KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Aleksandr Kiričiuk

Užklausų formulavimo natūralia lietuvių kalba ontologijos pagrindu automatizavimas ir tyrimas

Magistro darbas

Darbo vadovas: doc. Rita Butkienė

KAUNAS, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Aleksandr Kiričiuk

Užklausų formulavimo natūralia lietuvių kalba ontologijos pagrindu automatizavimas ir tyrimas

Magistro darbas

Recenzentas
doc. Tomas Blažauskas
2012-05-23

Darbo vadovas
doc. Rita Butkienė
2012-05-23

Atliko
IFM-0/4 gr. stud.
Aleksandr Kiričiuk
2012-05-23

KAUNAS, 2012

Turinys

SANTRAUKA

Semantinis tinklas yra interneto tinklas kaip duomenų bazė, kurioje yra dirbtinio intelekto požymių. Viena iš pagrindinių semantinio tinklo idėjų siekia, kad žmogus su kompiuteriu bendrautų natūralia kalba, kai vartotojas klausia, o kompiuteris atsako.

Norint servisams perduoti daugiau ar mažiau intelektualias užduotis, informacinei sistemai reikia suteikti žinių, kurių pagalba galėtų suprasti įvestos informacijos turinį. Tam tikslui yra naudojama ontologija, kurios viena iš aprašymo kalbų yra RDF, o duomenų išrinkimui ir užklausos pateikimui naudojama SPARQL kalba.

Kad klausimus sistemai vartotojas galėtų užduoti natūralia lietuvių kalba reikalinga posistemė, kuri pagal taisykles (ontologija) transformuotų NLK į formalią kalbą ir sudarytų SPARQL užklausas. Taip pat ši posistemė turėtų transformuoti SPARQL gautus rezultatus į logišką, taisyklingą NLK sakinį.

Darbo tikslas – pasiūlyti įrankį, priemonę kuri leis formuluoti NLK semantines užklausas, jas įvykdyti sukuriant prototipinę paieškos ontologijoje sistemą ir užklausos rezultatus pateiktų NLK.

Darbo uždaviniai:

1. Išanalizuoti ontologijos užrašymo kalbas ir jų kūrimo įrankius;
2. Išanalizuoti SPARQL užklausų kalbą ir užklausų vykdymo įrankius;
3. Nustatyti lietuvių kalbos gramatikos taisyklių įtakojančių užklausų formulavimą natūralia lietuvių kalba pagrindinius aspektus;
4. Suprojektuoti ir realizuoti prototipinę paieškos ontologijoje sistemą, kurioje užklausos būtų formuluojamos NLK;
5. Nustatyti sukurtos sistemos ypatybes atliekant eksperimentinį tyrimą;

Tyrimo objektas – užklausų formulavimas natūralia lietuvių kalba ir jų transformavimas į SPARQL užklausas.

SUMMARY

Semantic network is the Internet network as a database, which contains an artificial intelligence character. One of the main idea of the Semantic Web seeks to make people interact with computers in natural language, when the user asks the computer is responsible.

Service centers to transfer intellectual tasks to information systems to provide knowledge which could help to understand the content of the information entered. For this purpose, an ontology is used, which one of description language is RDF, for data retrieval and query presentation system uses SPARQL language. The user to ask questions system in natural Lithuanian language needed subsystem which transforms NLL to formal language and and create queries according to the rules. Also the subsystem transforms SPARQL results in the correct NLL sentence.

The purpose of the work - to offer a tool that will allow to formulate semantic queries using NLL, to carry out them in the development of a prototype search system and give correct results.

Objectives:

1. To analyze the transcription ontology languages and their development tools;
2. To analyze the SPARQL query language and query processing tools;
3. Set the Lithuanian language grammar rules affect the formulation of queries in natural Lithuanian language key aspects;
4. To design and implement a prototype system for ontology search, where queries are formulated in NLL
5. Identify the characteristics of the system established by experimental research.

Research object - query formulation in natural Lithuanian language and the transformation to SPARQL queries.

1. ĮVADAS

Šiuolaikinis pasaulis jau nebeįsivaizduojamas be informacinių technologijų ar tiesiog be kompiuterio. Nors pasaulinė bendruomenė yra pripažinusi anglų kalbą kaip tarptautinę bendravimo, verslo, bei kitų sričių kalba, būtina puoselėti ir lietuvių kalbą. Vienas iš gimtosios kalbos puoselėjimo ir stiprinimo būdų – tarptautinių produktų adaptavimas lietuvių rinkai, tai jau yra sėkmingai vykdoma.

Technologijoms vis sparčiau artėjant prie Semantinio tinklo (ang. *Semantic Web*) idėjų, kurios leidžia informacijos ieškoti pagal prasmę, o ne vien pagal raktinius žodžius, kur paieška atliekama ieškant žodžių, frazių atitikmenų internetiniame tinkle. Viena iš pagrindinių semantinio tinklo idėjų siekia, kad žmogus su kompiuteriu bendrautų natūralia kalba. Žingsnį prie intelektualaus internetinio tinklo kūrimo suteikia ontologijos.

Semantinis tinklas yra interneto tinklas kaip duomenų bazė, kurioje yra dirbtinio intelekto požymių. Viena iš pagrindinių semantinio tinklo idėjų siekia, kad žmogus su kompiuteriu bendrautų natūralia kalba, t.y. vartotojas klausia, o kompiuteris atsako. Kompiuteris internetiniame tinkle gali saugoti informaciją, tačiau tam, kad informaciją suprasti, interpretuoti bei surinkti yra reikalingas žmogus. Internetinis tinklas mums leidžia atskirai identifikuoti elektronines bibliotekas, duomenų bazes, daugialypes terpes (ang. *multimedia*), tačiau vartotojas ne retai nori užduoti klausimus, į kuriuos atsakant esamos technologijos susiduria su sunkumais ieškant reikalingos informacijos, nes atsakyti į užklausą reikalingos numanomos žinios. Pavyzdžiui, rasti gyvūnus, kurie naudojasi echolokacija, tačiau nėra delfinai arba šikšnosparniai. Šis klausimas yra sudėtingas, nes, pavyzdžiui, užėjus į www.google.lt, galima duoti užklausą, kad ieškoma tam tikro žodžio, ir google.com jį ras, tačiau atlikti paieškos pagal sakinio prasmę mes negalime, nes tam reikalingos intelektualios žinios. Servisas kuriame mes galėtumėme įvesti intelektualią užklausą natūralia lietuvių kalba (NLK) mums kol kas nėra teikiamas. Norint perduoti daugiau ar mažiau intelektualias užduotis servisams, pavyzdžiui, „įsigyti kelionę į šiltus kraštus, nelabai toli, kur yra kalbama lietuviškai“, informacinei sistemai reikia suteikti žinių, kurių pagalba galėtų suprasti įvestos informacijos turinį. Pagal anksčiau pateiktą pavyzdį šiuo metu, abejotina, ar kokia sistema duos atsakymą, tačiau teoriškai šis uždavinys turi sprendimą.

Intelektualias užklausas sunku realizuoti, nes vartotojas žiūrėdamas į pateiktą informaciją pagal duomenų atvaizdavimą (šriftą, struktūrą, spalvą), intuityviai žino, kur kokia informacija yra nurodoma. Žmonėms taip pat prieinama ir semantinė informacija. Ši informacija yra nesuprantama kompiuteriui. Kompiuteriui svetainėje esanti tekstinė informacija yra hieroglifai, simboliai – jokios naudingos informacijos kompiuteris iš šio teksto negali išgauti. Tam, kad tekstas būtų aiškesnis kompiuteriui, tekstą turime struktūrizuoti žymų (angl. *tag*) pagalba. Kompiuteris neatskirs kas yra

žymų viduje, tačiau jis žinos, kad tarp žymų <pavadinimas></pavadinimas> yra pavadinimas, jei ieškoma kas susiję su pavadinimu, kompiuteris galės panaudoti tarp žymų esančią informaciją atsakant į užklausą. Problema kyla dėl to, kad kompiuteriui žymos taip pat neturi jokios prasmės, nes tik vartotojas žino, kad <pavadinimas> – mūsų srities pavadinimas. Kompiuteris mato žymas kaip sintaksę, jis gali patikrinti tik dokumento teisingumą, bet žymos pavadinimas kompiuteriui nieko nesako, todėl reikalinga semantika, loginiai ryšiai tarp saugomų duomenų. Kad kompiuteris žinotų žymos tikrąją reikšmę, ryšį su kitais duomenimis, yra naudojamos ontologijoje.[1][2]

Ontologija – dirbtinio intelekto srityje ji suvokiama kaip tam tikros srities sąvokų visumos specifikavimas išreikštu pavidalu (angl. *explicit specification of a conceptualization. [19]*)

Ontologijų viena iš aprašomų kalbų yra RDF, duomenų išrinkimui užklausos pateikiamos SPARQL kalba. Kad klausimus sistemos vartotojas galėtų užduoti NLK reikalinga posistemė, kuri pagal taisykles (ontologija) transformuotų NLK į formalią kalbą ir sudarytų SPARQL užklausas. Taip pat ši posistemė turėtų transformuoti SPARQL gautus rezultatus į logišką, taisyklingą NLK sakinį. NLK užduodamus klausimus yra sunku apdoroti programiškai, nes juos sudarantys žodžiai yra įvairių formų: skirtingų linksnių, giminių, laikų, tai kelia daug problemų, kai tuos žodžius reikia sutapatinti su informacijos šaltinio (ontologijos), kuriame ieškoma informacijos, elementais. Taigi siekiama pasiūlyti įrankį, priemonę, kuri leis suformuoti užklausą iš NLK, užklausos rezultatus pateiktų NLK.

Darbo tikslas – pasiūlyti įrankį, priemonę kuri leis formuluoti NLK semantines užklausas, jas įvykdyti sukuriant prototipinę paieškos ontologijoje sistemą ir užklausos rezultatus pateiktų NLK.

Darbo uždaviniai:

1. Išanalizuoti ontologijos užrašymo kalbas ir jų kūrimo įrankius;
2. Išanalizuoti SPARQL užklausų kalbą ir užklausų vykdymo įrankius;
3. Nustatyti lietuvių kalbos gramatikos taisyklių įtakojančių užklausų formulavimą natūralia lietuvių kalba pagrindinius aspektus;
4. Suprojektuoti ir realizuoti prototipinę paieškos ontologijoje sistemą, kurioje užklausos būtų formuluojamos NLK;
5. Nustatyti sukurtos sistemos ypatybes atliekant eksperimentinį tyrimą;

Tyrimo objektas – užklausų formulavimas natūralia lietuvių kalba ir jų transformavimas į SPARQL užklausas.

Tyrimo sritis yra:

- Ontologijos ir jų užrašymo kalbos:
  - RDF (angl. *Resource Description Framework*);
  - RDFS (angl. *Resource Description Framework Schema*);
  - OWL (angl. *Web Ontology Language*);
- Užklausų kalba SPARQL (angl. *Simple Protocol and RDF Query Language*) ;
- Priemonės , įrankiai užklausoms formuluoti ir vykdyti;
- Lietuvių kalba ir jos gramatika.

2. ANALIZĖ

2.1. Analizės tikslas

Tikslas įgyti gilesnį supratimą, suvokimą apie ontologijas, lietuvių kalbos gramatikos specifiką, be kurios neišeina sklandžiai formuluoti sakinių. Išanalizuoti užklausų formulavimo galimybes.

2.2. Tyrimo objekto analizė

2.2.1. Ontologijos ir jų užrašymo formatai

Kas yra semantinis tinklas? Didelė dalis dabartinio internetinio turinio orientuota žmonių naudojimui. Pateikimo kalbos, tokios kaip HTML, savyje turi internetinėms naršyklėms skirtas instrukcijas, nurodančias kaip priimtinai žmogaus vizualiniam ir klausos suvokimui atvaizduoti turinį. Tačiau, jeigu mes norėtumėm kompiuterinės programos pagalba atlikti paiešką internete esančios informacijos, tokia sistema susidurtų su sunkumais iš šių internetinių puslapių išgaunant kokią nors prasmę, jeigu ji neturėtų išplėstų žmogaus kalbos suvokimo įgūdžių.

Semantinis tinklas orientuojamas į tai, kad interneto tūrinį padaryti suprantamą mašinoms, tam kad tūrinį galėtų analizuoti programų sistemomis ir platinti tarp interneto servisų. Šiuo tikslu W3C rekomenduoja kelias orientuotas į internetą kalbas, kurios gali būti naudojamos internetinio tūrinio formalizavimui, duomenų ir informacijos nurodoma formaliai apibrėžta prasme, informacija susijusi semantiniais ryšiais. Tam, kad kuriamai sistemai galėtumėm teikti intelektualius klausimus, tam mums reikia sukurti ontologiją, kuri sistemai suteiks dirbtinio intelekto požymį.

XML – tai duomenų struktūros bei jų turinio aprašomoji kalba. Pagrindinis XML kalbos paskirtis yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingų sistemų, dažniausiai sujungtų internetu. Visos pagrindinės ontologijos kalbos pagrįstos XML struktūra, tačiau semantinio tinklo kalbos turi daug didesnę išraiškos galimybę.

RDF – pagrindinė semantinio tinklo technologija, kuri gali būti naudojama formaliam ryšių aprašymui tarp resursų, kurių pagalba kompiuterinėms sistemoms leidžiama naudotis visa struktūrizuota informacija, esančia internetiniame tinkle. RDF pagrindinis tikslas – vienu paprastu būdu bet kokį faktą aprašyti tokia struktūra, kad juos galima būtų apdoroti kompiuterinėmis programomis.

RDF skiriasi nuo XML ir kitų technologijų tuo, kad RDF yra skirtas platinti duomenis visame pasauliniame tinkle. RDF‘ui ypač svarbi prasmė, viskas kas aprašoma RDF faile, turi savo prasmę, nurodo į konkretų objektą, abstrakčią sąvoką, faktą. RDF pagrindu sudaryti standartai aprašantys logines išraiškas susiejančias šiuos faktus ir nurodo kaip galima surasti pačius faktus didžiulėje duomenų bazėje pateiktoje RDF‘e.

RDF naudojamas kai:

- reikia apjungti duomenis iš skirtingų šaltinių nekuriant specialių programų;
- reikia suteikti kitiems perėjimą prie jūsų duomenų;
- reikia išskleisti duomenis, kad jais nesinaudotų kas nors vienas;
- reikia padaryti kažką ypatingo su dideliais kiekiais informacijos – įvesti, išgauti, peržiūrėti, analizuoti, atlikti paiešką ir t.t.;

RDF galima apibrėžti kaip trijų taisyklių rinkinį:

- faktas yra išreiškiamas triguba forma (veiksnys, tarinys, papildinys) – panašu į natūralia kalba išreikšto sakinio struktūrą;
- veiksnys, tarinys, papildinys – tai yra realaus pasaulio subjekto vardai, konkretūs arba abstraktūs. Vardai gali būti:
  - globalūs, nurodo vieną ir tą patį objektą visose RDF dokumentuose, kuriuose jie yra naudojami;
  - lokalūs, į šiais vardais pavadintus objektus negalima tiesiogiai kreiptis iš už šio dokumento ribų;
- papildymai gali būti tekstinėmis eilutėmis – konstantomis;

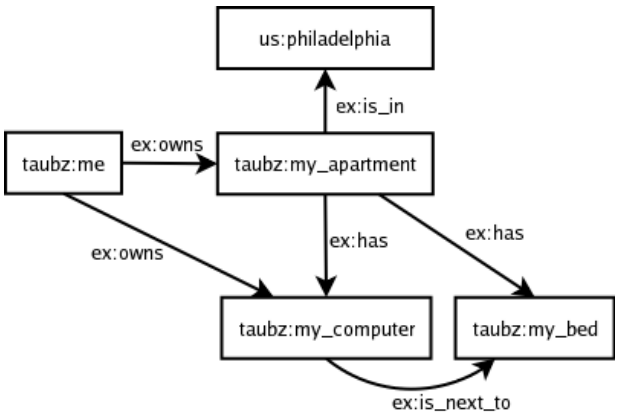
Yra du galimi būdai nagrinėti RDF pateiktą informaciją:

- RDF duomenų rinkinį laikyti teiginių rinkiniu, kuriuose kiekvienas teiginys yra faktas (kaip buvo pavaizduota aukščiau);
- RDF duomenų rinkinį laikyti grafu.

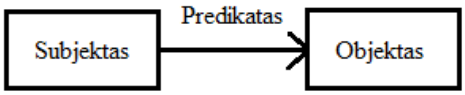
RDF duomenų rinkinyje informacija yra išreiškiama grafų pavidalu (Pav. 1). Grafo viršūnės atitinka resursus arba reikšmes, o lankai – resursų savybes.

Nagrinėjant RDF grafas yra išreiškiamas tripletais (Pav.2):

- Objektas (angl. *object*) – duomenys;
- Subjektas (angl. *subject*) – resursai;
- Predikatas (angl. *predicate*) – savybės, atributai [3],[4],[5].



Pav. 1 RDF grafas



Pav. 2 Grafo mazgas

RDF schema (RDFS) nustato objektiškai orientuotą modelį RDF'ui. RDF- schema apibrėžia klasių atstovavimą, paveldėjimo santykius, savybes, duomenų tipus ir pan. Pavyzdžiui, pavyzdyje (Pvz. 1) pavaizduotas RDFS kalba parašytas failas, kuriame aprašyta klasė „Gamintojas" ir savybė „Pavadinimas".

```
<rdf:RDF xml:base=" http://www.semanticweb.org/ontologies/2012/0/Auto.rdf"
      xmlns=" http://www.semanticweb.org/ontologies/2012/0/Auto.rdf#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

 <rdfs:Class rdf:ID="Gamintojas"/>
 <rdf:Property rdf:ID="Pavadinimas">
  <rdfs:domain rdf:resource="#Gamintojas"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </rdf:Property>
</rdf:RDF>
```

Pvz. 1 RDFs struktūros pavyzdys

RDF vardai yra dviejų tipų, globalūs ir lokalūs vardai. Globalūs vardai visur turi vieną ir tą pačią prasmę, visuomet turi URI (angl. *Uniform Resource Identifier*) pavidalą. URI sintaksė (formatas) gali būti tokia pati kaip ir svetainės adresas, globalus objekto vardas gali būti suteiktas taip: *http://www.w3.org/2000/01/rdf-schema#Class*. Vardo panašumas į svetainės adresą yra tik sutapimas.

