfs:label <ns:s2o#loans_by_loan_urgency_type> "Loans by loan urgency type"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#loans_by_loan_urgency_type> "Loans_by_loan_urgency_type"@en)
EquivalentClasses(<ns:s2o#loans_by_loan_urgency_type> <ns:s2o#loan>)
DisjointUnion(<ns:s2o#loans_by_loan_urgency_type> <ns:s2o#regular_loan> <ns:s2o#instant_loan>)
Declaration(Class(<ns:s2o#loans_by_status_type>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#loans_by_status_type> "Loans by status type")
AnnotationAssertion(rdfs:label <ns:s2o#loans_by_status_type> "Loans by status type"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#loans_by_status_type> "Loans_by_status_type"@en)
EquivalentClasses(<ns:s2o#loans_by_status_type> <ns:s2o#loan>)
Declaration(Class(<ns:s2o#outstanding_loan>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#outstanding_loan> "outstanding_loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#outstanding_loan> "outstanding loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#outstanding_loan> "outstanding loan")
SubClassOf(<ns:s2o#outstanding_loan> <ns:s2o#loans_by_status_type>)
Declaration(Class(<ns:s2o#owner>))
AnnotationAssertion(rdfs:label <ns:s2o#owner> "owner"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#owner> "owner")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#owner> "owner"@en)
SubClassOf(<ns:s2o#owner> <ns:s2o#role>)
Declaration(Class(<ns:s2o#parent_bank>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#parent_bank> "parent_bank"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#parent_bank> "parent bank")
AnnotationAssertion(rdfs:label <ns:s2o#parent_bank> "parent bank"@en)
SubClassOf(<ns:s2o#parent_bank> <ns:s2o#role>)
Declaration(Class(<ns:s2o#person>))
AnnotationAssertion(rdfs:label <ns:s2o#person> "person"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#person> "person"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#person> "person")
SubClassOf(<ns:s2o#person> ObjectSomeValuesFrom(<ns:s2o#own__account> <ns:s2o#account>))
SubClassOf(<ns:s2o#person> ObjectMaxCardinality(5 <ns:s2o#own__account> <ns:s2o#account>))

Declaration(Class(<ns:s2o#real_estate>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#real_estate> "real_estate"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#real_estate> "real estate")
AnnotationAssertion(rdfs:label <ns:s2o#real_estate> "real estate"@en)
SubClassOf(<ns:s2o#real_estate> ObjectExactCardinality(1 <ns:s2o#is_owned_by__owner> <ns:s2o#person>))
Declaration(Class(<ns:s2o#regular_loan>))
AnnotationAssertion(rdfs:label <ns:s2o#regular_loan> "regular loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#regular_loan> "regular loan")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#regular_loan> "regular_loan"@en)
SubClassOf(<ns:s2o#regular_loan> <ns:s2o#loans_by_loan_urgency_type>)
Declaration(Class(<ns:s2o#rejected_loan>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#rejected_loan> "rejected loan")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#rejected_loan> "rejected_loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#rejected_loan> "rejected loan"@en)
SubClassOf(<ns:s2o#rejected_loan> <ns:s2o#loans_by_status_type>)
Declaration(Class(<ns:s2o#reliable_loan>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#reliable_loan> "reliable_loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#reliable_loan> "reliable loan")
AnnotationAssertion(rdfs:label <ns:s2o#reliable_loan> "reliable loan"@en)
SubClassOf(<ns:s2o#reliable_loan> <ns:s2o#loan>)
Declaration(Class(<ns:s2o#returned_loan>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#returned_loan> "returned loan")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#returned_loan> "returned_loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#returned_loan> "returned loan"@en)
SubClassOf(<ns:s2o#returned_loan> <ns:s2o#loans_by_status_type>)
Declaration(Class(<ns:s2o#role>))
AnnotationAssertion(rdfs:label <ns:s2o#role> "role"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#role> "role"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#role> "role")
Declaration(Class(<ns:s2o#territory>))
AnnotationAssertion(rdfs:label <ns:s2o#territory> "territory"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#territory> "territory"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#territory> "territory")
Declaration(Class(<ns:s2o#valid_loan>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#valid_loan> "valid_loan"@en)
AnnotationAssertion(rdfs:label <ns:s2o#valid_loan> "valid loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#valid_loan> "valid loan")
SubClassOf(<ns:s2o#valid_loan> <ns:s2o#loan>)
Declaration(ObjectProperty(<ns:s2o#checks_reliability_of__loan>))
AnnotationAssertion(rdfs:label <ns:s2o#checks_reliability_of__loan> "bank checks reliability of loan"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#checks_reliability_of__loan> "bank checks_reliability_of loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#checks_reliability_of__loan> "bank checks reliability of loan")
ObjectPropertyDomain(<ns:s2o#checks_reliability_of__loan> <ns:s2o#bank>)
ObjectPropertyRange(<ns:s2o#checks_reliability_of__loan> <ns:s2o#loan>)
Declaration(ObjectProperty(<ns:s2o#checks_validity_of__loan>))

AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#checks_validity_of__loan> "bank checks_validity_of loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#checks_validity_of__loan> "bank checks validity of loan")
AnnotationAssertion(rdfs:label <ns:s2o#checks_validity_of__loan> "bank checks validity of loan"@en)
ObjectPropertyDomain(<ns:s2o#checks_validity_of__loan> <ns:s2o#bank>)
ObjectPropertyRange(<ns:s2o#checks_validity_of__loan> <ns:s2o#loan>)
Declaration(ObjectProperty(<ns:s2o#is_contained_in__bank>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_contained_in__bank> "account is_contained_in bank"@en)
AnnotationAssertion(rdfs:label <ns:s2o#is_contained_in__bank> "account is contained in bank"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_contained_in__bank> "account is contained in bank")
ObjectPropertyDomain(<ns:s2o#is_contained_in__bank> <ns:s2o#account>)
ObjectPropertyRange(<ns:s2o#is_contained_in__bank> <ns:s2o#bank>)
Declaration(ObjectProperty(<ns:s2o#is_given_by__bank>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_given_by__bank> "loan is_given_by bank"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_given_by__bank> "loan is given by bank")
AnnotationAssertion(rdfs:label <ns:s2o#is_given_by__bank> "loan is given by bank"@en)
ObjectPropertyDomain(<ns:s2o#is_given_by__bank> <ns:s2o#loan>)
ObjectPropertyRange(<ns:s2o#is_given_by__bank> <ns:s2o#bank>)
Declaration(ObjectProperty(<ns:s2o#is_got_by__debtor>))
AnnotationAssertion(rdfs:label <ns:s2o#is_got_by__debtor> "loan is got by debtor"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_got_by__debtor> "loan is_got_by debtor"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_got_by__debtor> "loan is got by debtor")
ObjectPropertyDomain(<ns:s2o#is_got_by__debtor> <ns:s2o#loan>)
ObjectPropertyRange(<ns:s2o#is_got_by__debtor> <ns:s2o#person>)
Declaration(ObjectProperty(<ns:s2o#is_owned_by__owner>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_owned_by__owner> "real estate is owned by owner")
AnnotationAssertion(rdfs:label <ns:s2o#is_owned_by__owner> "real estate is owned by owner"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_owned_by__owner> "real_estate is_owned_by owner"@en)
ObjectPropertyDomain(<ns:s2o#is_owned_by__owner> <ns:s2o#real_estate>)
ObjectPropertyRange(<ns:s2o#is_owned_by__owner> <ns:s2o#person>)
Declaration(ObjectProperty(<ns:s2o#own__account>))
AnnotationAssertion(rdfs:label <ns:s2o#own__account> "person own account"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#own__account> "person own account")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#own__account> "person own account"@en)
ObjectPropertyDomain(<ns:s2o#own__account> <ns:s2o#person>)
ObjectPropertyRange(<ns:s2o#own__account> <ns:s2o#account>)
Declaration(ObjectProperty(<ns:s2o#own__bail>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#own__bail> "loan own bail"@en)
AnnotationAssertion(rdfs:label <ns:s2o#own__bail> "loan own bail"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#own__bail> "loan own bail")

ObjectPropertyDomain(<ns:s2o#own__bail> <ns:s2o#loan>)
ObjectPropertyRange(<ns:s2o#own__bail> <ns:s2o#real_estate>)
Declaration(ObjectProperty(<ns:s2o#request__loan>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#request__loan> "debtor request loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#request__loan> "debtor request loan")
AnnotationAssertion(rdfs:label <ns:s2o#request__loan> "debtor request loan"@en)
ObjectPropertyDomain(<ns:s2o#request__loan> <ns:s2o#person>)
ObjectPropertyRange(<ns:s2o#request__loan> <ns:s2o#loan>)
Declaration(ObjectProperty(<ns:s2o#return__loan>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#return__loan> "debtor return loan"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#return__loan> "debtor return loan")
AnnotationAssertion(rdfs:label <ns:s2o#return__loan> "debtor return loan"@en)
ObjectPropertyDomain(<ns:s2o#return__loan> <ns:s2o#person>)
ObjectPropertyRange(<ns:s2o#return__loan> <ns:s2o#loan>)
Declaration(DataProperty(<ns:s2o#accept_date>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#accept_date> "accept_date"@en)
AnnotationAssertion(rdfs:label <ns:s2o#accept_date> "accept date"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#accept_date> "accept date")
DataPropertyDomain(<ns:s2o#accept_date> <ns:s2o#accepted_loan>)
DataPropertyRange(<ns:s2o#accept_date> xsd:integer)
Declaration(DataProperty(<ns:s2o#account_number>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#account_number> "account_number"@en)
AnnotationAssertion(rdfs:label <ns:s2o#account_number> "account number"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#account_number> "account number")
DataPropertyDomain(<ns:s2o#account_number> <ns:s2o#account>)
DataPropertyRange(<ns:s2o#account_number> xsd:string)
Declaration(DataProperty(<ns:s2o#address>))
AnnotationAssertion(rdfs:label <ns:s2o#address> "address"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#address> "address"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#address> "address")
DataPropertyDomain(<ns:s2o#address> <ns:s2o#real_estate>)
DataPropertyRange(<ns:s2o#address> xsd:string)
Declaration(DataProperty(<ns:s2o#amount>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#amount> "amount"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#amount> "amount")
AnnotationAssertion(rdfs:label <ns:s2o#amount> "amount"@en)
DataPropertyDomain(<ns:s2o#amount> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#amount> xsd:decimal)
Declaration(DataProperty(<ns:s2o#area>))
AnnotationAssertion(rdfs:label <ns:s2o#area> "area"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#area> "area"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#area> "area")
DataPropertyDomain(<ns:s2o#area> <ns:s2o#territory>)
DataPropertyRange(<ns:s2o#area> xsd:decimal)
Declaration(DataProperty(<ns:s2o#bank_code>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#bank_code> "bank code")
AnnotationAssertion(rdfs:label <ns:s2o#bank_code> "bank code"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#bank_code> "bank_code"@en)
DataPropertyDomain(<ns:s2o#bank_code> <ns:s2o#bank>)
DataPropertyRange(<ns:s2o#bank_code> xsd:string)