URI dažnai daliną į vardų aibės (angl. *namespace*) ir lokalius vardus, kur vardų aibė gali būti sutrumpinta prefiksinės notacijos pagalba. Pavyzdžiui, *rdfs:Class* yra

http://www.w3.org/2000/01/rdf-schema#Class trumpinys tuo atveju, jei prefiksas *rdfs* buvo paskelbtas failo antraštėje. Jeigu prefiksas nenurodytas (kaip „Gamintojas“ atveju), šis vardas bus naudojamas failo ribose, turės lokalų charakterį. Kad supaprastinti supratimą, kurtoje ontologijoje naudojama sutrumpinta notacija, paremta lokaliais vardais.

Vardų aibę galima sulyginti su paketais naudojamais į objektą orientuotose kalbose. Todėl, galime teigti, kad Pvz.1 pateiktam pavyzdyje yra apibrėžiamas *http://www.semanticweb.org/ontologies/2012/0/Auto.rdf#* paketas. Visi faile aprašyti resursai yra bendrai prienami, tam kad visi RDF failai tiesiogiai galėtų kreiptis vieni į kitus. Pavyzdžiui, galime sukurti kitą RDF failą, kuris nurodytu į pavyzdyje pateikto klases „Gamintojas“ egzempliorių ir nurodytų šiam egzemplioriui konkretų pavadinimą. Tokie egzemplioriai RDF‘e vadinami individais. Skirtingai nuo kitų į objektą orientuotų kalbų, individai gali būti tiesiogiai priskirti daugiau nei vienam tipui. Pavyzdžiui, individas *http://www.semanticweb.org/ontologies/2012/0/Produktai.rdf#VW* gali būti objektu ir kaip egzempliorius *http://www.semanticweb.org/ontologies/2012/0/Auto.rdf#Gamintojas*, ir kaip egzempliorius *http://www.semanticweb.org/ontologies/2012/0/Aukcionas.rdf#Preke*. Tai leidžia naudoti vieną ir tą patį resursą (nurodyta per jo URI) viename kontekste kaip produktą, o kitame kontekste kaip aukciono lotą.

RDF schemos klasės turi savo individus su bendromis charakteristikomis. Klasėse gali būti nurodyti hierarchiniai paveldėjimai, panašiai kaip ir į objektą orientuotose sistemose. Kaip ir UML, RDF schema palaiko daugialypį paveldėjimą. Pagrindinis skirtumas tarp RDF ir į objektą orientuotomis kalbomis – klasės gali persikirsti. Kadangi individas gali turėti kelis tipus, kai kurie egzemplioriai gali būti bendri kelioms klasėms. Taip pat, egzemplioriai gali keisti savo tipą egzistavimo ciklo metu. Užsakymas gali pradėti savo egzistavimą kaip klasės „UžsakymoTvarka“, o tada pakeisti savo tipą į „BazinisUžsakymas“, kaip sistema surinks daugiau duomenų apie užsakymo specifikaciją.

RDF savybės yra atskiri subjektai, kurie gali būti apibrėžti nepriklausomai nuo klasių ir juos gali naudoti kelios klasės vienu metu. Pavyzdžiui, galime apibrėžti savybę „Pavadinimas“ ir tada priskirti toms klasėms, kuriose pavadinimas turi prasmę.

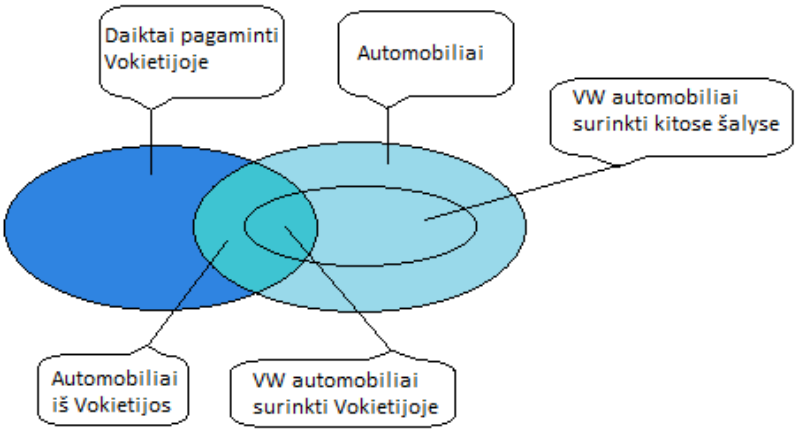
Tam, kad prijungti arba asocijuoti savybes su klasėm naudojama *rdfs:domain* išraiška. *Rdfs:domain* – tai žymė iš RDF *Schema* vardų aibės, kuri predikato pagalba savybę priskiria klasei. Paskutiniame pavyzdyje savybei „Pavadinimas“ domenas yra „Gamintojas“. Iš esmės, tai reiškia, kad visos klasės „Gamintojas“ esybės gali būti susietos su savybę „Pavadinimas“, tokiu būdu savybė „Pavadinimas“ tampa klasės gamintojas atributu. Taip pat RDF ir OWL ši išraiška turi papildomų išraiškų: bet kuris resursas, kuris yra „Pavadinimas“ subjektu yra klasės „Gamintojas“ egzemplioriumi. Kitaip tariant, domeno savybės gali būti naudojamos subjekto klasifikavimui. Todėl

grįžtant prie pavyzdžio, jeigu kas nors turi gamintojo pavadinimą, tai jį galima traktuoti, kaip „Gamintojas“ egzempliorių, net jei jis naudojamas ir kituose specifikacijose.

Tokiu būdu, paprastos savybės, tokios kaip „pavadinimas“ RDF vadinami literalais. Literalai priskiriami XML *Schema* duomenų tipui, tokiam kaip *xsd:string* arba *xsd:float*. Galima nurodyti apribojimus savybių reikšmėms *rdfs:range* išraiškos pagalba. Priskiriamų savybei reikšmių diapazonas gali turėti arba XML *Schema*, arba klasės tipą.

RDF *Schema* yra paprasta ontologijų modeliavimo kalba, panaši į objektą orientuotą kalbą. Galime nustatyti klases ir jų savybes, ir tada sukurti šių klasių egzempliorius. Tai yra naudinga įvairiems tikslams, tačiau daugeliuose taikomuosiuose srityse vienos tik RDF *Schemos* išraiškos neužtenka. Pavyzdžiui, RDF *Schema* negali nustatyti kardinaliai išreikštų ribų, taip kad vienas gamintojas turėtų tik vieną pavadinimą.

Internetinės ontologijos kalba (angl. *Web Ontology Language*, OWL) išplečia RDF schemą ir naudodama tokia pačia sintaksę kaip ir RDF ir jo bazinę gramatiką. OWL papildo duomenų savybių charakteristikų išreiškimo galimybes ir apibrėžia elementų grupavimą į klases tų elementų, kurie atitinka šią charakteristikas. OWL klasių komponentai – aksiomos.



Pav. 3 OWL klasės gali būti nagrinėjamos kaip subjektų turinčių bendrų savybių rinkinys.

Apskritimas kairėje, kuris pateiktas paveiksle 3 aprašo Vokietijoje pagamintų daiktų klasę, apskritimas dešinėje atstovauja visus gaminamų automobilių egzempliorius. Dviejų didžiųjų apskritimų sandūra atitinka visus automobilių egzempliorius, kurie yra pagaminti Vokietijoje. Mažuoju apskritimu nurodyta, viduryje automobilių pagamintų automobilių, pagamintų automobilių klasė, kurių gamintojas yra Volkswagen. Kairiojo apskritimo ir mažojo apskritimo sankirtoje nurodyti automobiliai surinkti Vokietijoje Volkswagen gamintojo.

Nepamirštant, kad savybes RDF/OWL nepriklauso nuo atskirų klasių ir gali būti naudojamos keliuose vietose. Pavyzdžiui, savybė *Kilmė* gali būti naudojamas gaminiams, bei automobiliams, gamintojams bei kitoms klasėms aprašyti. Siūlant, kad Vokietija kažkur aprašyta, kaip egzempliorius klasės *Šalis*, kai kas galėjo naudojantis OWL , kad formalizuotai aprašyti visų daiktų klases, kurių

savybė *Kilmė* turi reikšmę Vokietija. Tada visi atvejai, apibrėžti kam kitam, gali būti klasifikuojami pagal šį apibrėžimą.

OWL kalbos elementas, naudojamas išraiškų apibrėžimui, vadinamus apribojimais (angl. *restriction*). OWL turi įvairių tipų apribojimų. Aukščiau pateiktame pavyzdyje turime taip vadinamų apribojimą „turiReikšmę“ (angl. hasValue), kuris susietas savybe su konkrečiu individu. Pagrindinis OWL privalumas yra tame, kad klasėms gali būti nustatomi keli apribojimai ir klasių deriniai. Pavyzdžiui, galima būtų nustatyti klasę visų gamintojų iš Italijos, kurie surenka ne mažiau 3 automobilių modelius arba gamina mažiausiai 1000 automobilių per metus ir pan..

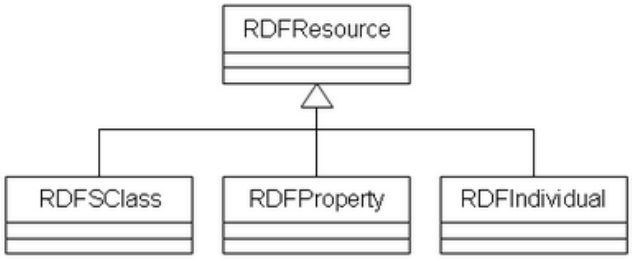
Objektiškai orientuotose sistemose panašūs išsireiškimai būtų paslėpti paties kodo viduje. Semantinių tinklų ontologijos loginės operacijos realizuojamos apibrėžiant per OWL klases ir kitais formaliais išsireiškimais. Tai ne tik palengvina vartotojams specifinių reikšmių supratimą, bet reiškia, kad ir kiti programiniai įrankiai gali panaudoti šiuos apibrėžimus. OWL modeliai tiesiog deklaruoja daiktus, o pareiga atlikti kokį naudingą darbą su šiomis deklaracijomis atitenka programinei įrangai.

Siekiant pasinaudoti semantinio internetinio technologijų objektiškai pagrįstų sistemų privalumais, programinės įrangos architektai turi suprasti projektavimo šablonus bei strategiją sklandžiam šių technologijų integravimui.

Kad suprasti architektūrą paremta ontologijomis reikia suprasti, kad ontologijoje :

- Savybės nepriklauso nuo atskirų klasių;
- Egzemplioriai gali turėti kelis tipus ir klasifikacijos metu gali keisti savo tipą;
- Klasės gali dinamiškai nusistatyti vykdymo metu.

Šie pagrindiniai išskirtinumai įtakoja tai, kad nėra taip lengva suprojektuoti RDFS/OWL klases objektiškai orientuotose klasėse, kuriuose atributai pririšti prie klasių ir t.t. Kaip parodyta paveiksle 4, tipinis objektinis modelis ontologijos pateikimui semantiniame tinkle, savyje sudaro klases resursų aprašymui, klasių savybes bei individus. Klases *RDFSClass* ir *RDFProperty* priskiriami klasėms *rdfs:Class* ir *rdfs:Property*, apibrėžia RDFS, tuo tarpu *RDFIndividual* neturi tiesioginio atitikmens RDFS‘e. Šio tipo struktūrą nesunkiai galima išplėsti suteikti daugiau OWL klasių tipų. Sudarant ontologiją bus naudojama paveiksle 4 pavaizduota ontologijos pateikimo RDF schema, kurią praplėsime OWL klasės tipais.[13],[14],[17]



Pav. 4 Paprasta objektiškai orientuotas modelis ontologijos pateikimo RDF schema

Informacijai iš RDF grafų išgauti reikalingos užklausų kalbos. Daugeliu atveju RDF duomenų masyvą sudaro daugybė grafų, todėl užklausos turi turėti galimybę duoti užklausą, kuri apimtų informaciją daugiau negu iš vieno grafo. Technologiniu požiūriu, užklausomis grįstų samprotavimų standartas yra SPARQL. SPARQL užklausa vykdoma į RDF duomenų rinkinį, kuris sudarytas iš daugelio grafų.

2.2.2. Užklausų kalba SPARQL

SPARQL (Simple Protocol and RDF Query Language) – tai 2008 metų sausio mėnesį W3C konsorciumo (angl. World Wide Web Consortium) rekomenduota ontologijų užklausų kalba, išsivysčiusi iš RDQL kalbos. Ši kalba – tai įrankis, leidžiantis išgauti norimą informaciją iš didelių RDF duomenų rinkinių. SPARQL kalboje naudojamos užklausos panašios į naudojamas SQL duomenų bazių užklausų kalboje.

Bendra SPARQL užklausų schema atrodo taip:

```
PREFIX foo: <http://example.com/resources/>
# Priešdėliniai aprašai – naudojami trumpinimams.
FROM ...
# Užklausos išteklis – nustato kokie RDF grafai yra užklausiami.
SELECT ...
# Nurodyti rezultatai – grąžinamas duomenų rinkinys, tenkinantys užduotus reikalavimus.
WHERE {...}
# Užklausos kriterijai – nurodoma, ką užklausti baziniame duomenų rinkinyje.
ORDER BY ...
# Užklausos modifikavimas – apriboja, rikiuoti ar kitaip transformuoti užklausos rezultatus.[16]
```