Declaration(DataProperty(<ns:s2o#duration>))
AnnotationAssertion(rdfs:label <ns:s2o#duration> "duration"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#duration> "duration")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#duration> "duration"@en)
DataPropertyDomain(<ns:s2o#duration> <ns:s2o#regular_loan>)
DataPropertyRange(<ns:s2o#duration> xsd:integer)
Declaration(DataProperty(<ns:s2o#is_bail>))
AnnotationAssertion(rdfs:label <ns:s2o#is_bail> "is bail"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_bail> "is_bail"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_bail> "is bail")
DataPropertyDomain(<ns:s2o#is_bail> <ns:s2o#real_estate>)
DataPropertyRange(<ns:s2o#is_bail> xsd:boolean)
Declaration(DataProperty(<ns:s2o#is_isued>))
AnnotationAssertion(rdfs:label <ns:s2o#is_isued> "is isued"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_isued> "is_isued"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_isued> "is isued")
DataPropertyDomain(<ns:s2o#is_isued> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#is_isued> xsd:boolean)
Declaration(DataProperty(<ns:s2o#is_outstanding>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_outstanding> "is_outstanding"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_outstanding> "is outstanding")
AnnotationAssertion(rdfs:label <ns:s2o#is_outstanding> "is outstanding"@en)
DataPropertyDomain(<ns:s2o#is_outstanding> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#is_outstanding> xsd:boolean)
Declaration(DataProperty(<ns:s2o#is_park>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_park> "is park")
AnnotationAssertion(rdfs:label <ns:s2o#is_park> "is park"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_park> "is_park"@en)
DataPropertyDomain(<ns:s2o#is_park> <ns:s2o#forest_parcel>)
DataPropertyRange(<ns:s2o#is_park> xsd:boolean)
Declaration(DataProperty(<ns:s2o#is_requested>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_requested> "is_requested"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_requested> "is requested")
AnnotationAssertion(rdfs:label <ns:s2o#is_requested> "is requested"@en)
DataPropertyDomain(<ns:s2o#is_requested> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#is_requested> xsd:boolean)
Declaration(DataProperty(<ns:s2o#is_returned>))
AnnotationAssertion(rdfs:label <ns:s2o#is_returned> "is returned"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#is_returned> "is_returned"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#is_returned> "is returned")
DataPropertyDomain(<ns:s2o#is_returned> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#is_returned> xsd:boolean)
Declaration(DataProperty(<ns:s2o#person_code>))
AnnotationAssertion(rdfs:label <ns:s2o#person_code> "person code"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#person_code> "person_code"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#person_code> "person code")
DataPropertyDomain(<ns:s2o#person_code> <ns:s2o#person>)
DataPropertyRange(<ns:s2o#person_code> xsd:string)
Declaration(DataProperty(<ns:s2o#price>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#price> "price")

AnnotationAssertion(rdfs:label <ns:s2o#price> "price"@en)
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#price> "price"@en)
DataPropertyDomain(<ns:s2o#price> <ns:s2o#real_estate>)
DataPropertyRange(<ns:s2o#price> xsd:decimal)
Declaration(DataProperty(<ns:s2o#reason>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#reason> "reason"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#reason> "reason")
AnnotationAssertion(rdfs:label <ns:s2o#reason> "reason"@en)
DataPropertyDomain(<ns:s2o#reason> <ns:s2o#rejected_loan>)
DataPropertyRange(<ns:s2o#reason> xsd:string)
Declaration(DataProperty(<ns:s2o#reject_date>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#reject_date> "reject_date"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#reject_date> "reject date")
AnnotationAssertion(rdfs:label <ns:s2o#reject_date> "reject date"@en)
DataPropertyDomain(<ns:s2o#reject_date> <ns:s2o#rejected_loan>)
DataPropertyRange(<ns:s2o#reject_date> xsd:integer)
Declaration(DataProperty(<ns:s2o#request_date>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#request_date> "request_date"@en)
AnnotationAssertion(rdfs:label <ns:s2o#request_date> "request date"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#request_date> "request date")
DataPropertyDomain(<ns:s2o#request_date> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#request_date> xsd:integer)
Declaration(DataProperty(<ns:s2o#return_date>))
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#return_date> "return date")
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#return_date> "return_date"@en)
AnnotationAssertion(rdfs:label <ns:s2o#return_date> "return date"@en)
DataPropertyDomain(<ns:s2o#return_date> <ns:s2o#loan>)
DataPropertyRange(<ns:s2o#return_date> xsd:integer)
Declaration(DataProperty(<ns:s2o#special_conditions>))
AnnotationAssertion(<ns:s2o#label_sbvr> <ns:s2o#special_conditions> "special_conditions"@en)
AnnotationAssertion(rdfs:label <ns:s2o#special_conditions> "special conditions"@en)
AnnotationAssertion(<ns:s2o#label_en> <ns:s2o#special_conditions> "special conditions")
DataPropertyDomain(<ns:s2o#special_conditions> <ns:s2o#instant_loan>)
DataPropertyRange(<ns:s2o#special_conditions> xsd:string)
Declaration(AnnotationProperty(<ns:s2o#label_en>))
Declaration(AnnotationProperty(<ns:s2o#label_sbvr>))
)