Pavyzdyje 2 pavaizduota užklausa kuri iš ontologijos Auto (PRIEDAS 9.1.) grąžina Komplekto klasės tipo pavadinimo egzempliorius, kuris siejasi su klase Salonas. Tai iliustruoja semantinio tinklo viziją kaip vieną didelę duomenų bazę.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Auto: <http://www.semanticweb.org/ontologies/2012/0/Auto.owl#>
 select distinct ?y
where{
?x rdf:type Auto:Komplektas.
?x Auto:Komplektas_Tipas ?y.
?x Auto:kas_sudarytas_iš_ko ?n.
?n rdf:type Auto:Salonas.
}
```

Pvz. 2 SPARQL užklausos pavyzdys

Žemiau pateikta dalis SPARQL užklausose naudojamų raktinių žodžių, visas sąrašas prieinamas oficialioje dokumentacijoje [6].

PREFIX – naudojamas URI trumpinimui.

OPTIONAL – pažymi nebūtiną šabloną.

GRAPH – jo pagalba yra formuojama užklausa, kuris vardiniui grafui pritaiko šabloną.

DISTINCT – nurodo, kad kiekvienas atsakyme esantis sprendinys bus unikalus.

LIMIT – nurodo didžiausią galimų rezultatų kiekį.

OFFSET – leidžia atsakyme neatvaizduoti pirmuosius n sprendinių.

ORDER BY – leidžia rezultatus surūšiuoti didėjimo (ASC()) arba mažėjimo (DESC()) tvarka. [6]

SPARQL užklausų kalbą naudosime prototipinėje paieškos sistemoje, duomenų išrinkimui iš ontologijų.

Kad galėtumėme SPARQL užklausų pagalbą kreiptis į ontologiją, mums reikalinga SPARQL prieigos taškas (angl. *endpoint*). SPARQL prieigos taške užklausos yra apdorojamos ir grąžinamas rezultatas įvairiu formatu. SPARQL prieigos taškas iš esmės sugalvotas kaip servisas, kurio pagalba galime kreiptis į ontologijas.

Egzistuoja dviejų tipų SPARQL prieigos taškai, bendro naudojimo bei lokalūs. Bendro naudojimo prieigos taškai gali atlikti bet kuriuose nurodytuose RDF dokumentuose, esančiuose tinkle. Lokalūs prieigos taškai gali gauti duomenys tik iš vieno resurso. Pagrindiniai SPARQL prieigos taškai yra Jena, Sesame, Joseki, OpenLink Virtuoso.

2.2.3. Užklausų formulavimas natūralia kalba

Kaip jau buvo minėta, viena iš pagrindinių semantinio tinklo idėjų siekia - žmogus su kompiuteriu bendrauja dialogu, t.y. vartotojas klausia, o kompiuteris atsako, pataria. Tam, kad tai

realizuoti, kompiuteris turi suprasti, ko mes iš jo norime. Šiam tikslui galime naudoti klausiamuosius arba liepiamuosius sakinius.

Klausiamiesiems sakiniai būdingi dialoginei kalbai, jais klausiama, teiraujamasi, norima sužinoti. Klausiamiesiems sakiniams būdingi klausiamieji žodžiai įvardžiai: kas, kuris, koks, katras, kelintas, keli, kieno; įvardiniai prieveiksmiai kur, kada, kaip. Pavyzdžiui „kur yra Volksvagen būstinė?“ . Taip pat yra patikrinamieji sakiniai su klausiamosiomis dalelytėmis ar, gal; be, bene, nejau, nejaugi, jaugi, negi. Pavyzdžiui, „Ar modelis Golf yra komplektuojamas su odiniu salonu?“. Į patikrinamuosius klausimus gali būti dvejopi atsakymai „taip“ arba „ne“. Yra klausiamųjų sakinių ir be klausiamųjų žodžių, jie dažnai primena sakinius su konstrukcija „ar“, tokie klausimai turi ryškesnę klausiamąją intonaciją „Volksvagen centrinė būstinė yra Volfsburge?“ Klausiamieji sakiniai neturi griežtos sakinio struktūros, klausimui didelę įtaka turi intonacinis pabrėžimas, dažnai yra naudojamos intuityvios žinios, todėl juos yra sunku programiškai analizuoti ir suprasti vartotojo užduoto sakinio turinį. [11]

Liepiamieji sakiniai kaip ir klausiamieji sakiniai būdingi dialoginiai kalbai. Įvertinus liepiamosios formos sakinio pobūdį, yra manoma vartotojui lengviausiai su kompiuteriu bendrauti liepiamųjų sakinių pagalba, vartotojas liepia – kompiuteris vykdo. Pavyzdžiui:

Rasti važiuoklės amortizatoriaus tipą, kuris sudaro komplektą Golf 6 Trendline

Liepiamojo sakinio struktūra yra artimiausia ontologijos užklausų (SPARQL) kalbos struktūrai. Pateiktame pavyzdyje galime matyti sąsają su SPARQL sintaksę:

„rasti“ – „select“ (norimi rezultatai);

„važiuoklės“ – „from“ (užklausos išteklis, ontologijos klasė);

„amortizatoriaus tipą“ – „select amortizatorius_tipas“ (nurodyti rezultatai, ontologijos klasės savybė);

„kuris“ – „where“ (užklausos kriterijai);

„sudaro komplektą Golf 6 Trendline“ – užklausos kriterijai.

Taip pat pateiktame pavyzdyje galime pastebėti ir atitikmenis su RDFS vardų aibės žymėmis:

„važiuoklė“, „komplektas“ – rdfs:Class;

„amortizatorius tipas“ – rdfs:Property;

„sudaro“ – rdfs:Property;

„Golf 6 Trendline“ – rdfs:Individual.

Remiantis šiais panašumais, galime teigti, kad liepiamieji klausimai savo struktūra yra panašūs į formalią kalbą, todėl šiuos klausimus sudarant – apdorojant programinei sistemai tūrėtų kilti mažiau sunkumų, šį formuluote ir buvo pasirinkta kuriant prototipą.

2.2.4. Užklausų įvedimo būdai

Norimas užklausas paieškos sistemoje galime formuluoti keliais būdais:

- nenaudojant natūralios kalbos;
- naudojant ribotą natūralią kalbą;
- naudojant natūralią kalbą.

Formuojant užklausą nenaudojant natūralios kalbos iš duotų sąrašų išrenkami raktiniai žodžiai, kriterijai, temos, tam tikri šablonai yra užpildomi reikšmėmis arba grafiškai suformuluojant užklausą. Toks būdas turi savo privalumų ir trūkumų. Privalumas, kad eliminuojamos galimybės suklysti formuluojant užklausas, yra žinoma, ko sistemos galima klausti, visuomet gaunamas tinkamas (tikėtinas) atsakymas. Šis būdas turi ir trukumų. Pagrindinis šių užklausų formulavimo trūkumas yra tas, kad sąsajos yra sudėtingos, ribotas užklausų kiekis, norint praplėsti užklausas reikalingas papildomas programavimas. Paveiksle 5 yra pavaizduojama kaip iš sąrašų yra išrenkami kriterijai pagal kuriuos filtruojami duomenys, klientas neturi galimybės laisvai vesti žodžių, vartotojas visus duomenys turi rinktis iš pateikto šablono.

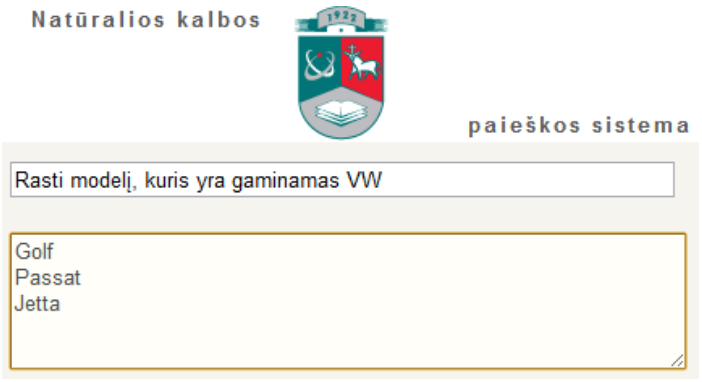
Pav. 5 Paieškos sistema naudojant formalią kalbą

Kitas būdas – naudojant ribotą natūralią kalbą. Šis būdas naudojamas įvestiems raktiniams žodžiams, frazėms ar klausimams, pritaikant ribotą kiekį taisyklių. NLK vedami klausimai turi tik tam tikrą struktūrą, formą, naudojami tik tam tikros dalykinės srities žodyno terminai (konceptus, ryšius, veiklos taisykles). Šio būdo privalumas yra tas, kad užklausos dydis nėra ribojamas, vartotojui yra suteikiama didelė laisvė formuluojant užklausas. Tačiau svarbus trūkumas tas, kad vartotojas gali suformuluoti dviprasmiškus klausimus, ne visada yra gaunami tinkami/tikėtini rezultatai, taip pat yra ribotas užklausos formų kiekis. Kaip paveiksle 6 parodyta iššokančiuose laukuose yra tipiniai pasirinkimo variantai, kuriuos klientas privalo pasirinkti, o tekstinio lauko laukelyje vartotojas gali įrašyti norimos savybės reikšmę.

Pav. 6 Paieškos sistema naudojant ribotą natūralią kalbą

Trečias būdas – užklausos natūralia kalba. Šios užklausos privalumai yra galimybė suformuoti bet kokią užklausą, nėra formų apribojimų, plečiant sistemą nereikalingas papildomas programavimas, tačiau išlieka galimybė suformuluoti dviprasmiškus klausimus, ne visada gaunamas tinkamas, tikėtinas atsakymas.

Naudojant natūralią kalbą duomenys turi būti įvedami taisyklingais lietuvių kalbos sakiniais, t.y. sakinyje žodžiai turi būt suderinti gimine, skaičiumi, linksniu. (Pav. 7)



Pav. 7 Paieškos sistema naudojama natūrali kalba

Norint natūralia kalba užduodamus klausimus transformuoti į SPARQL, reikia užklausas apdoroti programiškai. Tam, kad vartotojui palengvinti paiešką reikalingas galimų žodžių reikšmių sufleris. Ontologijoje tas pats žodis gali būti pavadintas sinonimine reikšme, todėl vartotojui iš sąrašo bus lengviau išsirinkti reikalingą reikšmę. Sufleruojamoji sakinio struktūra turi būti tinkamo linksnio, kad vartotojas suprastų kokių reikšmių galima tikėtis. Šiam sufleriui sudaryti reikalingos papildomos lietuvių kalbos morfologinės žinios.

2.2.5. Sakinio morfologija

Norint sudaryti natūralios kalbos suflerį reikia sistemai suteikti dirbtinio intelekto pažymį. Sistema turi linksniuoti ontologijoje saugomus vardus pagal sakinio struktūra, tokiu atveju, užklausas sistemai galėsime pateikti taisyklingą NLK, tam reikalingos morfologinės, kalbos dalių ypatumų žinios.

Morfologija yra gramatikos šaka, nagrinėjanti kalbos dalis (žodžių klases) ir jų sudėtį –

kaitybą, darybą.

Žodis – kalbėjimo ar teksto dalis, suvokiama kaip atskira nuo to paties lygmens kitos dalies ir rašoma atskirai. Pagal sudėtį žodžiai skirstomi į skaidžiuosius ir neskaidžiuosius. Skaidieji turi daugiau negu vieną reikšminę dalį, o neskaidieji reikšminių dalių neturi.

Žodį sudaro:

- Šaknis – pagrindinė skaidžiųjų žodžių dalis, prisijungianti afiksus ir galinti susijungti su kita šaknimi.
- Priesaga – žodžio dalis esanti tarp šaknies ir galūnės.
- Priešdėlis – žodžio dalis esanti prieš šaknį ir keičianti jos reikšmę.
- Galūnė – kintamoji žodžio dalis, rodanti jo ryšį su kitais žodžiais
- Kamienas – žodžio dalis be galūnės.

Žodžiai skirstomi kalbos dalimis: daiktavardis, būdvardis, skaitvardis, įvardis, veiksmažodis, prieveiksmis, dalelytė, prielinksnis, jungtukas, jaustukas, ištiktukas. (žiūr [9] psl.77-82)

Daiktavardis – savarankiška kalbos dalis, kurią sudaro žodžiai, žymintys daiktų, reikšmių, veiksmų ir ypatybių pavadinimus ir turintys savarankiškas giminės, skaičiaus ir linksnio kategorijas. (žiūr [9] psl.82)

Būdvardis – kalbos dalis, reiškianti ypatybę. Būdvardis derinamas su daiktavardžiu gimine, skaičiumi ir linksniu. (žiūr [9] psl.115)

Skaitvardis – kalbos dalis, reiškianti daiktų skaičių arba tikslią jų vietą erdvėje. Nuo kitų kiekybę reiškiančių žodžių skiriasi tuo, kad nurodo tikslų skaičiuojamų daiktų kiekį. (žiūr [9] psl.135)

Veiksmažodis- kalbos dalis reiškianti veiksmą. (žiūr [9] psl.150)

Prieveiksmis – savarankiška, nekaitoma kalbos dalis, reiškianti veiksmo aplinkybę ar ypatybės ypatybę. (žiūr [9] psl.185)

Dalelytė – tarnybiniai nekaitomi žodžiai, teikiantys kitiems žodžiams, žodžių junginiams ir sakiniams papildomų reikšmės atspalvių. (žiūr [9] psl.190)

Prielinksnis – tarnybinė nekaitoma kalbos dalis, esanti su linksniu ir rodanti jo ryšį su pagrindiniu jungiamuoju žodžiu. (žiūr [9] psl.194)

Jungtukas – tarnybinė kalbos dalis, rodanti žodžių ir sakinių ryšius. (žiūr [9] psl.198)

Įvardis – kalbos dalis, turinti bendrą rodomąją reikšmę. (žiūr [9] psl.142)

Jaustukas – žodžiai, rodantys žmogaus emocines reakcijas, valios aktus ar paskatas, pvz., oi! ach! ak!. Ši kalbos dalis mūsų sistemoje nebus naudojama, todėl plačiau jos nenagrinėsime. (žiūr [9] psl.202)

Ištiktukas – nekaitomi emociniai žodžiai, žymintys jausmus bei vaizdinius, sukeltus įvairių

aplinkos garsų, veiksmų ir pojūčių, pvz., ur, ur-r!. Ši kalbos dalis mūsų sistemoje nebus naudojama, todėl plačiau jos nenagrinėsime. (žiūr [9] psl.205)

Išnagrinėjus ontologijos sudarymą, galime pastebėti, kad klasių bei klasių savybių pavadinimams įvardinti yra naudojami daiktavardžiai; klasės savybėse gali būti naudojami daiktavardžiai, būdvardžiai bei skaitvardžiai; ryšiams tarp klasių aprašymui dažniausiai naudojami veiksmažodis. Iš šių pastebėjimų galime daryti išvadas, kad mums ontologijos linksniavimo sudarymui užtenka turėti sulinksniuotus daiktavardžius, būdvardžius, skaitvardžius.

Ontologijų elementų įvardinimo taisyklės yra pritaikytos anglų kalbai. Ontologijų metamodelyje numatyta struktūra, kurioje būtų nurodoma koks linksnis turi būti naudojamas, todėl reikia ieškoti sprendimo. Veiksmažodžiai pasižymi tuo, kad valdo daiktavardžius ir būdvardžius, t.y. po veiksmažodžio einančio žodžio linksnis priklauso nuo veiksmažodžiu ir perteikiamos žodžiais prasmės. Tačiau pagal veiksmažodį negalime nustatyti jo reikalaujamo linksnio, nes tam tikri veiksmažodžiai valdo ne vieną konkretų, o kelis linksnius. Tai suprasdami reikia rasti būdą kaip **ontologijoje nurodyti linksnį, kurio reikalauja veiksmažodinis klasių ryšys, jei ontologijos elementų vardai yra lietuviški**. Pavyzdžiui, *komplektuoja_ką* – šis ryšys su savimi neša linksnio informaciją - „ką“ yra galininkas, t.y. sekantis žodis bus galininko linksnyje (pvz: Rasti komplekto tipą, kuris komplektuoja modelį). Kitus linksnius galime nustatyti pagal komponento tipą. Užklausos formavimo algoritmas turi atrodyti taip:

1. Įvestas žodis „Rasti“, tokiu atveju išrenkami visų klasių pavadinimai sulinksniuojami kilmininko bei galininko linksniais. Jei vartotojas pasirenka kilmininko linksnį vykdoma 2 taisyklė, jei pasirenka galininko linksnį vykdoma 3 taisyklė.
2. Išrenkamos klasės savybės ir sulinksniuojamos galininko linksniu. Kai vartotojas pasirenka arba įvedą norimą klasės savybę vykdoma 3 taisyklė.
3. Išrenkami visi ryšiai jungiantys paskutinę klasę su kitomis klasėmis, iš ryšio pavadinimo pašalinama linksnio informacija, prieš ryšį pridedamas įvardis *kuris* ir vartotojui pateikiama galimų ryšių aibė pasirinkti. Kai vartotojas pasirenka norimą ryšį vykdoma 4 taisyklė.
4. Išrenkamos visos klasės, kurios siejasi su paskutiniu pasirinktu ryšiu ir klasių pavadinimai linksniuojami pagal pasirinkto ryšio linksnio informaciją. Kai vartotojas pasirenka norimą klasę vykdoma 5 taisyklė.
5. Išrenkamos paskutinės pasirinktos klasės esybės vardininko linksniu, taip pat išrenkamos visos klasės savybės ir pateikiamos vardininko linksniu ir žodžio pradžioje pridedamas įvardis *kurio*, paraleliai vykdoma 3 taisyklė. Jei vartotojas pasirenka klasės esybę vykdoma 3 taisyklė. Jei vartotojas pasirenka savybę vykdoma 6 taisyklė.
6. Išrenkamos visos paskutinio pasirinkto elemento esybės vardininko linksniu, pasirinkus esybę, vykdoma 3 taisyklė.

Šio algoritmo ir linksniavimo serviso pagalbą galime realizuoti natūralios kalbos užklausos pateikimą paieškos sistemai.

2.2.6. Natūralios kalbos vertimas į formalią

NLK vertimas į formalią yra tarpinis variantas, kuriuo metu yra išnagrinėjamas ir konvertuojamas į formalią užklausų kalbą. Esant sakiniui parašytam NK, jis išnagrinėjamas pagal ontologijos sudėtį ir suprantant ko sakinyje yra ieškoma, kokie kriterijai yra nurodomi.

Tarkime turime sakinį:

Rasti važiuoklės amortizatoriaus tipą, kuris sudaro komplektą Golf 6 Trendline

Iš ontologijos (PRIEDAS 9.1.) matome kad yra ieškoma *Važiuoklė* klasės savybė *Amortizatorių_tipas*, kuris siejasi ryšiu *kas_sudaro_ką* su klase *Komplektacija*, kuri turi elementą *Golf_6_Trendline*.

Formuojant NK užklausą duomenų struktūroje turi būti saugomas ir formalus užklausos aprašas, Formalaus aprašo struktūra:

Masyvo pirmajame elemente nurodomas ieškomas kintamasis, jis gali būti „?x“ – jei ieškome klasės esybės arba „?g“ jei ieškome klasės savybės reikšmės. Nuo antrojo masyvo elemento pradedamas užklausos formavimas, antrojo elemento (ieškomos klasės) aprašas „?x rdf:type Auto:xxx.“, xxx – klasės pavadinimas. Jei ieškomos klasės savybė formalios užklausos elemento struktūra atrodys: „?x Auto:xxx ?g.“, xxx – savybės pavadinimas. Paieškos kriterijus įvedant naudojamos struktūros:

- „?x Auto:xxx ?OPid“ – pasirinktas ryšys, xxx- ryšio pavadinimas, id – objekto eilės numeris;
- „?OPid rdf:type Auto:xxx.“- pasirinkus klasę, xxx – klasės pavadinimas, id - paskutinio objekto eilės numeris;
- „1*2*\"xxx\".“ – pasirinkta klasės savybės esybė, pirmas parametras reiškia kad bus grįžta vienu objektu atgal, antras parametras reiškia, kad bus keičiamas antras paieškos elementas, trečias parametras , kad antrame parametre nurodytas elementas bus keičiamas į klasės savybės esybės pavadinimą.
- „2*2*Auto:xxx.“ – pasirinkta klasės esybė, pirmas parametras reiškia kad bus grįžta dviem objektais atgal, antras parametras reiškia, kad bus keičiamas antras paieškos elementas, trečias parametras reiškia, kad antrame parametre nurodytas elementas bus keičiamas į klasės esybės pavadinimą.

Formalus aprašo pavyzdys:

- *„Rasti“* - "?g";
- „Važiuoklė“ – „?x rdf:type Auto:Važiuoklė.“;
- „Amortizatorių_tipas“ – „?x Auto: Amortizatorių_tipas ?g.“;
- „„ kuris“ – „“;
- „kas_sudaro_ką“ – „?x Auto:kas_sudaro_ką ?OP3 .“;
- „Komplektas“ – „?OP3 rdf:type Auto:Komplektas.“;
- „Golf_6_Trendline“ – „2*2*Auto:Golf_6_Trendline.“ (parametrų reikšmės: kelintas objektas nuo pabaigos yra redaguojamas * kelintas užklausos kintamasis yra keičiamas (1 - pirmas, 2 – antras) * į kokią reikšmę keičiamas);

Pagal šį struktūrinį aprašą NLK užklausa transformuojamą į formalią SPARQL užklausą, kuri atrodo:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Auto: <http://www.semanticweb.org/ontologies/2012/0/Auto.owl#>
select distinct ?g
        where{
        ?x rdf:type Auto:Važiuoklė.
        ?x Auto:Važiuoklė_Amortizatorių_tipas ?g.
        ?x Auto:kas_sudaro_ką Auto:Golf_6_Trendline.
         }
```

Įvedamos užklausos struktūrinis aprašymas bus saugomas sudarant NLK užklausą. Įvedant kiekvieną naują žodį jis yra išanalizuojamas pagal ontologiją ir viso sakinio struktūros aprašas bus saugomas masyve. Sudarytas sakinio aprašo masyvas padės lengviau išnagrinėti įvestą užklausą ir konvertuoti į formalia SPARQL kalbą.

2.2.7. Esami sprendimai

2.2.7.1. Ontologijos redaktoriai

Dalykinės srities ontologijai sudaryti reikalingas ontologijos redaktorius. Yra žinomi virš 50 ontologijų redaktorių, tokių kaip Protege, OntoEdit, TopBraid Composer, JOE (Java Ontology Editor) ir kt.[17] Patirties dirbant su ontologijų redaktoriais neturime, visų redaktorių išnagrinėti nėra galimybių, todėl pagrindinis aspektas redaktoriaus pasirinkime buvo jo populiarumas, tyrimų šią tema nėra, tačiau noriu sulyginti ir suprasti kokiais iš jų žmonės dažniausiai naudojasi.