9.5. priedas. Testavimo duomenų pavyzdiniai failai

Pavyzdinis testinis failas OWL.xmi:

```
<xmi:XMI xmi:version="2.0">
<owl2:Ontology axioms="/4 /7 /11 /26 /27 /32 /14 /35 /36 /41 /16 /18 /19 /44 /45 /50 /20 /22" ontologyIRI="/1" versionIRI="/2"/>
<owl2:IRI lexicalValue="http://isd.ktu.lt/semantika/s2o"/>
<owl2:IRI lexicalValue="http://isd.ktu.lt/semantika"/>
<owl2:AnnotationProperty entityIRI="/5" declaration="/4"/>
<owl2:Declaration ontology="/0" entity="/3"/>
<owl2:IRI lexicalValue="ns:s2o#label_sbvr"/>
<owl2:AnnotationProperty entityIRI="/8" declaration="/7"/>
```

```
<owl2:Declaration ontology="/0" entity="/6"/>
<owl2:IRI lexicalValue="ns:s2o#label_en"/>
<owl2:Class entityIRI="/10" declaration="/11"/>
<owl2:IRI lexicalValue="ns:s2o#photo_camera"/>
<owl2:Declaration ontology="/0" entity="/9"/>
<owl2:Class entityIRI="/13" declaration="/14"/>
<owl2:IRI lexicalValue="ns:s2o#white_balance_preset"/>
<owl2:Declaration ontology="/0" entity="/12"/>
<owl2:ObjectProperty entityIRI="/17" declaration="/16" domains="/18" ranges="/19"/>
<owl2:Declaration ontology="/0" entity="/15"/>
<owl2:IRI lexicalValue="ns:s2o#has__white_balance_preset"/>
<owl2:ObjectPropertyDomain ontology="/0" objectPropertyExpression="/15" domain="/9"/>
<owl2:ObjectPropertyRange ontology="/0" objectPropertyExpression="/15" range="/12"/>
<owl2:SubClassOf ontology="/0" subClassExpression="/9" superClassExpression="/21"/>
<owl2:ObjectMinCardinality cardinality="2" classExpression="/12" objectPropertyExpression="/15"/>
<owl2:SubClassOf ontology="/0" subClassExpression="/9" superClassExpression="/23"/>
<owl2:ObjectMaxCardinality cardinality="9" classExpression="/12" objectPropertyExpression="/15"/>
<owl2:AnnotationProperty entityIRI="/25"/>
<owl2:IRI lexicalValue="ns:s2o#label_sbvr"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/24" annotationSubject="/9" annotationValue="photo_camera" annotationLanguage="@en"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/28" annotationSubject="/9" annotationValue="photo camera" annotationLanguage="@en"/>
<owl2:AnnotationProperty entityIRI="/29"/>
<owl2:IRI lexicalValue="http://www.w3.org/2000/01/rdf-schema#label"/>
<owl2:AnnotationProperty entityIRI="/31"/>
<owl2:IRI lexicalValue="ns:s2o#label_en"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/30" annotationSubject="/9" annotationValue="photo camera"/>
<owl2:AnnotationProperty entityIRI="/34"/>
<owl2:IRI lexicalValue="ns:s2o#label_sbvr"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/33" annotationSubject="/12" annotationValue="white_balance_preset"
annotationLanguage="@en"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/37" annotationSubject="/12" annotationValue="white balance preset"
annotationLanguage="@en"/>
<owl2:AnnotationProperty entityIRI="/38"/>
<owl2:IRI lexicalValue="http://www.w3.org/2000/01/rdf-schema#label"/>
<owl2:AnnotationProperty entityIRI="/40"/>
<owl2:IRI lexicalValue="ns:s2o#label_en"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/39" annotationSubject="/12" annotationValue="white balance preset"/>
<owl2:AnnotationProperty entityIRI="/43"/>
<owl2:IRI lexicalValue="ns:s2o#label_sbvr"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/42" annotationSubject="/15" annotationValue="photo_camera has white_balance_preset"
annotationLanguage="@en"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/46" annotationSubject="/15" annotationValue="photo camera has white balance preset"
annotationLanguage="@en"/>
<owl2:AnnotationProperty entityIRI="/47"/>
<owl2:IRI lexicalValue="http://www.w3.org/2000/01/rdf-schema#label"/>
<owl2:AnnotationProperty entityIRI="/49"/>
<owl2:IRI lexicalValue="ns:s2o#label_en"/>
<owl2:AnnotationAssertion ontology="/0" annotationProperty="/48" annotationSubject="/15" annotationValue="photo camera has white balance preset"/>
</xmi:XMI>
```

Atspindintis ontologiją:

```
Prefix( xsd:=<http://www.w3.org/2001/XMLSchema#> )
Prefix( ns:=<http://isd.ktu.lt/semantika/> )
Ontology( <http://isd.ktu.lt/semantika/s2o>
Declaration( AnnotationProperty( <ns:s2o#label_sbvr> ) )
Declaration( AnnotationProperty( <ns:s2o#label_en> ) )
Declaration( Class( <ns:s2o#photo_camera> ) )
AnnotationAssertion( <ns:s2o#label_sbvr> <ns:s2o#photo_camera> "photo_camera"@en )
AnnotationAssertion( <http://www.w3.org/2000/01/rdf-schema#label> <ns:s2o#photo_camera> "photo camera"@en )
AnnotationAssertion( <ns:s2o#label_en> <ns:s2o#photo_camera> "photo camera" )
Declaration( Class( <ns:s2o#white_balance_preset> ) )
AnnotationAssertion( <ns:s2o#label_sbvr> <ns:s2o#white_balance_preset> "white_balance_preset"@en )
```

```
AnnotationAssertion( <http://www.w3.org/2000/01/rdf-schema#label> <ns:s2o#white_balance_preset> "white balance preset"@en )
AnnotationAssertion( <ns:s2o#label_en> <ns:s2o#white_balance_preset> "white balance preset" )
Declaration( ObjectProperty( <ns:s2o#has__white_balance_preset> ) )
ObjectPropertyDomain( <ns:s2o#has__white_balance_preset> <ns:s2o#photo_camera> )
ObjectPropertyRange( <ns:s2o#has__white_balance_preset> <ns:s2o#white_balance_preset> )
AnnotationAssertion( <ns:s2o#label_sbvr> <ns:s2o#has__white_balance_preset> "photo_camera has white_balance_preset"@en )
AnnotationAssertion( <http://www.w3.org/2000/01/rdf-schema#label> <ns:s2o#has__white_balance_preset> "photo camera has white balance preset"@en )
AnnotationAssertion( <ns:s2o#label_en> <ns:s2o#has__white_balance_preset> "photo camera has white balance preset" )
SubClassOf( <ns:s2o#photo_camera> ObjectMinCardinality( 2 <ns:s2o#has__white_balance_preset> <ns:s2o#white_balance_preset> ) )
SubClassOf( <ns:s2o#photo_camera> ObjectMaxCardinality( 9 <ns:s2o#has__white_balance_preset> <ns:s2o#white_balance_preset> ) )
)
```