Šiuo atveju geriausias būdas sužinoti jų populiarumą- pasinaudoti paieškos sistemomis ir jų

galimybėmis kiekvienai užklausai atvaizduoti rastų puslapių kiekį, šiais kiekiais naudojantis galime daryti išvadas apie palyginama redaktorių populiarumą. Taip pat kaip būda galime naudoti paieškos statistikas, tačiau šiuo atveju sunku išsiaiškinti populiarumą, nes pirmiausia daugelis vartotojų gali skirtingai ieškoti redaktorių ir formuluoti užklausas, taip pat pagal užklausas (pvz: „Protege“) gali būti ieškomas ne ontologijų redaktorius.

Tyrimui buvo naudotas Vikipedijoje pateiktu ontologijų redaktorių sąrašas [17]. Kiekvienam redaktoriui buvo sudaryta užklausa: *„<redaktoriaus pavadinimas>“ ontology editor*.

Ontologijų redaktorių nuorodų internete pakartojamumas (2012.03.01):

	Ontologijos redaktorius	Paieškos rezultatai
1.	Protege	110 000
	Protégé	74 600
2.	DOME	54 500
3.	OBO-Edit	42 000
4.	Altova SemanticWorks	17 200
5.	SWOOP	19 800
6.	Ontolingua	12 100
7.	KAON	9 030
8.	TopBraid	6 440
	TopBraid Composer	4 950
9.	Knoodl	7 160
10.	WebODE	3 840
11.	CMapTools	2 570
12.	Model Futures OWL Editor	2 460
13.	Synaptica	2 380
14.	OntoSaurus	2 070
15.	JOE (Java Ontology Editor)	786

Iš šių duomenų tiesiogiai negalime daryti tiesioginių išvadų apie redaktorių populiarumą šiuo metu. Pavyzdžiui KAON yra vienas iš lyderiu, tai susiję su tuo, kad KAON OIModeler buvo vienas iš pirmųjų vizualinių redaktorių 2004 metais, kai gerų priemonių kuriant ontologijas buvo mažai. 2005 metais buvo nutrauktas jo palaikymas ir plėtojimas. Tai reiškia, kad pagal paminėjimą sudėtinga daryti išvadas apie šiandienį įrankio populiarumą.

Pagal gautus paieškos rezultatus matome, kad populiariausias ontologijos redaktorius yra Protégé. Su toliau einančiu populiarumu nelabai niekas neaišku, man atrodo, kad toliau turi eiti grupė

su kitais dviem gerais komerciniais redaktoriais TopBraid Composer ir Altova SemanticWorks.

Protege – atviro kodo ontologijų kūrimo įrankis, žinių bazių kūrimui. Programa parašyta Java kalba. Su Protege sukurtos ontologijos gali būto eksportuojamos į daugelį formatų, tokių kaip RDF, OWL, XML Schema.

Protege turi atvirą, įskiepių pagalba lengvai išplečiamą programos funkcionalumą.

Protege yra palaikomas didelės bendruomenės, kurią sudaro kūrėjai, mokslininkai, valstybinių bei verslo įstaigų atstovai, naudojantys uždavinių, susijusių su žiniomis, sprendimus, tokiose srityse kaip biomedicina, duomenų surinkimas bei įmonių modeliavimas.[7] Šio įrankio pagalbą buvo sudarytas automobilių ontologijos prototipas, pateiktas PRIEDAS 9.1.

2.2.7.2. Užklausų vykdymo priemonės

Renkantis įrankius pagrindinis dėmėsis buvo skiriamas įrankiams palaikomiems .Net aplinkoje, į kurį galima kreiptis C# kalbos pagalba. Tačiau .Net aplinkai yra sukurta nedidelis kiekis išbaigtų projektų, tačiau Java kalbai yra sukurti Joseki, Sesame, Virtuoso ir kt., todėl buvo nuspręsta naudoti Java kalbai skirtus įrankius, kreipiantis į juos per dotNetRDF biblioteką.

dotNetRDF – yra atviro kodo projektas. Šį biblioteka pradėrta kurti 2009 metais ir aktyviai plėtojama, pasižymi gera dokumentacija. Ši .net biblioteka naudoja naujausią .Net karkaso (angl. *framework*) versiją, jį yra galinga ir lengvai naudojama darbui su semantinėmis technologijomis. Ši biblioteka palaiko Joseki, Sesame, Virtuoso prieigos taškus.[18]

Joseki – pakankamai paprasta serverio realizacija, kuri pateikia galimybę prieiti prie RDF duomenų SPARQL užklausų pagalba, duomenys grąžinami RDF/XML formatu. Darbui su RDF, SPARQL naudosime Joseki prieigos tašką. Jis leidžia greitai konfigūruoti servisus, kuriems galima siųsti užklausas SPARQL kalba. Joseki nėra nei W3C standartas, nei rekomendacija, tačiau Joseki kūrėjai priklauso pasaulinio internetinio tinklo kūrėjų konsorciumui (W3C) ir aktyviai dalyvauja SPARQL tobulinime, todėl tikėtina, kad ateityje gali atsirasti rekomendacija analogiška Joseki protokolui. [8]

2.2.7.3. Žodžių kaitymo sprendimai

Žodžių linksniavimo tematika yra sukurti du prototipai. Vienas iš jų yra morfologinis lietuvių kalbos žodynas, kitas – tinklo servisas skirtas žodžių linksniavimui.

Vytauto Zinkevičiaus yra sukurtas morfologinis lietuvių kalbos žodynas [10]. Šiame žodyne yra apibūdinami žodžiai pagal to žodžio kalbos dalies ypatybes, nusako kokios taisyklės gali būti taikomos linksniuojant. Šiuo žodynu galima naudotis nagrinėjant žodžio sudėtį, linksniuojant norimus žodžius. Šis žodynas yra nefunkcionalus, nes faile yra aprašomos tik žodžių linksniavimo taisyklių logika. Tam kad galėtumėm linksniuoti žodžius papildomai mums reikia sukurti servisą kuriame būtų aprašytos visų kalbos dalių taisyklės ir naudodamiesi morfologinis lietuvių kalbos

žodyno faile esančiais žodžių linksniuoti taisyklių aprašus pritaikyti tam tikram žodžiui būdingas taisykles. Šis linksniavimo sprendimas mums nėra tinkamas, nes neturime pakankamai resursų naujos linksniavimo sistemos projektavimui bei kūrimui.

Kitas linksniavimo realizavimo būdas yra naudojantis studento sukurtu tinklo servisu, kuris yra prieinamas http://linksniavimas.virsmas.lt/linksniai.json adresu (prisijungimo vardas: wsktu; slaptažodis: lns_test1). Šis servisas yra pakankamai patogus, nes prieinamas per tinklą. Testuojant šį servisą jam buvo pateikiami žodžiai ir vertinamas gautas rezultatas. Buvo linksniuoti:

daiktavardis - automobilis ;

būdvardis – juodas;

skaitvardis – vienas;

veiksmažodis – važiuoja.

Servisui pateikiama užklausa :

http://linksniavimas.virsmas.lt/linksniai.json?formos=automobilis,juodas,vienas,važiuoja

Gauti rezultatai:


```
[{
        "v" : "automobilis",
        "k" : "automobilio",
        "n" : "automobiliui",
        "g" : "automobilį",
        "įn" : "automobiliu",
        "vt" : "automobilyje",
        "š" : "automobili",
        "forma" : "automobilis"
    },
    {
        "v" : "juodos",
        "k" : "juodų",
        "n" : "juodoms",
        "g" : "juodas",
        "įn" : "juodomis",
        "vt" : "juodose",
        "š" : "juodos",
        "forma" : "juodas"
    },
    {
        "v" : "vienos",
        "k" : "vienų",
        "n" : "vienoms",
        "g" : "vienas",
        "įn" : "vienomis",
        "vt" : "vienose",
        "š" : "vienos",
        "forma" : "vienas"
    },
    {
        "I asm." : "važiuojame",
        "II asm." : "važiuojate",
        "III asm." : "važiuoja",
        "forma" : "važiuoja"
    }]
```

Gavus sulinksniuotus rezultatus, buvo pastebėta, kad sulinksniuoto žodžio giminė ne visada atitinką pradinio žodžio giminę (pvz. pradinė forma juodas (vyr.g. vns. Vard.), gautas rezultatas: „juodos“ (mot. g. vns. vard)). Tai nėra kritinė klaida, gauti rezultatai tenkina, todėl šį servisą naudosime žodžių linksniavimui.

2.2.7.4. Esami NLK paieškos analogai

Šiuo metu rinkoje nėra įrankio, kurio pagalba galėtumėme internetiniame tinkle atlikti intelektualia paiešką naudojant NLK. Tokie įrankiai yra sukurti anglų kalbai. Anglų kalbos sakiniai yra pakankamai struktūrizuoti, jie neturi linksnių. Dėl šių priežasčių anglų kalbos įrankiai lietuvių kalbai nepritaikomi.

2.2.8. Vartotojo analizė

Mūsų kuriamos sistemos vartotojai bus žmonės, kurie norės rašyti užklausas natūralia lietuvių kalba ir gauti teisingus atsakymus į juos dominančius klausimus. Šie žmonės bus morfologinio nagrinėjimo posistemės tiesioginiai vartotojai.

2.3. Siekiamas sprendimas

Siekiama sukurti universalų užklausų variklį, kuris leistų formuoti užklausas natūralia lietuvių kalba. Vartotojas naudodamasis šia sistema viename laukelyje galės formuluoti užklausas suflerio pagalba, kuris siūlys užklausos formuluotės žodžius paremtus iš ontologijos išriktų galimų elementų pavadinimais. Sudarant užklausą NLK siekiama įvertinti šiuos esminius kriterijus: užklausos taisyklingumą, tikslumą, intuityvumą, universalumą. Suformuluota užklausa bus transformuojama į formalią SPARQL kalbą ir įvykdžius šią užklausą pateikiami gauti rezultatai, bei pačios SPARQL užklausos variantas.

2.4. Išvados

1. Išanalizavus ontologijų užrašymo kalbas ir jų sudarymo taisykles nustatyta, kad nėra aiškiai apibrėžta kaip nurodyti linksnius, kurių reikalauja lietuvių kalbos žodžių kaitymo taisyklės.
2. Išanalizavus užklausų įvedimo būdus buvo nustatyta, kad užklausos gali būti pateikiamos trimis tipais, iš jų išrinktas natūralios kalbos užklausų įvedimo būdas, užklausą formuluojant vienu sakiniu. (Pav. 7)
3. Išanalizavus lietuvių kalbos klausimų formulavimo struktūrą, buvo prieita išvados, kad liepiamojo klausimo struktūra yra artima SPARQL užklausų kalbai. Dėl šios priežasties buvo pasirinktas liepiamojo klausimo stilius, o ne klausiamojo klausimo tipas.
4. Išanalizuota kaip galima konvertuoti NLK parašytą sakinį į formalia SPARQL kalbą.
5. Išnagrinėjus egzistuojančius linksniavimo sprendimus iš ontologijos išrinktų žodžių linksniavimui buvo pasirinktas studentų sukurtas tinklo servisas. Iš testavus šį servisą buvo gauti ne visada tikėtinos reikšmės, tačiau nesutapimai nėra kritinės klaidos, gauti rezultatai tenkina.
6. Šiuo metu rinkoje analogišku sprendimų nėra, o šį sistema yra reikalinga plečiant semantinį tinklą, todėl reikia sukurti šios sistemos prototipą.
7. Sistemos ontologijai kurti buvo pasirinktas atviro kodo įrankis – Protege. Dėl paprastumo ir pastovaus tobulinimo SPARQL prieigos tašku buvo pasirinktas Joseki. Priėjimui prie Java kalbai skirto Joseki įrankio pasirinktas dotNetRDF biblioteka, dėl geros dokumentacijos ir pastovaus plėtojimo.

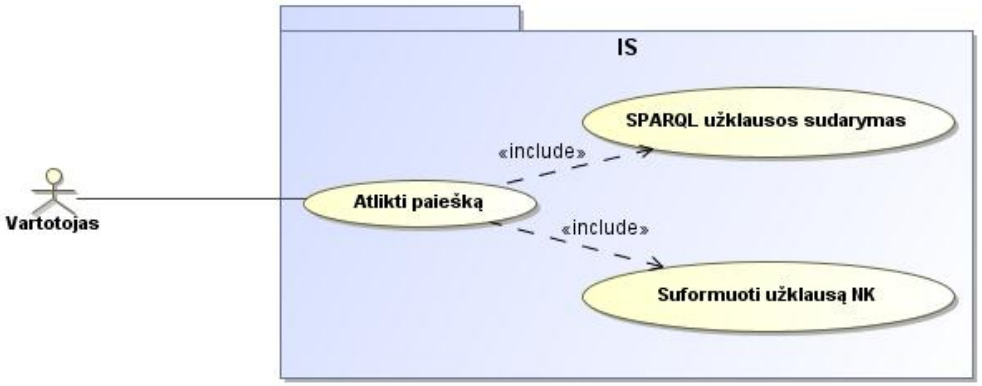
3. SISTEMOS REIKALAVIMAI

3.1. Reikalavimų specifikacija

3.1.1. Funkciniai reikalavimai

3.1.1.1. Kompiuterizuojamos sistemos funkcijos (panaudojimo atvejų diagrama)

Detalizuojant 2.3. poskyryje pateiktą metodo idėją sudaryti panaudojimo atvejai (8 pav.), kuriuos turi apimti kuriama sistema. Panaudojimo atvejų specifikacijos pateiktos 1 – 3 lentelėse ir 9 – 10 paveikslėliuose.



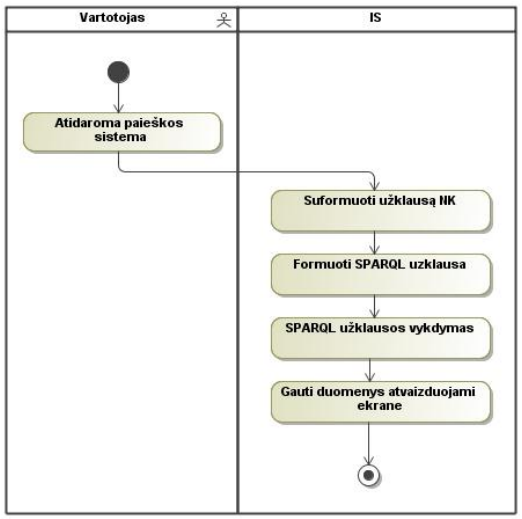
Pav. 8 Vartotojo panaudojimo atvejų diagrama

3.1.2. Panaudojimo atvejų detalizavimas

Paveikslėlyje 9 pavaizduota „Atlikti paiešką“ veiklos diagrama ir 1 lentelėje aprašoma panaudojimo atvejo detali specifikacija. Įvykdžius šiuos veiksmus vartotojui pateikiami paieškos metu išrinkti duomenys.

1 lentelė. Panaudojimo atvejo „Atlikti paiešką“ specifikacija

1 PA „Atlikti paiešką“		
Aprašymas		Tai pagrindinis sistemos panaudojimo atvejis. Vartotojas suflerio pasiūlymų pagalba įveda norimą užklausą ir gauna norimus duomenis.
Aktoriai		Sistemos vartotojas.
Prieš sąlyga		Vartotojas turi būti prisijungęs prie sistemos
Sužadinimo sąlyga		Atsiradęs poreikis atlikti duomenų paiešką ontologijoje.
Susiję panaudojimo atvejai	Išplečia PA	-
	Apima PA	Suformuoti užklausą NK; SPARQL užklausos sudarymas.
	Specializuoja PA	-
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas atidaro paieškos sistemą 2. Baigiamas PA.		2.1. Sistema padeda suformuoti NK užklausą 2.2. Sistema apdorodama NLK formavimo metu pasirinktus duomenis ir suformuoja SPARQL užklausą. 2.3. Sistema išrenka duomenis iš ontologijos. 2.4. Sistema atvaizduoja surinktus duomenis ekrane.
Po sąlyga		Ekrane atvaizduojami surasti duomenys

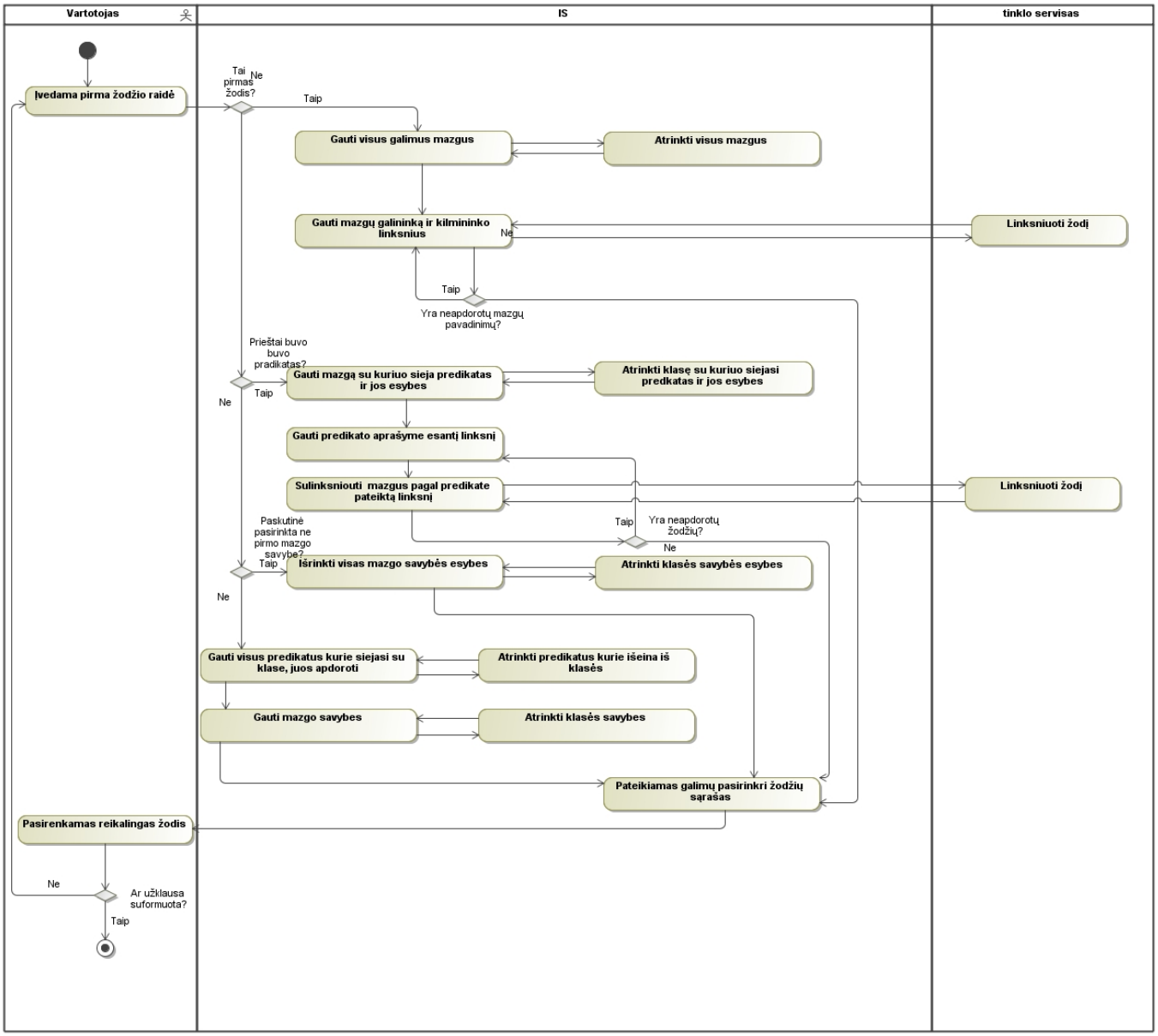


Pav. 9 Panaudojimo atvejo „Atlikti paiešką“ veiklų diagrama

10 paveikslėlyje pavaizduota „Suformuoti užklausą NK“ panaudojimo atvejo veiklų diagrama ir 2 lentelėje aprašoma panaudojimo atvejo detali specifikacija. Įvykdžius šiuos veiksmus vartotojui pateikiamas galimų NLK užklausų pasirinkimų sąrašas.

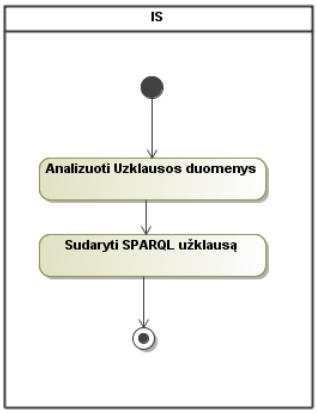
2 lentelė. Panaudojimo atvejo „Suformuoti užklausą NK“ specifikacija

2 PA „Suformuoti užklausą NK“		
Aprašymas		Šis PA skirtas sufleriui, bei užklausos sudarymui reikalingų duomenų išrinkimui iš ontologijos. Šis PA duoda vartotojui galimybę paieškos sistemoje formuoti užklausą natūralia lietuvių kalba.
Aktoriai		Sistemos vartotojas.
Prieš sąlyga		Sistemos vartotojas paieškos lauke įveda simbolį.
Sužadinimo sąlyga		Atsiradęs poreikis sudaryti užklausą NLK.
Susiję panaudojimo atvejai	Išplečia PA	-
	Apima PA	-
	Specializuoja PA	-
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda pirmą žodžio raidę. 2. Vartotojas pasirenka (įveda) reikalingo žodžio reikšmę. 3. Baigiamas PA.		1.1. Sistema tikrina, jei įvedamas pirmas paieškos žodis, iš ontologijos ištrenkami visi mazgai, jie sulinksniuojami kilmininko bei linksniais tinklo serviso pagalba. Pereinama į 2 žingsnį. Priešingu atveju pereinama į 1.2. žingsnį. 1.2. Sistema tikrina ar prieš tai pasirinktas predikatas. Jei pasirinktas predikatas, iš ontologijos išrenkamas objektas, su kuriuo siejasi pasirinktas predikatas ir jo esybės. Išrinktos reikšmės sulinksniuojamos pagal predikate nurodytą linksnį. Pereinama į 2 žingsnį. Priešingu atveju pereinama į 1.3. žingsnį 1.3. Tikrina jei prieš tai pasirinkta ne pirmo objekto savybė ištenkamos visos objekto savybės esybės. Pereinama į 2 žingsnį. Priešingu atveju pereinama į 1.3. žingsnį 1.4. Sistema išrenka predikatus, kurie siejasi su subjektu. Nuo predikato nukerpame pradžioje bei gale nurodytos linksnio žymės. Pereinama į 2 žingsnį.
Po sąlyga		Vartotojui pateikiamas galimų NLK užklausų pasirinkimų sąrašas



Pav. 10 Panaudojimo atvejo „Suformuoti užklausą NK" veiklų diagrama

11 paveikslėlyje yra specifikuojamas „Suformuoti SPARQL užklausą" veiklų diagrama ir 3 lentelėje panaudojimo atvejo detali specifikacija. Įvykdžius šiuos veiksmus grąžinama SPARQL užklausa



Pav. 11 Panaudojimo atvejo „Suformuoti SPARQL užklausą" veiklų diagrama

3 lentelė. Panaudojimo atvejo „Suformuoti SPARQL užklausą“ specifikacija

3 PA „Suformuoti SPARQL užklausą“		
Aprašymas		Duoti vartotojui galimybę paieškos sistemoje formuoti užklausą natūralia lietuvių kalba
Aktoriai		Sistemos vartotojas
Prieš sąlyga		Sistemos vartotojas paspaudžia paieškos mygtuką.
Sužadinimo sąlyga		Atsiradęs poreikis gauti atsakymą į sudarytą užklausą NLK.
Susiję panaudojimo atvejai	Išplečia PA	-
	Apima PA	-
	Specializuoja PA	-
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Sistema išanalizuoja sistemos vartotojo pasirinktus duomenys sudarant užklausą NLK. 2. Užklausos duomenys transformuojami į SPARQL užklausą. 3. Baigiamas PA.		
Po sąlyga		Grąžinama SPARQL užklausa

3.1.3. Nefunkciniai reikalavimai

1. **Reikalavimai patogumui**

Sistema turi būti lengvai pasiekiama vartotojui. Intuityvi, lengvai skaitoma, aiškiai atvaizduojami atsakymai į užklausą.

2. **Reikalavimai panaudojamumui**

Naudojimosi paprastumas. Sistema turi būti lengva naudotis visiems žmonėms, nepriklausomai kokios jų žinios IT srityje.

3. **Reikalavimai greitaveikai**

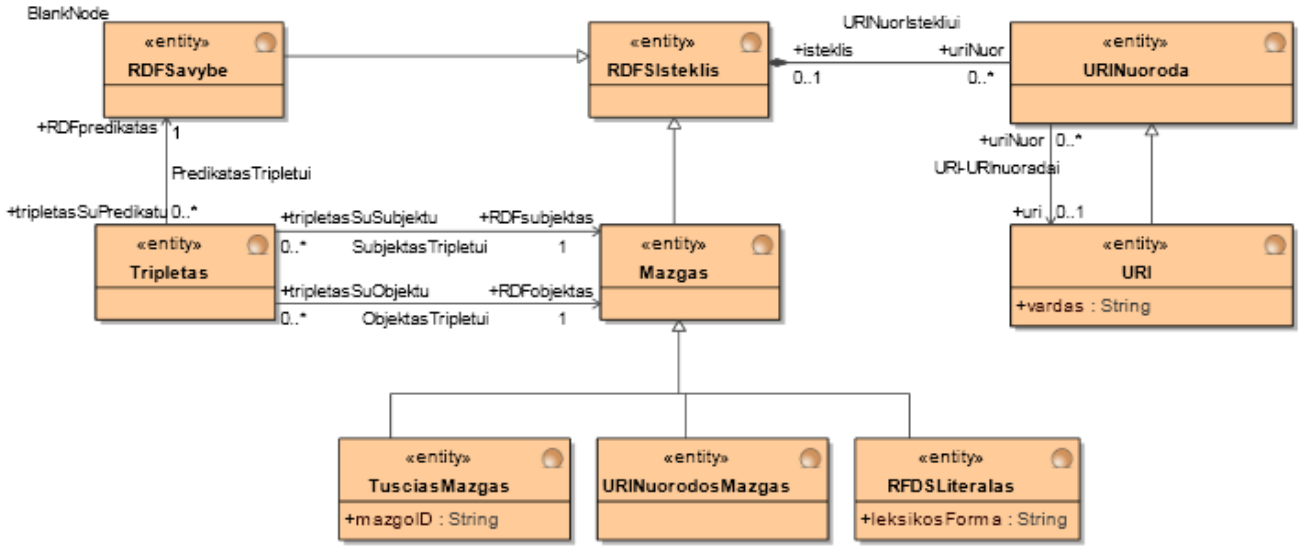
Užklausos sufleris turi pateikti galimus variantus ne ilgiau kaip per 3 – 4 sekundes. Į užklausą turi būti atsakyta ne ilgiau kaip per 5 - 10 sekundes.

4. **Reikalavimai plečiamumui**

Architektūra turi būti tokia, kad būtų galima pakeisti komponentų kiekį nekoreguojant visos sistemos.

3.2. Dalykinės srities modelis

Paveikslėlyje 12 pavaizduotas RDF duomenų grafo metamodelis [12]. Šio metamodelio pagalba ontologijoje yra aprašomos dalykinės srities esybės. Pagal šį metamodelį RDFS ištekliui priklauso visi jame esantys mazgai bei RDF savybes. RDFS išteklius gali būti globalus ir turėti URI nuorodą. Kiekvienas mazgas turi po vieną tripletą su objektą arba subjektą, kuris gali turėti RDF savybes. Mazgas gali turėti klasės tipą, URI tipą arba RDFS literalo tipą.



Pav. 12 RDF grafo metamodelis

3.3. Reikalavimų analizės apibendrinimas

Reikalavimų analizės metu konkretizuota sistemos taikymo sritis, detalizuoti funkciniai reikalavimai, išskirti nefunkciniai reikalavimai ir nustatytas dalykinės srities modelis. Pagrindinė sistemos funkcija yra leisti vartotojui atlikti paiešką ontologijoje NLK, tačiau jį negali būti vykdoma, jei prieš tai sistemai nebus suteikti reikalingi duomenys, NLK užklausos transformavimui į formalia SPARQL kalbą, kurie yra surenkami įvedant NLK užklausą. Siekiant naudojamos sistemos universalumo iškeltas nefunkcinis sistemos plečiamumo reikalavimas. Sistemos dalykinės srities modelis apima sistemos funkcionavimui reikalingos ontologijos duomenų struktūros sudarymą, saugojimą.

4. SISTEMOS PROJEKTAS

4.1.Sistemos architektūros projektas

4.1.1. Sistemos loginė architektūra

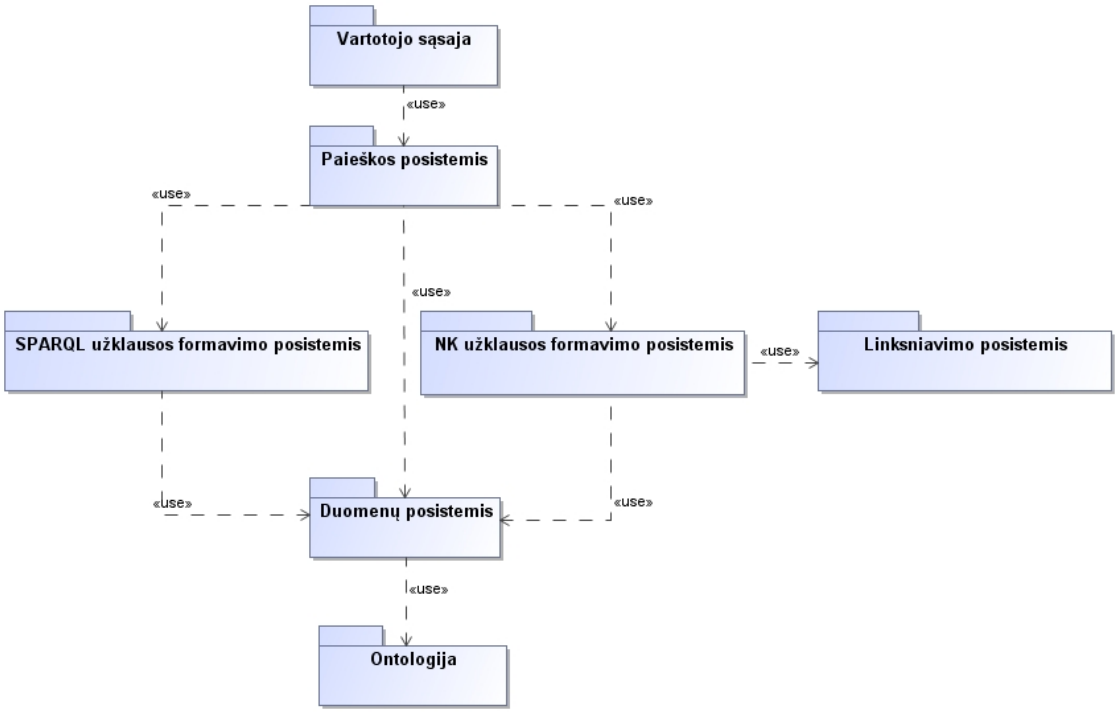
Kaip matoma paveikslėlyje 13 pateiktoje loginėje schemoje, sistemą sudaro vartotojo sąsajos posistemis, kuriame surenkami paieškai reikalingi duomenys.

Paieškos posistemyje yra aprašyta veikimo logika, ši posistemė naudoja NK užklausos formulavimo posistemį, SPARQL užklausos formavimo posistemį, duomenų posistemį.

NK užklausos formulavimo posistemėje iš ontologijos išrenkamos galimų duomenų mazgų, ryšių, savybių reikšmės. Kreipiantis į žodžio linksniavimo posistemį iš kurio gaunamos žodžio morfologinis aprašas, kuriame nurodoma kokiame linksnyje koks yra nurodytas žodis. NK užklausos formavimo posistemėje pagal sudaromą užklausą nustatoma kokio linksnio žodis reikalingas ir iš morfologinio aprašo išrenka reikalingą linksnį. Formuojant NK užklausą duomenų masyve yra saugomi duomenys reikalingi tolimesniam užklausos apdorojimui: pradinė žodžio forma, sulinksniuota žodžio forma, ontologijos objekto tipas, linksnis, užklausos elementas.

Inicijavus SPARQL užklausos formavimo posistemį paieškos posistemis pateikia NK užklausos formavimo metu sudaryto duomenų masyvo duomenis. SPARQL užklausos formavimo posistemyje yra apdorojamas paduotas duomenų masyvas ir grąžinama SPARQL užklausa.

Duomenų posistemis yra naudojamas sistemos bendravimui su ontologija.



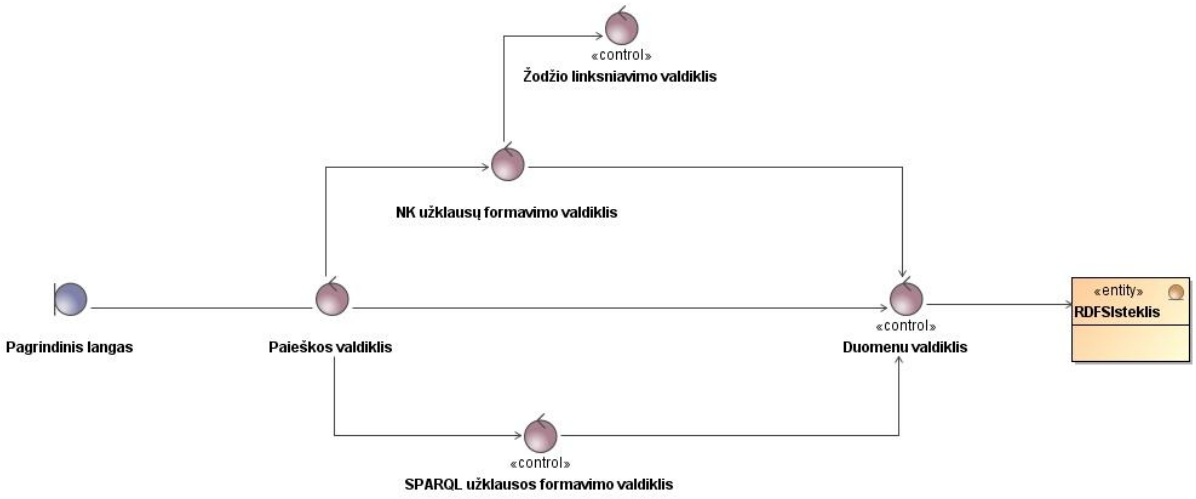
Pav. 13 Sistemos loginė architektūra

4.1.2. Vartotojo paslaugos

Vartotojo sąsają sudaro vienas grafinė sąsajos langas. Šiame grafinės sąsajos lange bus atliekami visi paieškai duomenų veiksmai. Vartotojas grafinės sąsajos lange vartotojas paieškos laukelyje paspaudus klaviatūros klavišą jam suflerio pagalba atvaizduojami iš NK užklausos sudarymo posistemio grąžinti duomenys, vartotojas pasirenką norimą reikšmę ir tai kartoja kol nesuformuojama visa reikalinga užklausa. Suformavus užklausą vartotojas pasirenka klavišą *Apdoroti*. Pasirinkus klavišą *Apdoroti* yra inicijuojama SPARQL užklausos valdiklis, kuris grąžina sudarytą SPARQL užklausą, kurią paduodame į duomenų posistemį ir kurio gražinamas rezultatas, bei SPARQL užklausa atvaizduojamas grafinės sąsajos lange.