Eksperimentinio tyrimo IS gautas rezultatas:

```
<xmi:Fields xmlns:xmi="yourNamespaceHere" xmlns:owl2="otherNamespace" xmlns:SBVR="otherNamespace">
 <AtomicFormulation id="_XWmfCdTPEeWnj45K7EEoEw" factType="_XWme89TPEeWnj45K7EEoEw" />
 <ConceptualSchema id="_XWme8NTPEeWnj45K7EEoEw" xmlns:p4="http://www.omg.org/spec/XMI/20110701"
xmlns="http://www.eclipse.org/sbvr/1.0.0/SBVR">
  <concept type="SBVR:ObjectType" id="_XWme8dTPEeWnj45K7EEoEw" />
  <concept type="SBVR:ObjectType" id="_XWme8tTPEeWnj45K7EEoEw" />
  <concept type="SBVR:AssociativeFactType" id="_XWme89TPEeWnj45K7EEoEw">
   <role id="_XWme9NTPEeWnj45K7EEoEw" objectType="_XWme8dTPEeWnj45K7EEoEw" />
   <role id="_XWme9dTPEeWnj45K7EEoEw" objectType="_XWme8tTPEeWnj45K7EEoEw" />
  </concept>
 </ConceptualSchema>
 <Designation id="_XWme-NTPEeWnj45K7EEoEw" meaning="_XWme8dTPEeWnj45K7EEoEw" expression="_XWme9tTPEeWnj45K7EEoEw" />
 <slas id="_XWme-9TPEeWnj45K7EEoEw" meaning="_XWme8tTPEeWnj45K7EEoEw" expression="_XWme-dTPEeWnj45K7EEoEw" />
 <Designation id="_XWme_dTPEeWnj45K7EEoEw" meaning="_XWme89TPEeWnj45K7EEoEw" expression="_XWme_NTPEeWnj45K7EEoEw" />
 <Designation id="_XWme_tTPEeWnj45K7EEoEw" meaning="_XWme9NTPEeWnj45K7EEoEw" expression="_XWme9tTPEeWnj45K7EEoEw" />
 <Designation id="_XWmfANTPEeWnj45K7EEoEw" meaning="_XWme9dTPEeWnj45K7EEoEw" expression="_XWme-dTPEeWnj45K7EEoEw" />
 <FactSymbol id="_XWmfCNTPEeWnj45K7EEoEw" meaning="_XWme89TPEeWnj45K7EEoEw" expression="_XWme_NTPEeWnj45K7EEoEw"
preferred="true" />
 <NecessityFormulation id="_XWmfE9TPEeWnj45K7EEoEw" proposition="_XWmfEtTPEeWnj45K7EEoEw"
logicalFormulation="_XWmfEdTPEeWnj45K7EEoEw" />
 <NonNegativeInteger id="_XWmfCtTPEeWnj45K7EEoEw" value="2" />
 <NonNegativeInteger id="_XWmfC9TPEeWnj45K7EEoEw" value="9" />
 <NumericRangeQuantification id="_XWmfENTPEeWnj45K7EEoEw" variable="_XWmfDtTPEeWnj45K7EEoEw"
scopeFormulation="_XWmfCdTPEeWnj45K7EEoEw" minimumCardinality="_XWmfCtTPEeWnj45K7EEoEw"
maximumCardinality="_XWmfC9TPEeWnj45K7EEoEw" />
 <Placeholder id="_XWmfBdTPEeWnj45K7EEoEw" meaning="_XWme9NTPEeWnj45K7EEoEw" expression="_XWme9tTPEeWnj45K7EEoEw"
isAtStartingCharacterPosition="_XWmfAtTPEeWnj45K7EEoEw" />
 <Placeholder id="_XWmfBtTPEeWnj45K7EEoEw" meaning="_XWme9dTPEeWnj45K7EEoEw" expression="_XWme-dTPEeWnj45K7EEoEw"
isAtStartingCharacterPosition="_XWmfB9TPEeWnj45K7EEoEw" />
 <PositiveInteger id="_XWmfAtTPEeWnj45K7EEoEw" value="1" />
 <PositiveInteger id="_XWmfB9TPEeWnj45K7EEoEw" value="18" />
 <Proposition id="_XWmfEtTPEeWnj45K7EEoEw" />
 <RoleBinding id="_XWmfDdTPEeWnj45K7EEoEw" atomicFormulation="_XWmfCdTPEeWnj45K7EEoEw"
bindableTarget="_XWmfDNTPEeWnj45K7EEoEw" />
 <RoleBinding id="_XWmfD9TPEeWnj45K7EEoEw" atomicFormulation="_XWmfCdTPEeWnj45K7EEoEw"
bindableTarget="_XWmfDtTPEeWnj45K7EEoEw" />
 <SententialForm id="_XWmfA9TPEeWnj45K7EEoEw" meaning="_XWme89TPEeWnj45K7EEoEw" expression="_XWmfBNTPEeWnj45K7EEoEw"
designation="_XWmfBdTPEeWnj45K7EEoEw _XWmfBtTPEeWnj45K7EEoEw" />
 <Statement id="_XWmfFdTPEeWnj45K7EEoEw" meaning="_XWmfEtTPEeWnj45K7EEoEw" expression="_XWmfFNTPEeWnj45K7EEoEw" />
 <Term id="_XWme99TPEeWnj45K7EEoEw" meaning="_XWme8dTPEeWnj45K7EEoEw" expression="_XWme9tTPEeWnj45K7EEoEw" preferred="true" />
 <Term id="_XWme-tTPEeWnj45K7EEoEw" meaning="_XWme8tTPEeWnj45K7EEoEw" expression="_XWme-dTPEeWnj45K7EEoEw" preferred="true" />
 <Term id="_XWme_9TPEeWnj45K7EEoEw" meaning="_XWme9NTPEeWnj45K7EEoEw" expression="_XWme9tTPEeWnj45K7EEoEw" />
 <Term id="_XWmfAdTPEeWnj45K7EEoEw" meaning="_XWme9dTPEeWnj45K7EEoEw" expression="_XWme-dTPEeWnj45K7EEoEw" />
 <Text id="_XWme9tTPEeWnj45K7EEoEw" value="photo_camera" />
 <Text id="_XWme-dTPEeWnj45K7EEoEw" value="white_balance_preset" />
 <Text id="_XWme_NTPEeWnj45K7EEoEw" value="has" />
 <Text id="_XWmfBNTPEeWnj45K7EEoEw" value="photo_camera has white_balance_preset" />
 <Text id="_XWmfFNTPEeWnj45K7EEoEw" />
```

```
  <UniversalQuantification id="_XWmfEdTPEeWnj45K7EEoEw" variable="_XWmfDNTPEeWnj45K7EEoEw"
scopeFormulation="_XWmfENTPEeWnj45K7EEoEw" />
  <Variable id="_XWmfDNTPEeWnj45K7EEoEw" rangedOverConcept="_XWme8dTPEeWnj45K7EEoEw" />
  <Variable id="_XWmfDtTPEeWnj45K7EEoEw" rangedOverConcept="_XWme8tTPEeWnj45K7EEoEw" />
</xmi:Fields>
```

Gautas S2O OWL xmi rezultatas:

```
<xmi:Fields xmlns:xmi="yourNamespaceHere" xmlns:owl2="otherNamespace" xmlns:SBVR="otherNamespace">
  <AtomicFormulation id="_zhiZ4tTGEeWy8dwhQm454g" factType="_zhiZwdTGEeWy8dwhQm454g" />
  <ConceptualSchema id="_zhCDcdTGEeWy8dwhQm454g">
    <concept type="SBVR:ObjectType" id="_zhhysNTGEeWy8dwhQm454g" />
    <concept type="SBVR:ObjectType" id="_zhiZwNTGEeWy8dwhQm454g" />
    <concept type="SBVR:AssociativeFactType" id="_zhiZwdTGEeWy8dwhQm454g">
      <role id="_zhiZwtTGEeWy8dwhQm454g" objectType="_zhhysNTGEeWy8dwhQm454g" />
      <role id="_zhiZw9TGEeWy8dwhQm454g" objectType="_zhiZwNTGEeWy8dwhQm454g" />
    </concept>
  </ConceptualSchema>
  <Designation id="_zhiZ0NTGEeWy8dwhQm454g" meaning="_zhhysNTGEeWy8dwhQm454g" expression="_zhiZz9TGEeWy8dwhQm454g" />
  <Designation id="_zhiZ09TGEeWy8dwhQm454g" meaning="_zhiZwNTGEeWy8dwhQm454g" expression="_zhiZ0tTGEeWy8dwhQm454g" />
  <Designation id="_zhiZ1dTGEeWy8dwhQm454g" meaning="_zhiZwtTGEeWy8dwhQm454g" expression="_zhiZz9TGEeWy8dwhQm454g" />
  <Designation id="_zhiZ19TGEeWy8dwhQm454g" meaning="_zhiZw9TGEeWy8dwhQm454g" expression="_zhiZ0tTGEeWy8dwhQm454g" />
  <Designation id="_zhiZ2dTGEeWy8dwhQm454g" meaning="_zhiZwdTGEeWy8dwhQm454g" expression="_zhiZ3NTGEeWy8dwhQm454g" />
  <FactSymbol id="_zhiZ4dTGEeWy8dwhQm454g" meaning="_zhiZwdTGEeWy8dwhQm454g" expression="_zhiZ3NTGEeWy8dwhQm454g" preferred="true"
/>
  <NecessityFormulation id="_zhjA0NTGEeWy8dwhQm454g" proposition="_zhjA0tTGEeWy8dwhQm454g"
logicalFormulation="_zhjA0dTGEeWy8dwhQm454g" />
  <NonNegativeInteger id="_zhiZ5tTGEeWy8dwhQm454g" value="2" />
  <NonNegativeInteger id="_zhiZ59TGEeWy8dwhQm454g" value="9" />
  <NumericRangeQuantification id="_zhiZ5dTGEeWy8dwhQm454g" variable="_zhiZ6dTGEeWy8dwhQm454g"
scopeFormulation="_zhiZ4tTGEeWy8dwhQm454g" minimumCardinality="_zhiZ5tTGEeWy8dwhQm454g"
maximumCardinality="_zhiZ59TGEeWy8dwhQm454g" />
  <Placeholder id="_zhiZ3dTGEeWy8dwhQm454g" meaning="_zhiZwtTGEeWy8dwhQm454g" expression="_zhiZz9TGEeWy8dwhQm454g"
isAtStartingCharacterPosition="_zhiZ29TGEeWy8dwhQm454g" />
  <Placeholder id="_zhiZ4NTGEeWy8dwhQm454g" meaning="_zhiZw9TGEeWy8dwhQm454g" expression="_zhiZ0tTGEeWy8dwhQm454g"
isAtStartingCharacterPosition="_zhiZ3tTGEeWy8dwhQm454g" />
  <PositiveInteger id="_zhiZ29TGEeWy8dwhQm454g" value="1" />
  <PositiveInteger id="_zhiZ3tTGEeWy8dwhQm454g" value="18" />
  <Proposition id="_zhjA0tTGEeWy8dwhQm454g" />
  <RoleBinding id="_zhiZ49TGEeWy8dwhQm454g" atomicFormulation="_zhiZ4tTGEeWy8dwhQm454g" bindableTarget="_zhiZ6NTGEeWy8dwhQm454g" />
  <RoleBinding id="_zhiZ5NTGEeWy8dwhQm454g" atomicFormulation="_zhiZ4tTGEeWy8dwhQm454g" bindableTarget="_zhiZ6dTGEeWy8dwhQm454g" />
  <SententialForm id="_zhiZ2tTGEeWy8dwhQm454g" meaning="_zhiZwdTGEeWy8dwhQm454g" expression="_zhiZ39TGEeWy8dwhQm454g"
designation="_zhiZ3dTGEeWy8dwhQm454g _zhiZ4NTGEeWy8dwhQm454g" />
  <Statement id="_zhjA09TGEeWy8dwhQm454g" meaning="_zhjA0tTGEeWy8dwhQm454g" expression="_zhjA1NTGEeWy8dwhQm454g" />
  <Term id="_zhiZ0dTGEeWy8dwhQm454g" meaning="_zhhysNTGEeWy8dwhQm454g" expression="_zhiZz9TGEeWy8dwhQm454g" preferred="true" />
  <Term id="_zhiZ1NTGEeWy8dwhQm454g" meaning="_zhiZwNTGEeWy8dwhQm454g" expression="_zhiZ0tTGEeWy8dwhQm454g" preferred="true" />
  <Term id="_zhiZ1tTGEeWy8dwhQm454g" meaning="_zhiZwtTGEeWy8dwhQm454g" expression="_zhiZz9TGEeWy8dwhQm454g" />
  <Term id="_zhiZ2NTGEeWy8dwhQm454g" meaning="_zhiZw9TGEeWy8dwhQm454g" expression="_zhiZ0tTGEeWy8dwhQm454g" />
  <Text id="_zhiZz9TGEeWy8dwhQm454g" value="photo_camera" />
  <Text id="_zhiZ0tTGEeWy8dwhQm454g" value="white_balance_preset" />
  <Text id="_zhiZ3NTGEeWy8dwhQm454g" value="has" />
  <Text id="_zhiZ39TGEeWy8dwhQm454g" value="photo_camera has white_balance_preset" />
  <Text id="_zhjA1NTGEeWy8dwhQm454g" value="" />
  <UniversalQuantification id="_zhjA0dTGEeWy8dwhQm454g" variable="_zhiZ6NTGEeWy8dwhQm454g"
scopeFormulation="_zhiZ5dTGEeWy8dwhQm454g" />
  <Variable id="_zhiZ6NTGEeWy8dwhQm454g" rangedOverConcept="_zhhysNTGEeWy8dwhQm454g" />
  <Variable id="_zhiZ6dTGEeWy8dwhQm454g" rangedOverConcept="_zhiZwNTGEeWy8dwhQm454g" />
</xmi:Fields>
```