4.1.3. Projekto klasių modelis

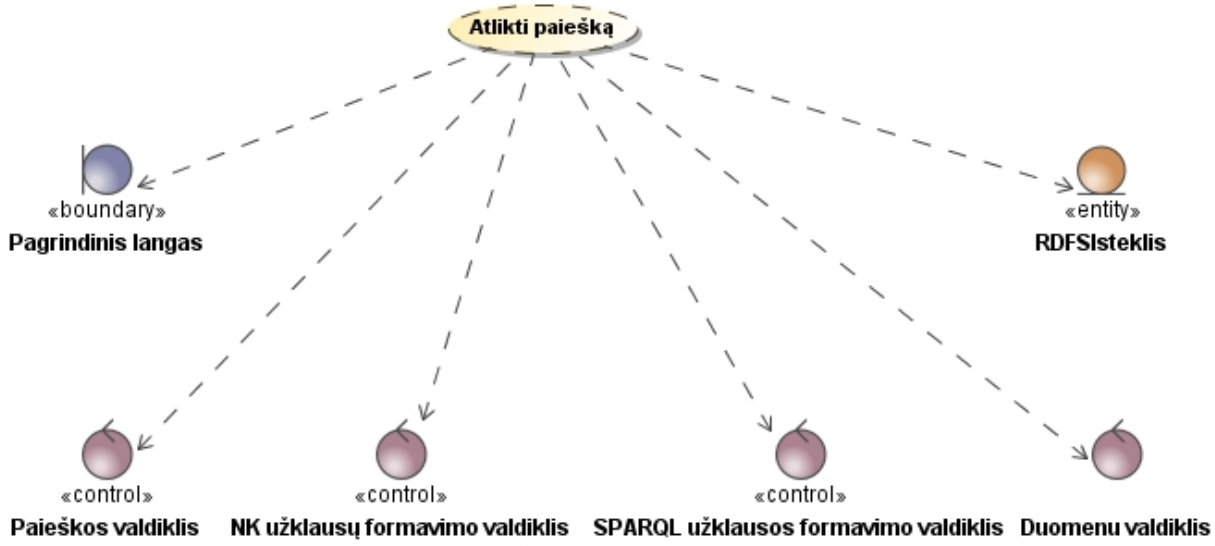
Paveikslėlyje 14 pavaizduotame modelyje yra atvaizduojama IS vaizdų, valdiklių ir esybių sudėtis. Kiekvienas valdiklis turi savo klasę, kurios skirtos atlikti tam tikrus veiksmus: paieškos klasė skirta paieškos veikimo logikai aprašyti; NK užklausų formulavimo klasė skirta NLK užklausos formulavimui; žodžių linksniavimo klasė skirta žodžių linksniavimui, linksnio nustatymui; duomenų klasė skirta SPARQL užklausų perdavimui į Joseki prieigos tašką; SPARQL užklausų formulavimo klasė skirta iš NLK sudaryto sakinio transformavimui į formalią SPARQL kalbą;



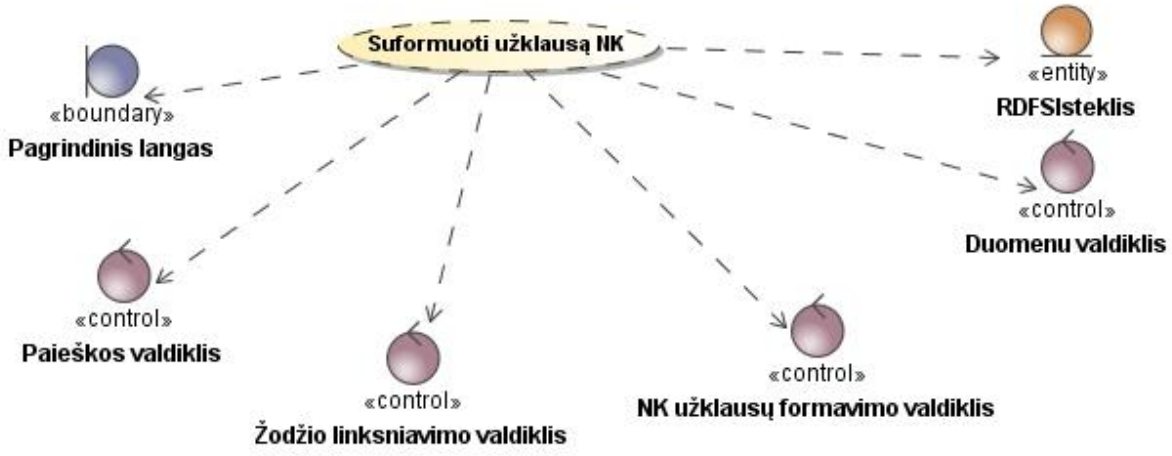
Pav. 14 Projekto klasių modelis

4.1.4. Panaudojimo atvejų realizacija projekto klasėmis

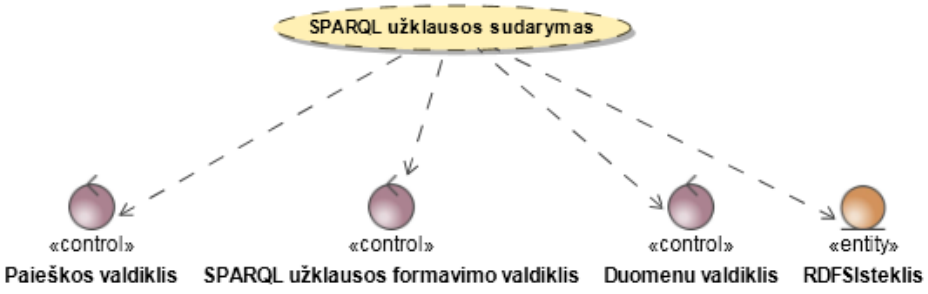
Projekto atvejų realizacija projekto klasėms diagramos vaizduoja su kokiomis esybėmis turi ryšį panaudojimo atvejai. Sekančiuose paveikslėliuose pavaizduota PA „Atlikti paiešką“ realizacija (Pav. 15), PA „Suformuoti užklausą NK“ (Pav. 16), PA „SPARQL užklausos sudarymas“ (Pav. 17).



Pav. 15 „Atlikti paiešką“ realizacija projekto klasėmis



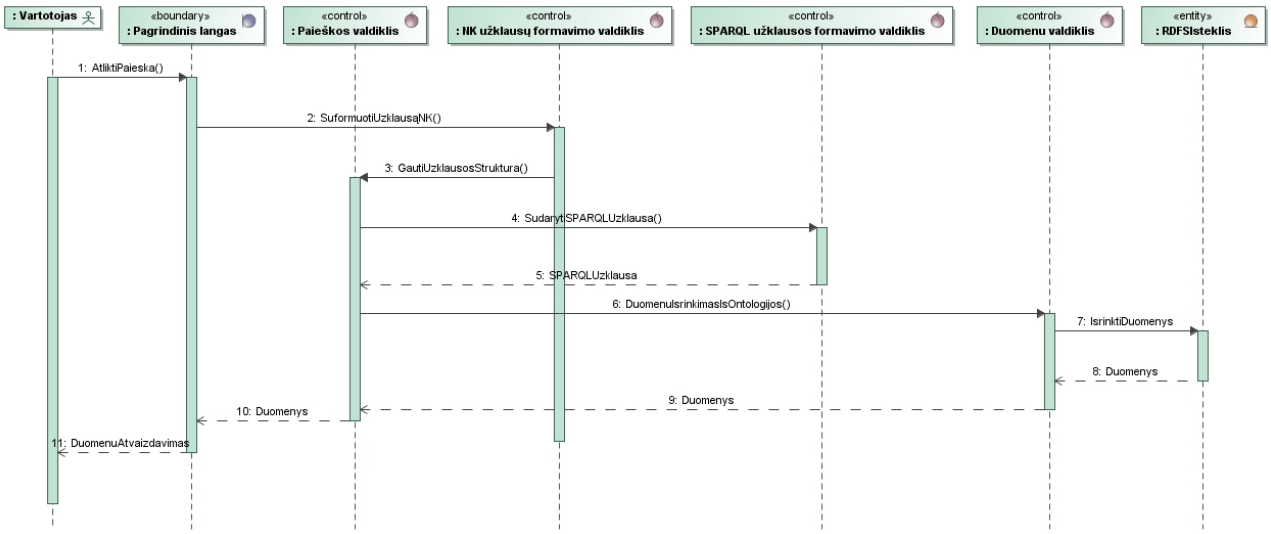
Pav. 16 „Suformuoti užklausą NK“ realizacija projekto klasėmis



Pav. 17 „SPARQL užklausos sudarymas“ realizacija projekto klasėmis

4.2.Panaudojimo atvejų sekų diagrama

18 paveikslėlyje pavaizduota paieškos atlikimo sekų diagrama, joje matome, jog vartotojas atsidaręs paieškos langą suformuoja užklausą NLK užklausą NK užklausų formavimo valdiklio pagalba. Tada iš NK užklausų formavimo valdiklio Paieškos valdiklis perduoda sudarytos užklausos struktūra aprašyta dvimačiame masyve į SPARQL užklausos formulavimo valdiklį, kuriame masyve aprašyta struktūra konvertuojama į SPARQL užklausą. Tada paieškos valdiklis įvykdo šią SPARQL užklausą ir grąžina rezultatus, kurie atvaizduojami pagrindiniame lange.



Pav. 18 „Atlikti paiešką“ sekų diagrama

19 paveikslėlyje pavaizduota užklausos formavimo NK sekų diagrama. Joje matome, kad iš pradžių, vartotojui paspaudus tarpo klavišą iš ontologijos išrenkamos visų klasių (mazgų) pavadinimai, kurie sulinksniuojami kilmininko ir galininko linksniais ir pateikiami vartotojui, taip pradedamas paieškomo objekto nustatymas. Iš pasirinkto žodžių sąrašo vartotojas pasirenka (įveda) norimą reikšmę, jei ši reikšmė (klasės pavadinimas) yra galininko linksnyje – ieškoma klasė, jei kilmininko linksnyje – vartotojas ieško klasės savybės, todėl ontologijos dar išrenkamos pasirinktos klasės visų savybių pavadinimai kurie sulinksniuojami galininko linksnyje, vartotojas pasirenka norimos surasti klasės savybės pavadinimą, taip baigiamas formuoti paieškomas objektas, tada į paieškos eilutę įrašoma reikšmė *kuris* ir pradedami formuoti apribojimai.

Apribojimai pradedami formuoti sklendžiant ontologijos predikatais. Iš ontologijos išrenkami visi ryšiai kurie išeina iš pasirinktos klasės, vartotojas pasirenka norimą ryšį, tada išrenkamos klasės su kuriomis jungiasi šis ryšys, jos sulinksniuojamos pagal predikate pateiktą linksnį, vartotojas vėl renkasi. Pasirinkus klasę iš ontologijos išrenkamos visos šios klasės esybės, klasės savybių pavadinimai, bei ryšiai kurie išeina iš šios klasės. Prieš klasės savybės pavadinimą pridedamas žodelis *kurio*, priešais ryšio pavadinimą pridedamas žodelis *kuris*, taip pat ryšyje nuimami

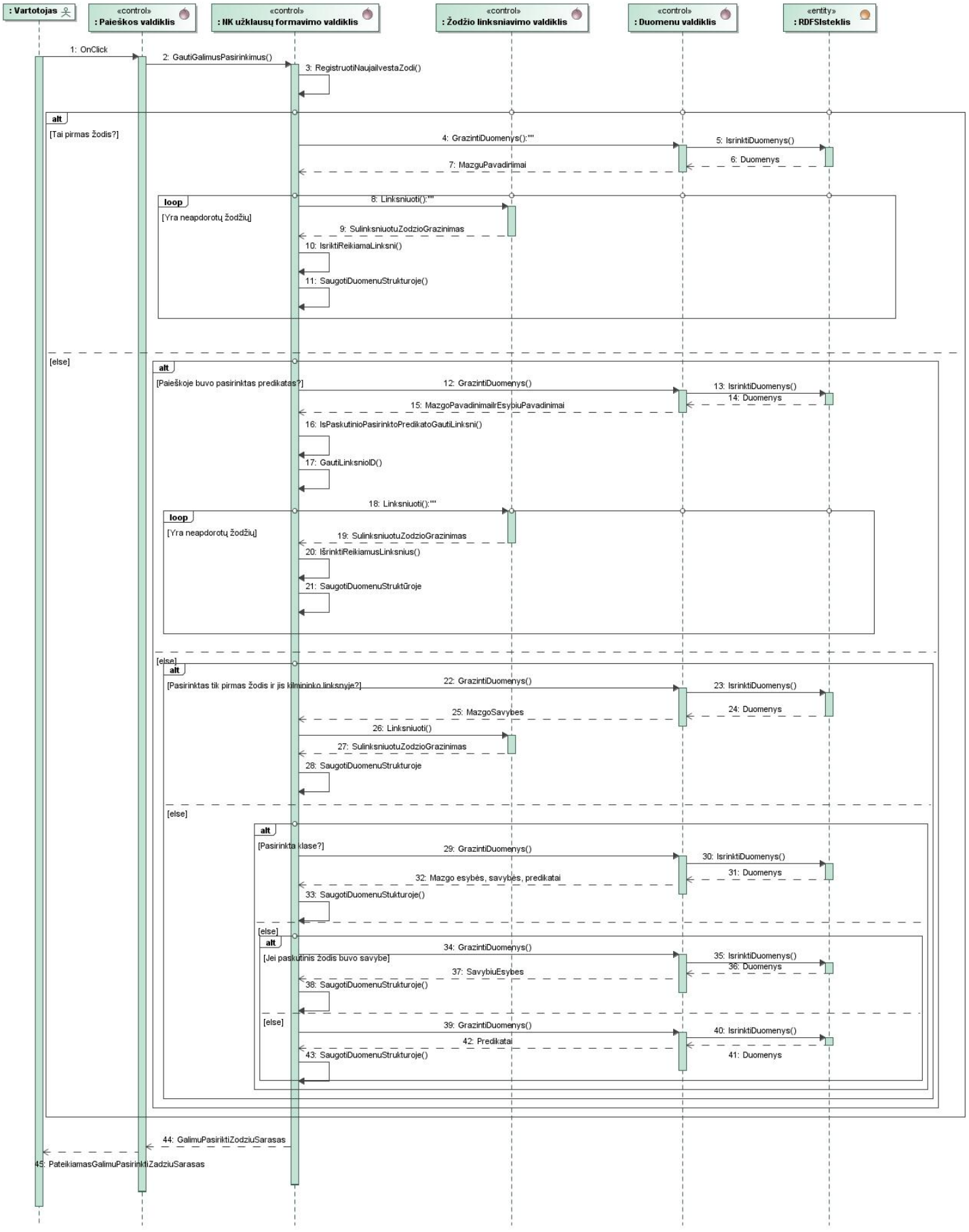
pagalbiniai linksniavimo žodžiai. Šie išrinkti žodžiai pateikiami vartotojo pasirinkimui, vartotojas turi tris pasirinkimus:

- Jei pasirenkamas klasės esybės pavadinimas, tokiu atveju iš ontologijos išrenkami klasės ryšiai ir vartotojas toliau gali eiti per kitas klases;
- Jei vartotojas pasirenka savybės pavadinimą, jam nufiltruojamos pasirinktos savybės esybė, pasirinkus norimą esybę, nufiltruojamos visi iš klasės išeinantys ryšiai ir klientas toliau gali eiti per klases;
- Jei vartotojas pasirenka ryšį, jis toliau eina per klases, analogiškai pastarajai klasei.

Kiekvieno iš ontologijos išrinkimo metu į laikiną duomenų masyvą yra įrašomas išrinkto, sulinksniuoto žodžio aprašas, kuris atrodo:

- pradinė žodžio reikšmė;
- sakinio pradžia + sulinksniuota žodis reikšmė;
- objekto tipas (*Pagalbinis*, *Class*, *Data property*, *Object properties*, *Individual about*);
- žodžio linksnis;
- Užklausos dalis (pvz. „*?x Auto:Gamintojas_Pavadinimas ?g.*“) – užklausos dalis reikalinga NK sakinio transformavimui į formalią SPARQL užklausą. Ši užklausos dalis gali turėti ir papildomą struktūrą (pvz: *„1*2*\"Vidutinė klasė\".*“*), jos struktūra yra: kelintas objektas nuo pabaigos yra redaguojamas * kelintas užklausos kintamasis yra keičiamas (1 - pirma, 2 – antra) * į ką keičiamas, nauja reikšmė.

Pasirinkus žodį, pasirinktojo žodžio aprašas yra perkeliamas į užklausos masyvą. Taip yra sudaromas NLK įvesto sakinio aprašas.

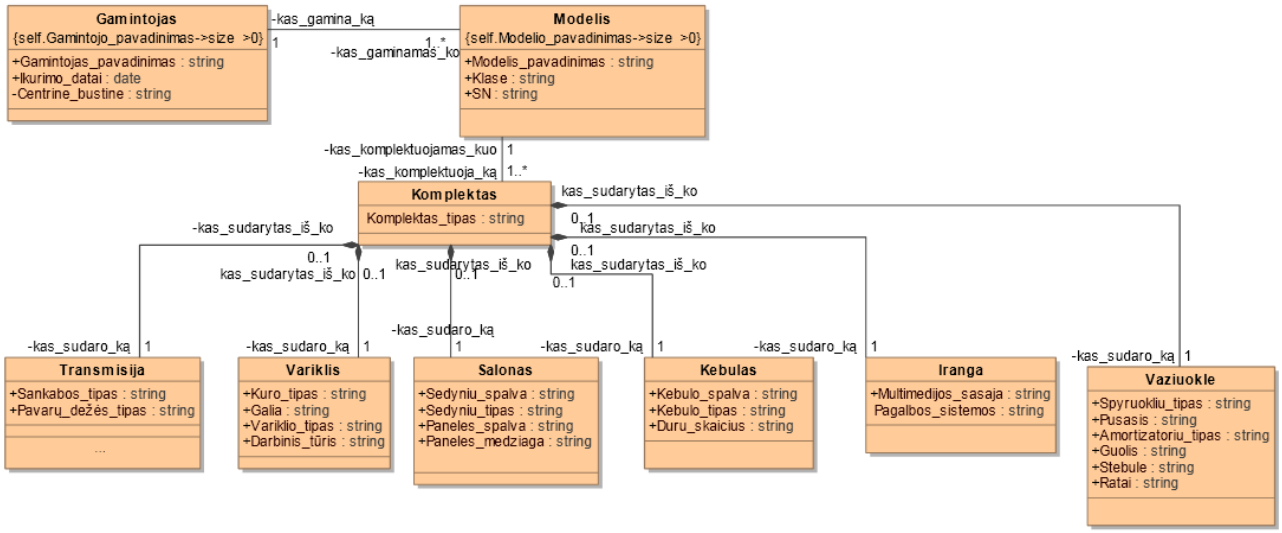


Pav. 19 „Suformuoti užklausą NK“ sekų diagrama

4.3. Ontologijos konceptų schema

Šiame skyriuje aptariama sistemoje naudojamos ontologijos schema, kuri naudojama sukurtos programos demonstracijai.