9.6. priedas. Foto įrangos veiklos žodynas ir taisyklės

Photo_equipment
    General_concept: vocabulary
    Language: "en"
    Namespace_URI: "http://isd.ktu.lt/Photo_equipment"

product

product contains product
    Concept_type: partitive_verb_concept
    Concept_type: transitive_verb_concept

product is_replaceable_by product
    Concept_type: purely_reflexive_verb_concept

replacing_product is_replacing_product_for product
    Concept_type: purely_reflexive_verb_concept

replaceable_product
    Definition: product that is_replaceable_by product
    Concept_type: verb_concept_role
    General_concept: product
replacing_product
    Definition: product that is_replacing_product_for product
    Concept_type: verb_concept_role
    General_concept: product

electronics_product
    General_concept: product
creative_product
    General_concept: product
other_product
    General_concept: product

photo_camera
    General_concept: electronics_product
digital_photo_camera
    General_concept: photo_camera
film_photo_camera
    General_concept: photo_camera

camera_model

photo_camera has camera_model
camera_model is_smart
camera_model is_fast
camera_model is_professional
concept `camera_model` incorporates characteristic `camera_model is_smart`
concept `camera_model` incorporates characteristic `camera_model is_fast`
concept `camera_model` incorporates characteristic `camera_model is_professional`

agent
    Definition: agent is organization or agent is person
person
    General_concept: agent
commercial_organization
    General_concept: organization
other_organization
    General_concept: organization
organization
    Definition: organization is commercial_organization or organization is other_organization
    General_concept: agent

agent has_produced product
    Synonymous_form: product is_produced_by agent
product is_produced_by agent
    See: agent has_produced product
    Concept_type: inverse_verb_concept

camera_maker
    Definition: agent that has_produced photo_camera
    Concept_type: verb_concept_role
    General_concept: agent

agent has_created creative_product
    General_concept: agent has_produced product

picture
photo
    General_concept: creative_product

photo is_coextensive_with picture
photo is_taken_by photo_camera
    Synonymous_form: photo_camera takes photo
photo_camera takes photo
    See: photo is_taken_by photo_camera
    Concept_type: inverse_verb_concept

person captured photo
    General_concept: agent has_created creative_product
    Synonymous_form: photo is_captured_by person
photo is_captured_by person
    See: person captured photo
    Concept_type: inverse_verb_concept
photographer
    Definition: person that captured photo
    Concept_type: verb_concept_role
    General_concept: person

product is_accessory_of product

accessory
    Definition: other_product that is_accessory_of product
    Concept_type: verb_concept_role
    General_concept: other_product
product has accessory

Concept_type: asymmetric_verb_concept

time
time is_production_date_of product
Synonymous_form: product has production_date

production_date
Definition: time that is_production_date_of product
General_concept: time
Concept_type: verb_concept_role
time_value
General_concept: datetime
Concept_type: role

production_date is_production_date_of product
Synonymous_form: product has production_date
product has production_date
See: production_date is_production_date_of product
Concept_type: inverse_verb_concept

time has time_value
Concept_type: property_association

feature
Concept_type: role
General_concept: number
product has feature
Concept_type: property_association

product_weight
Concept_type: role
General_concept: nonnegative_integer
General_concept: feature

product_price
Concept_type: role
General_concept: number
General_concept: feature

product_number
Concept_type: role
General_concept: number
General_concept: feature

product has product_number
Concept_type: property_association
General_concept: product has feature

product has product_price
Concept_type: property_association
General_concept: product has feature

product has product_weight
Concept_type: property_association
General_concept: product has feature

camera_weight
Concept_type: role
General_concept: nonnegative_integer
General_concept: product_weight

camera_price
Concept_type: role
General_concept: number
General_concept: product_price

bag_weight
Concept_type: role
General_concept: nonnegative_integer
General_concept: product_weight

photo_camera has camera_weight
Concept_type: property_association
General_concept: product has product_weight
photo_camera has camera_price
Concept_type: property_association
General_concept: product has product_price
camera_bag
General_concept: other_product
camera_bag has bag_weight
Concept_type: property_association
General_concept: product has product_weight
weight
Concept_type: categorization_type
Necessity: is_for general_concept photo_camera

Camera_by_weight
Necessity: segmentation for general_concept photo_camera that subdivides photo_camera by weight

lightweight_camera
Definition: photo_camera that has camera_weight less_than_or_equal_to 400
General_concept: photo_camera
Necessity: is_included_in Camera_by_weight
moderate_weight_camera
Definition: photo_camera that has camera_weight greater_than 400 and has camera_weight less_than_or_equal_to 800
General_concept: photo_camera
Necessity: is_included_in Camera_by_weight
heavy_camera
Definition: photo_camera that has camera_weight greater_than 800
General_concept: photo_camera
Necessity: is_included_in Camera_by_weight

flash
General_concept: electronics_product

photo_camera contains flash
Concept_type: partitive_verb_concept
General_concept: product contains product

memory_card
    General_concept: electronics_product
photo_camera contains memory_card
    Concept_type: partitive_verb_concept
    General_concept: product contains product
lens
    General_concept: electronics_product
battery
    General_concept: electronics_product
photo_element
    Definition: photo_element is photographic_film or photo_element is digital_sensor
    General_concept: electronics_product
photographic_film
    Definition: photo_element that is not digital_sensor
    General_concept: photo_element
digital_sensor
    Definition: photo_element that is not photographic_film
    General_concept: photo_element

resolution
    General_concept: positive_integer

digital_sensor has resolution
    Concept_type: property_association

photo_camera contains battery
    Concept_type: partitive_verb_concept
    General_concept: product contains product
photo_camera contains lens
    Concept_type: partitive_verb_concept
    General_concept: product contains product
photo_camera contains photo_element
    Concept_type: partitive_verb_concept
    General_concept: product contains product

shooting_mode
digital_photo_camera has shooting_mode

memory_card_slot
    General_concept: other_product
photo_camera contains memory_card_slot
    Concept_type: partitive_verb_concept
    General_concept: product contains product

white_balance_preset
photo_camera has white_balance_preset

photo_camera has_bag camera_bag
    General_concept: product has accessory

photographer uses photo_camera
    Synonymous_form: photo_camera is_used_by photographer
photo_camera is_used_by photographer
    See: photographer uses photo_camera
    Concept_type: inverse_verb_concept

product is_rated_by agent
rating
product_rating
Concept_type: categorization_type
Necessity: is_for general_concept product
rating is_rating_of product
Synonymous_form: product has rating
product has rating
See: rating is_rating_of product
Concept_type: inverse_verb_concept

rating is_given_by agent

Product_by_product_rating
Necessity: categorization_scheme for general_concept product that subdivides product by product_rating
Top_rating
General_concept: rating
Good_rating
General_concept: rating
Acceptable_rating
General_concept: rating
top_rated_product
General_concept: product
Necessity: is_included_in Product_by_product_rating
average_rated_product
General_concept: product
Necessity: is_included_in Product_by_product_rating
low_rated_product
General_concept: product
Necessity: is_included_in Product_by_product_rating

photo_camera photographed something
something is_presented_in picture

award
photo_exhibition
photographer has_participated_in photo_exhibition

award received_by photographer
award received_for photo
award received_in photo_exhibition

prize_value
Concept_type: role
General_concept: number
award has prize_value
Concept_type: property_association
first_name
General_concept: text
person has first_name
Concept_type: property_association

Nikon_D5100

General_concept: photo_camera

Olympus_EPL5
General_concept: photo_camera

Jim
General_concept: person
John
General_concept: person
Gytis_Gudas
General_concept: person
G._Gudas
General_concept: person
Mike
General_concept: person
Photo_1
General_concept: photo
Jim has_produced Photo_1

G._Gudas
General_concept: person
Gytis
General_concept: first_name
G._Gudas has first_name Gytis

It is impossible that digital_sensor is photographic_film;

It is impossible that electronics_product is creative_product;

It is impossible that electronics_product is other_product;

It is necessary that rating is_given_by exactly 1 agent;

It is necessary that rating is_rating_of exactly 1 product;

It is impossible that digital_photo_camera is film_photo_camera;

It is necessary that photo_camera has at_most 1 camera_model;

It is necessary that photo_camera contains exactly 1 lens;

It is necessary that photo_camera contains exactly 1 photo_element;

It is necessary that photo_camera contains exactly 1 flash;

It is necessary that photo_camera contains at_most 2 memory_card;

It is necessary that photo_camera contains at_least 1 memory_card;

It is necessary that photo_camera contains exactly 2 battery;

It is necessary that photo is_taken_by at_most 1 photo_camera;

It is impossible that digital_photo_camera is film_photo_camera;

It is necessary that award received_in at_most 1 photo_exhibition;

It is necessary that award received_for at_least 1 photo;

It is necessary that award received_by at_most 1 photographer;

It is impossible that Jim is John;

It is necessary that G._Gudas is Gytis_Gudas;

It is impossible that person is organization;

It is impossible that commercial_organization is other_organization;

It is necessary that photo_camera contains at_most 3 memory_card;

It is necessary that photo_camera contains at_least 1 memory_card;

It is necessary that digital_photo_camera has at_least 2 shooting_mode;

It is necessary that photo_camera contains at_most 2 memory_card_slot;

It is necessary that photo_camera has at_least 2 and at_most 9 white_balance_preset;

It is necessary that digital_photo_camera contains photo_element that is digital_sensor;