Paveiksle 20 yra pavaizduota sudarytos Auto ontologijos schema. Ši ontologija aprašo gamintojus, gamintojo gaminamus automobilių modelius ir jų komplektaciją. Ontologija OWL kalb yra pateikta PRIEDAS 9.1. Ontologijoje yra aprašytos devynios klasės: Gamintojas; Modelis; Komplektacija; Transmisija; Variklis; Salonas; Kėbulas; Įranga; Važiuoklė.



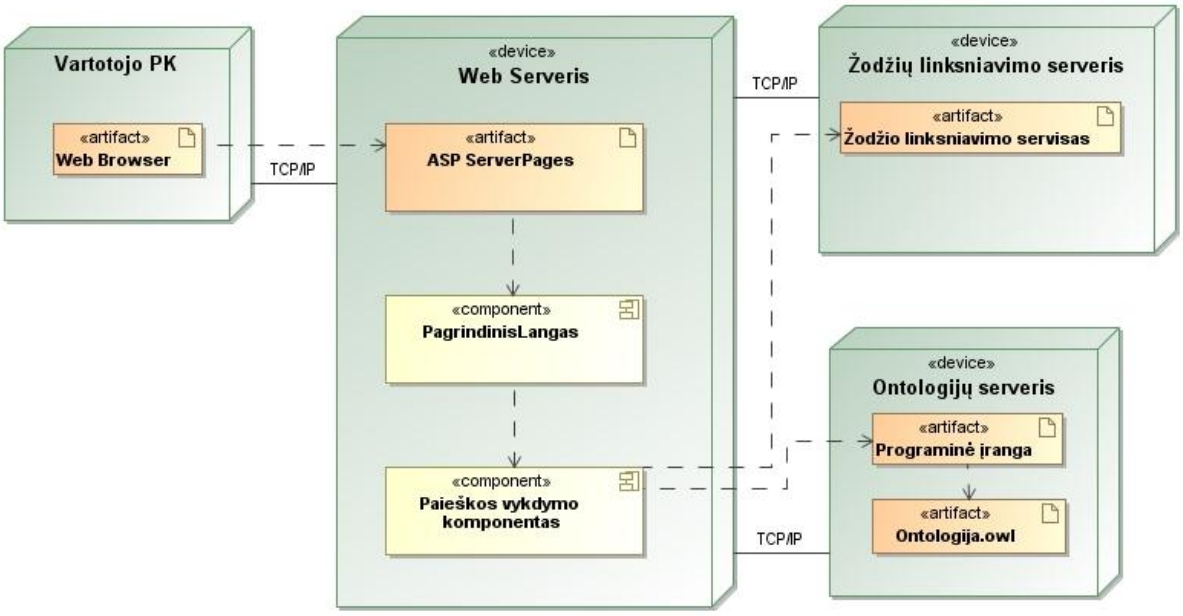
Pav. 20 Ontologijos konceptų diagrama

Klasė *Gamintojas* turi savybes *Gamintojas_pavadinimas*, *įkūrimo_data*, *cetrinė_bustinė*. Klasė *gamintojas* ryšių *kas_gamina_ką* siejamas su klase *Modelis*, ryšyje nurodytas linksnis parodo, kad ryšys *gamina* reikalauja galininko linksnio (ką) iš klasės *Modelis*. Klasė *Modelis* turi *Modelio_pavadinimas*, *Klasė*, *Serijinis_numeris.* Klasė *Modelis* atvirkštiniu ryšiu *kas_gaminama_kuo* jungiasi su klase *Gamintojas* iš gamintojo reikalaujamas kilmininko linksnis. Modelis taip pat ryšiu kas_komplektas_kuo jungiama su *Komplektas* klase, ryšys reikalauja kilmininko linksnio, klasė turi *komplektas_tipas* savybę. Klasė *komplektas* atvirkštiniu ryšiu *kas_komplektuoja_ką* jungiamas su klase *Modelis* ir reikalauja galininko linksnio. Klasė *komplektas* ryšiu kas_sudarytas_iš_ko jungiamas su klasėmis *Transmisija*, *Variklis*, *Salinas*, *Kėbulas*, *Įranga*, *Važiuoklė* ir iš šių klasių reikalauja kilmininko linksnio, atvirkštinis ryšiu *kas_sudaro_ką* šios klasės sujungtos su *Komplektas* klase ir iš jo reikalauja galininko linksnio. Klasės turi tokias savybes:

- Klasė *Transmisija* turi savybes *Sankabos_tipas*, *Pavarų_dežės_tipas.*
- Klasė *Variklis* turi savybes *Kuro_tipas*, *Galia*, *Variklio_tipas*, *Darbinis_tūris*.
- Klasė *Salonas* turi savybes *Sėdynių_spalva*, *Sėdynių_tipas*, *Paneles_spalva*, *Paneles_medžiaga*
- Klasė Kėbulas turi savybes *Kėbulo_spalva*, *Kėbul_ tipas*, *Durų_skaičius*.
- Klasė *Įranga* turi savybes *Multimedijos_sąsaja*, *Pagalbos_sistemos*.
- Klasės *Važiuoklė* turi savybes *Spyruoklių_tipas*, *Pusašis*, *Amortizatorių_tipas*, *Guolis*, *Stebulė*, *Ratai*.

4.4. Diegimo modelis

Diegimo modelis (Pav. 21) demonstruoja kaip sistema turi būti realizuota. Šiame modelyje pateikiama informacija, jog ontologijų serveryje bus įdiegta programinė įranga kuri apdoros iš Web serveryje esančio paieškos valdymo komponento atsiųstas SPARQL užklausas ir iš ontologijos išrinks reikalingus duomenis. Web serveryje bus įdiegta IS, šį IS bus pasiekiama naudojant naršyklę. Schemoje matome, kad paieškos valdymo komponentas taip pat kreipsis ir į žodžių linksniavimo servisą.



Pav. 21 Realizacijos modelis

Vartotojas asmeninio kompiuteryje esančios naršyklės pagalba per internetini tinklą jungiasi prie serveryje esančios ASP puslapio.

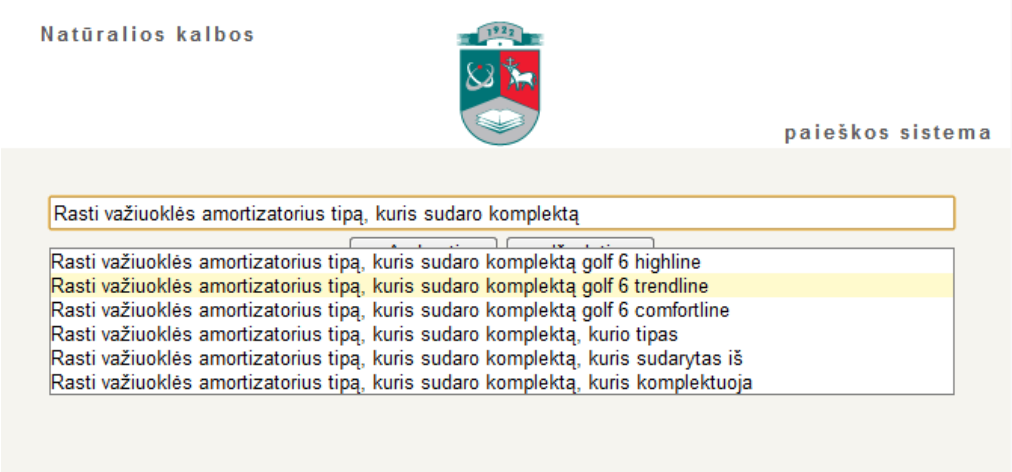
4.5. Sistemos realizavimo programinė įranga

Sistemai kurti buvo pasirinktas Microsoft Visual Studio 2008. Ontologijos redaktorius buvo pasirinktas Protege 4.0.2. Užklausoms į ontologijas pateikti, duomenų grąžinimui pasirinktas Joseki 3.4.4 prieigos taškas. .Net komutacijai su Joseki prieigos tašku naudojamas dotNetRDF biblioteka. Svetainės talpinimui serveryje yra naudojamas Microsoft IIS 7.

5. SISTEMOS REALIZACIJA

5.1. Sistemos veikimo aprašymas

Natūralios kalbos paieškos sistema – paieškos sistema skirta paieškai atlikti ontologijos failuose, užklausas įvedant natūralia lietuvių kalba. Šis paieškos puslapis yra pasiekiamas interaktyviai [20]. Siekiant sukurti kuo vartotojui patogesnę informacijos paieškos galimybę, naudojamas vieno laukelio principas. Šio principo pagalba vartotojas be specialaus parošimo, specifinių žinių, intuityviai gali atlikti paiešką ontologijos faile užklausą vesdamas NLK (Pav. 22). 22 paveiksle galime matyti integruoto suflerio pateikiamų duomenų pavyzdį, kuris padeda atrinkti reikalingus duomenys.

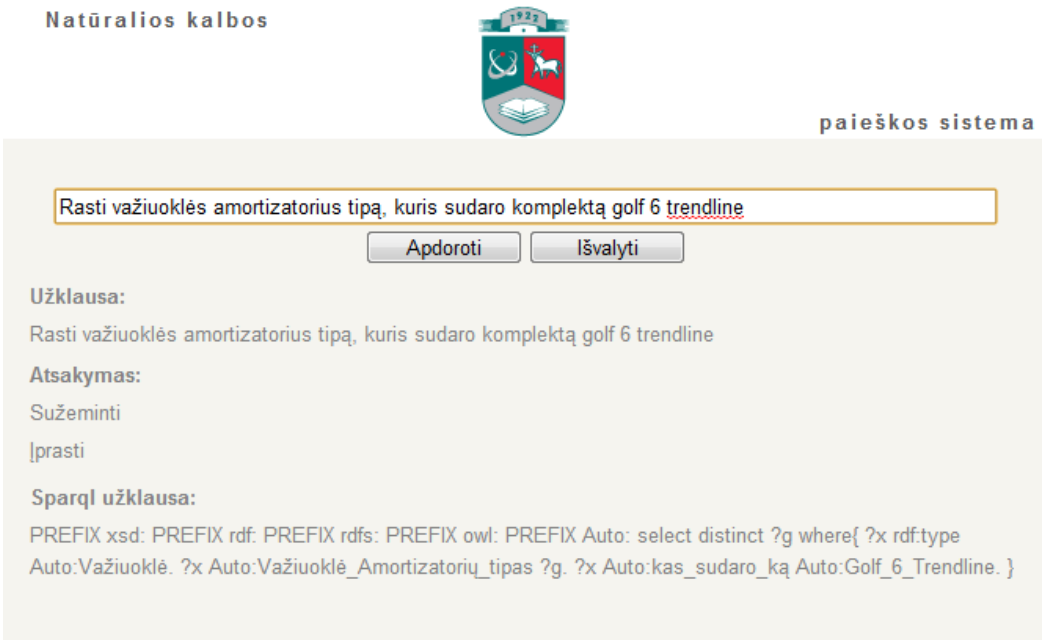


Pav. 22 Grafinė sąsaja užklausos įvedimas

Šis funkcionalumas įgyvendintas naudojantis 2.5.7.3. dalyje aptarto linksniavimo serviso funkcionalumą (PRIEDAS 9.2.3) , sudarytų SPARQL užklausų šablonais (PRIEDAS 9.2.2) , pritaikius 2.5.3 dalyje pastebėtą liepiamųjų sakinių panašumą į formalią užklausų kalbą, taip pat pritaikius 2.5.5. dalyje pastebėtą daiktavardžių, būdvardžių bei skaitvardžių priklausomybės nuo prieš tai einančio veiksmažodžio. Šios realizacijos kodas yra pateiktas PRIEDAS 9.2.1.

Šią sistema gali naudotis bet kuris norintis, kurio kompiuteryje yra įdiegta naršyklė ir turi prieigą prie interneto. Šiuo metu paieška atliekama (Pav. 23) sudarytoje bandomojoje ontologijoje, kuri pateikta PRIEDAS 9.1. Vartotojas atlikęs paiešką rezultatų lauke mato ne tik gautus rezultatus iš ontologijos, tačiau ir pačia SPARQL užklausą, kuri yra įvestos užklausos NLK analogas.

Realizacijos metu pagal 20 paveiksle pateiktą ontologijos konceptų diagramą buvo sudaryta dalykinės srities ontologija, kuri pateikta PRIEDAS 9.1.

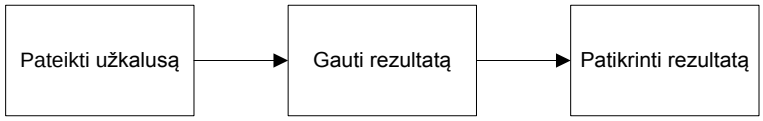


Pav. 23 Grafinė sąsaja atsakymų gavimas

5.2. Testavimo modelis

Pagrindinis testavimo tikslas – įsitikinti, kad tinkamai veikia realizuotos informacinės sistemos dalys, kurios naudojamos NLK sakiniui sudaryti, taip pat NK parašyto sakinio transformavimas į formalia SPARQL kalbą korektiškumas, gaunamo atsakymo teisingumas.

Testavimas bus atliekamas pagal 24 paveiksle pavaizduotą modelį, sistemai bus paduodamos užklausos ir gautas rezultatas bus tikrinamas su numanomu atsakymu.



Pav. 24 Testavimo modelis

5.3. Testavimo duomenys ir rezultatai

5.3.1. SPARQL užklausos sistemai testuoti

Užklausa skirta išrinkanti visas klases ontologijoje:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Auto: <http://www.semanticweb.org/ontologies/2012/0/Auto.owl#>
select ?IsrinktosKlase
where{
?IsrinktosKlase rdf:type owl:Class;
}
```

Laukiamas rezultatas:

<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Įranga>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Komplektacija>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Važiuoklė>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Gamintojas>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Salonas>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Transmisija>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Kėbulas>

Užklausa skirta išrinkanti visus ryšius su kuria siejasi klasė Modelis:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Auto: <http://www.semanticweb.org/ontologies/2012/0/Auto.owl#>
select ?x where{
?x  rdf:type owl:ObjectProperty;
 rdfs:domain Auto:Modelis.
}
```

Laukiamas rezultatas:

<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_gaminamas_ko>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_komplektuojamas_kuo>

Užklausa skirta išrinkanti visus Modelis klases duomenų tipus:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Auto: <http://www.semanticweb.org/ontologies/2012/0/Auto.owl#>
select ?x
where{
?x  rdf:type owl:DatatypeProperty .
?x rdfs:domain Auto:Modelis.
}
```

Laukiamas rezultatas:

<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis_Klasė>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis_Serijinis_numeris>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis_Pavadinimas>

Užklausa skirta išrinkanti Komplektas klases esybes:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX Auto: <http://www.semanticweb.org/ontologies/2012/0/Auto.owl#>
select *
 where{
?x rdf:type owl:NamedIndividual.
?x   rdf:type Auto:Komplektas .
}
```

Laukiamas rezultatas:

<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6_Comfortline>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6_Trendline>
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6_Highline>

Atlikus šias užklausas buvo gauti analogiški atsakymai. Galime teigti, kad šį posistemė veikia korektiškai.

5.3.2. Morfologinio nagrinėjimo posistemio testavimas

Linksniuojame žodį komplektas, norime gauti visus jo linksnius.

Laukiamas rezultatas:

```
"v" : "Komplektas",
"k" : "Komplekto",
"n" : "Komplektui",
"g" : "Komplektą",
"įn" : "Komplektu",
"vt" : "Komplekte",
"š" : "Komplekte",
"forma" : "Komplektas"
```

Laukiamas rezultatas sutampa su gautuoju rezultatu.

Linksniuojame žodį salonas, norime gauti visus jo linksnius.

Laukiamas rezultatas:


```
"v" : "Salonas",
"k" : "Salono",
"n" : "Salonui",
"g" : "Saloną",
"įn" : "Salonu",
"vt" : "Salone",
"š" : "Salone",
"forma" : "Salonas"
```


Laukiamas rezultatas sutampa su gautuoju rezultatu.

Linksniuojame žodį gamintojas, norime gauti visus jo linksnius.

Laukiamas rezultatas:


```
"v" : "Gamintojas",
"k" : "Gamintojo",
"n" : "Gamintojui",
"g" : "Gamintoją",
"įn" : "Gamintoju",
"vt" : "Gamintojuje",
"š" : "Gamintojau",
"forma" : "Gamintojas"
```


Gautas rezultatas:


```
"v" : "gamintojos",
"k" : "gamintojų",
"n" : "gamintojoms",
"g" : "gamintojas",
"įn" : "gamintojomis",
"vt" : "gamintojose",
"š" : "gamintojos",
"forma" : "gamintojas"
```


Laukiamas rezultatas nesutampa su gautuoju rezultatu. Gautame rezultate yra sulinksniuota daugiskaitos forma. Šios klaidos ištaisyti negalime, todėl bus kreiptasi į šį tinklo servisą sukūrusius studentus.

5.4. Realizacijos etapo rezultatai

1. Sudaryta NLK paieškos sistemos prototipas, kuris prieinamas http://zeta.teledema.lt/teledema/Svetaines/Kiriciuk/PaieskosSistema/ internetiniu adresu.
2. Sudaryta dalykinės srities bandomoji ontologija, kuri pateikta PRIEDAS 9.1.
3. Ištestuotos atskiros posistemės. Buvo nustatyta, kad iš ontologijos duomenų išrinkimas vyksta korektiškai. Testuojant linksniavimo posistemę buvo rasti linksnių nesutapimai nesutapimas, šį klaida nėra kritinė, sistema gali toliau funkcionuoti, kol nebus ištaisyta klaida.
4. Pastebėti paieškos sistemos apribojimai: pateikiant užklausą neatitinkančią lietuvių kalbos taisykles, paieškos sistema neranda atitikmenų pagal asociacijas, todėl vartotojas turi būti susipažinęs su paieškos sistemos užklausų pateikimo formatu.
5. Pastebėti paieškos sistemos privalumai: vartotojas pratinamas naudoti taisyklingą lietuvių kalbos sakinio formavimo struktūrą, laikantis taisyklių; vartotojui pasiūlomi paieškos variantai; greitaveika atitinka keliamus reikalavimus; sistema lengvai pasiekiama vartotojui; intuityvi; lengvai skaitoma; aiškiai atvaizduojami atsakymai į užklausas; sistema lengvai koreguojama.

6. EKSPERIMENTINIS SISTEMOS TYRIMAS

6.1.Eksperimento planas

Magistrinio darbo eksperimento tyrimo metu buvo siekiama įvertinti sudaromų užklausų teisingumą, korektiškumą, intuityvumą. Eksperimento metu buvo sudaryti šeši iš dalykinės srities (Pav. 19) NLK klausimai, kuriuos norėjome pateikti sistemai eksperimento metu. 4 lentelėje pateikiami gauto eksperimento rezultatai.

6.2.Eksperimento rezultatai

4 lentelė. Eksperimento rezultatų lentelė

		Ko laukiama iš sistemos		
Eil. Nr.	Užklausa NLK	Suformuluota NLK užklausa sistemoje	SPARQL užklausa	Įvykdymo rezultatai
1.	Rasti salono tipą, kuris sudaro komplektą Golf 6 Trendline	Rasti salono tipą, kuris sudaro komplektą golf 6 trendline	select distinct ?g where{ ?x rdf:type Auto:Salonas. ?x Auto:Salonas_Tipas ?g. ?x Auto:kas_sudaro_ką Auto:Golf_6_Trendline. }	Veliūras
2.	Rasti variklį, kuis sudaro komplektą, kurio tipas Trendline, kuris sudarytas iš salono Elite	Rasti variklį, kuris sudaro komplektą, kurio tipas trendline, kuris sudarytas iš salono elitas	select distinct ?x where{ ?x rdf:type Auto:Variklis. ?x Auto:kas_sudaro_ką ?OP2 . ?OP2 rdf:type Auto:Komplektas. ?OP2 Auto:Komplektas_Tipas "Trendline". ?OP2 Auto:kas_sudarytas_iš_ko Auto:Salonas_Elite. }	Variklis Benzinas 1.4 Variklis Dyzelinis 2.0
3.	Rasti važiuoklės amortizatorių tipą, kuris sudaro komplektą, kuris sudarytas iš salono Elite	Rasti važiuoklės amortizatorius tipą, kuris sudaro komplektą, kuris sudarytas iš salono elitas	select distinct ?g where{ ?x rdf:type Auto:Važiuoklė. ?x Auto:Važiuoklė_Amortizatorių_tipas ?g. ?x Auto:kas_sudaro_ką ?OP3. ?OP3 rdf:type Auto:Komplektas. ?OP3 Auto:kas_sudarytas_iš_ko Auto:Salonas_Elite.}	Sužeminti Įprasti
4.	Rasti variklį, kuris sudaro komplektą	Rasti variklį, kuris sudaro komplektą	select distinct ?x where{ ?x rdf:type Auto:Variklis. ?x Auto:kas_sudaro_ką ?OP2. ?OP2 rdf:type Auto:Komplektas.}	Variklis Dyzelinis 1.9 Variklis Benzinas 1.4 Variklis Dyzelinis 2.0
5.	Rasti komplektus	Rasti komplektą	select distinct ?x where{ ?x rdf:type Auto:Komplektas.}	Golf 6 Highline Golf 6 Trendline Golf 6 Comfortline
6.	Rasti salono tipą, kuris kažką sudaro.	Rasti salono tipą, kuris sudaro	select distinct ?g where{ ?x rdf:type Auto:Salonas. ?x Auto:Salonas_Tipas ?g. ?x Auto:kas_sudaro_ką ?OP3 .}	Odinis Veliūras

Iš 4 lentelės pateiktų eksperimento rezultatų, matome, kad žmogaus sugalvotų liepiamųjų sakinių struktūra artima sudarytų sakinių struktūrai, kartasi net identiška. Iš pateikiamų užklausų pobūdžio galime teigti, kad sudaromos užklausos yra universalios, t.y. sistemos galima klausti visų klausimų susijusių su ontologijoje esančių klasių, klasių savybių, esybių bei ryšių elementais. Eksperimento metu buvo pastebėta, kad užklausos sistemoje yra sudaromos intuityviai, suflerio NLK sakinių formuluotės yra korektiškos, teisingos.

7. IŠVADOS

1. Išanalizavus ontologijos užrašymo kalbas ir jų kūrimo įrankius, galima teigti, kad nėra aiškiai apibrėžta kaip nurodyti linksnius, kurių reikalauja lietuvių kalbos žodžių kaitymo taisyklės. Ontologijai kurti buvo pasirinkta ontologijos pateikimo RDF schema, kuri yra praplėsta OWL klasė tipais. Ontologijai kurti pasirinktas Protege redaktorius.
2. Išanalizuoti SPARQL užklausų kalbą bei vykdymo įrankius, pareiškėjo, kad šį užklausų kalba yra rekomenduota W3C konsorciumo naudojama duomenų išrinkimui iš ontologijos, nes jis yra standartas. Sistema naudodama dotNetRDF biblioteką užklausas perduoda į Joseki SPARQL prieigos tašką.
3. Aptarus lietuvių kalbos gramatikos taisykles, įtakojančias užklausų formulavimą natūralią lietuvių kalba, nustatyti pagrindiniai aspektai: veiksmažodžiai pasižymi tuo, kad valdo daiktavardžius ir būdvardžius, tačiau pagal jį negalima nustatyti reikalaujamo linksnio; klasių bei klasių savybių pavadinimams įvardinti yra naudojami daiktavardžiai; klasės savybėse gali būti naudojami būdvardžiai arba skaitvardžiai. Todėl galima teigti, kad pagrindinė lietuviu kalbos gramatikos taisykle įtakojanti užklausų formulavimą – turėti sulinksniuotus daiktavardžius būdvardžius ir skaitvardžius.
4. Suprojektavus ir realizavus prototipinę paieškos ontologijoje sistemą išaiškėjo tokie jos trūkumai: pateikiant užklausą neatitinkančią lietuvių kalbos taisykles, paieškos sistema neranda atitikmenų pagal asociacijas, todėl vartotojas turi būti susipažinęs su paieškos sistemos užklausų pateikimo formatu. Pastebėti privalumai: vartotojas pratinamas naudoti taisyklingą lietuvių kalbos sakinio formavimo struktūrą, laikantis taisyklių; vartotojui pasiūlomi paieškos variantai; greitaveika atitinka keliamus reikalavimus; sistema lengvai pasiekiama vartotojui; intuityvi; lengvai skaitoma; aiškiai atvaizduojami atsakymai į užklausas; sistema lengvai koreguojama.
5. Atlikto eksperimento metu buvo pastebėta, kad liepiamųjų sakinių struktūra yra artima arba identiška sistemos sudarytų sakinių struktūrai. Iš sistemai pateikiamų užklausų struktūrų, galime daryti išvadas, kad sistemai pateikiamos užklausos yra universalios, t.y. sistemos galima klausti visų klausimų susijusių su ontologijoje esančių klasių, klasių savybių, esybių bei ryšių elementais. Eksperimento metu buvo pastebėta, kad užklausos sistemoje yra sudaromos intuityviai, suflerio NLK sakinių formuluotės yra korektiškos, teisingos.

8. LITERATŪRA

1. „Онтологии и Представление Знаний“, Boris Kone [interaktyvus] [žiūrėta 2010-12-20] Prieiga per internetą: http://shcherbak.net/2010/11/video-lekcij-po-ontologiyam-i-predstavleniyu-znanij-prodolzhenie/
2. „Semantic Web Road map“ [interaktyvus] [žiūrėta 2010-10-18] Prieiga per internetą: http://www.w3.org/DesignIssues/Semantic.html
3. „Краткое введение в RDF“ [interaktyvus] [žiūrėta 2010-10-18] Prieiga per internetą: http://xmlhack.ru/texts/06/rdf-quickintro/rdf-quickintro.html
4. „Онтологии в корпоративных системах“ [interaktyvus] [žiūrėta 2010-11-25] Prieiga per internetą: http://www.management.com.ua/ims/ims116.html
5. „Введение в RDF и Jena RDF API“, Braen Makbrad [interaktyvus] [žiūrėta 2010-11-17] http://www.semantictools.ru/tools/13---rdf--jena-rdf-api.html
6. „SPARQL query language for RDF“ [interaktyvus] [žiūrėta 2010-11-17] Prieiga per internetą: http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/
7. „Protege overview“ [interaktyvus] [žiūrėta 2010-11-17] Prieiga per internetą: http://protege.stanford.edu/overview/
8. „Joseki“ [interaktyvus] [žiūrėta 2010-11-17] Prieiga per internetą: http://www.joseki.org/
9. Pranas Kniūkška, „Lietuvių kalbos žinynas“, Kaunas, 2003.
10. „Lietuvių kalbos automatinis morfologinis atpažinimas ir sintezė (lietuvių kalbos morfologinio žodyno kūrimas)“, Vytautas Zinkevičius (darbo vadovas ir vykdytojas), Erika Rimkutė (VDU Kompiuterinės lingvistikos centro lingvistė, doktorantė), 2005.
11. „Sintaksinių santykių reiškimo priemonės“ [interaktyvus] [žiūrėta 2010-12-18] Prieiga per internetą: http://ualgiman.dtiltas.lt/sintaksiniu_santykiu_reiskimo_priemones.html
12. „Ontology Definition Metamodel“ [interaktyvus] [žiūrėta 2011-04-15] Prieiga per internetą: http://www.omg.org/spec/ODM/1.0/PDF/
13. „OWL Web Ontology Language“ [interaktyvus] [žiūrėta 2012-04-18] Prieiga per internetą: http://www.w3.org/TR/owl-features/
14. „A Semantic Web Primer for Object-Oriented Software Developers“ [interaktyvus] [žiūrėta 2012-04-15] Prieiga per internetą: http://www.w3.org/TR/2006/NOTE-sw-oosd-primer-20060309/
15. „RDF Primer“ [interaktyvus] [žiūrėta 2012-04-15] Prieiga per internetą: http://www.w3.org/TR/rdf-primer/
16. „SPARQL by Example“ [interaktyvus] [žiūrėta 2012-05-15] Prieiga per internetą: http://www.cambridgesemantics.com/semantic-university/sparql-by-example#%284%29

17. „Ontology editor“ [interaktyvus] [žiūrėta 2012-05-12] Prieiga per internetą: http://en.wikipedia.org/wiki/Ontology_editor
18. „dotNetRDF“ [interaktyvus] [žiūrėta 2012-05-01] Prieiga per internetą: http://www.dotnetrdf.org/
19. T.R. Guber 1993
20. „Natūralios kalbos paieškos sistema“ Prieiga per internetą: http://zeta.teledema.lt/teledema/Svetaines/Kiriciuk/PaieskosSistema/

9. PRIEDAI

9.1.Ontologija

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY Auto "http://www.semanticweb.org/ontologies/2012/0/Auto.owl#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2012/0/Auto.owl#"
     xml:base="http://www.semanticweb.org/ontologies/2012/0/Auto.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:Auto="http://www.semanticweb.org/ontologies/2012/0/Auto.owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>
     <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_da&#382;omas_kuo -->
    <owl:ObjectProperty rdf:about="#kas_da&#382;omas_kuo">
        <rdfs:domain rdf:resource="#K&#279;bulas"/>
        <rdfs:range rdf:resource="#Spalva"/>
        <owl:inverseOf rdf:resource="#kuo_da&#382;omi_kas"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_gamina_k&#261; -->
    <owl:ObjectProperty rdf:about="#kas_gamina_k&#261;">
        <rdfs:domain rdf:resource="#Gamintojas"/>
        <rdfs:range rdf:resource="#Modelis"/>
        <owl:inverseOf rdf:resource="#kas_gaminamas_ko"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_gaminamas_ko -->
    <owl:ObjectProperty rdf:about="#kas_gaminamas_ko">
        <rdfs:range rdf:resource="#Gamintojas"/>
        <rdfs:domain rdf:resource="#Modelis"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_komplektuoja_k&#261; -->
    <owl:ObjectProperty rdf:about="#kas_komplektuoja_k&#261;">
        <rdfs:domain rdf:resource="#Komplektas"/>
        <rdfs:range rdf:resource="#Modelis"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_komplektuojamas_kuo -->
    <owl:ObjectProperty rdf:about="#kas_komplektuojamas_kuo">
        <rdfs:range rdf:resource="#Komplektas"/>
        <rdfs:domain rdf:resource="#Modelis"/>
        <owl:inverseOf rdf:resource="#kas_komplektuoja_k&#261;"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_sudaro_k&#261; -->
   <owl:ObjectProperty rdf:about="#kas_sudaro_k&#261;">
        <rdfs:range rdf:resource="#Komplektas"/>
        <rdfs:domain rdf:resource="#K&#279;bulas"/>
```

```
            <rdfs:domain rdf:resource="#Salonas"/>
            <rdfs:domain rdf:resource="#Transmisija"/>
            <rdfs:domain rdf:resource="#Variklis"/>
            <rdfs:domain rdf:resource="#Va&#382;iuokl&#279;"/>
            <rdfs:domain rdf:resource="#&#302;ranga"/>
        </owl:ObjectProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kas_sudarytas_i&#353;_ko -->
        <owl:ObjectProperty rdf:about="#kas_sudarytas_i&#353;_ko">
            <rdfs:domain rdf:resource="#Komplektas"/>
            <rdfs:range rdf:resource="#K&#279;bulas"/>
            <rdfs:range rdf:resource="#Salonas"/>
            <rdfs:range rdf:resource="#Transmisija"/>
            <rdfs:range rdf:resource="#Variklis"/>
            <rdfs:range rdf:resource="#Va&#382;iuokl&#279;"/>
            <owl:inverseOf rdf:resource="#kas_sudaro_k&#261;"/>
            <rdfs:range rdf:resource="#&#302;ranga"/>
        </owl:ObjectProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#kuo_da&#382;omi_kas -->
        <owl:ObjectProperty rdf:about="#kuo_da&#382;omi_kas">
            <rdfs:range rdf:resource="#K&#279;bulas"/>
            <rdfs:domain rdf:resource="#Spalva"/>
        </owl:ObjectProperty>
        <!--
        ///////////////////////////////////////////////////////////////////////////////////
        //
        // Data properties
        //
        ///////////////////////////////////////////////////////////////////////////////////
         -->
        <!--
http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Gamintojas_Centrin&#279;_b&#363;stin&#279; -->
        <owl:DatatypeProperty rdf:about="#Gamintojas_Centrin&#279;_b&#363;stin&#279;">
            <rdfs:domain rdf:resource="#Gamintojas"/>
            <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Gamintojas_Pavadinimas -->
        <owl:DatatypeProperty rdf:about="#Gamintojas_Pavadinimas">
            <rdfs:domain rdf:resource="#Gamintojas"/>
            <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Gamintojas_&#302;k&#363;rimo_data -->
        <owl:DatatypeProperty rdf:about="#Gamintojas_&#302;k&#363;rimo_data">
            <rdfs:domain rdf:resource="#Gamintojas"/>
            <rdfs:range rdf:resource="&xsd;date"/>
        </owl:DatatypeProperty>
         <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Komplektas_Tipas -->
        <owl:DatatypeProperty rdf:about="#Komplektas_Tipas">
            <rdfs:domain rdf:resource="#Komplektas"/>
            <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#K&#279;bulas_Dur&#371;_skai&#269;ius -->
        <owl:DatatypeProperty rdf:about="#K&#279;bulas_Dur&#371;_skai&#269;ius">
            <rdfs:domain rdf:resource="#K&#279;bulas"/>
            <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#K&#279;bulas_Spalva -->
        <owl:DatatypeProperty rdf:about="#K&#279;bulas_Spalva">
            <rdfs:domain rdf:resource="#K&#279;bulas"/>
            <rdfs:range rdf:resource="&xsd;string"/>
        </owl:DatatypeProperty>
        <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#K&#279;bulas_Tipas -->
        <owl:DatatypeProperty rdf:about="#K&#279;bulas_Tipas">
```

<rdfs:domain rdf:resource="#Kėbulas"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis_Klasė -->
<owl:DatatypeProperty rdf:about="#Modelis_Klasė">
<rdfs:domain rdf:resource="#Modelis"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis_Pavadinimas -->
<owl:DatatypeProperty rdf:about="#Modelis_Pavadinimas">
<rdfs:domain rdf:resource="#Modelis"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis_Serijinis_numeris -->
<owl:DatatypeProperty rdf:about="#Modelis_Serijinis_numeris">
<rdfs:domain rdf:resource="#Modelis"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Salonas_Spalva -->
<owl:DatatypeProperty rdf:about="#Salonas_Spalva">
<rdfs:domain rdf:resource="#Salonas"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Salonas_Tipas -->
<owl:DatatypeProperty rdf:about="#Salonas_Tipas">
<rdfs:domain rdf:resource="#Salonas"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Spalva_Spalva -->
<owl:DatatypeProperty rdf:about="#Spalva_Spalva">
<rdfs:domain rdf:resource="#Spalva"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!--
http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Transmisija_Pavarų_dežės_tipas -->
<owl:DatatypeProperty rdf:about="#Transmisija_Pavarų_dežės_tipas">
<rdfs:domain rdf:resource="#Transmisija"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Transmisija_Sankabos_tipas -->
<owl:DatatypeProperty rdf:about="#Transmisija_Sankabos_tipas">
<rdfs:domain rdf:resource="#Transmisija"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Darbinis_tūris -->
<owl:DatatypeProperty rdf:about="#Variklis_Darbinis_tūris">
<rdfs:domain rdf:resource="#Variklis"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Galia -->
<owl:DatatypeProperty rdf:about="#Variklis_Galia">
<rdfs:domain rdf:resource="#Variklis"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Kuro_tipas -->
<owl:DatatypeProperty rdf:about="#Variklis_Kuro_tipas">
<rdfs:domain rdf:resource="#Variklis"/>
<rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Tipas -->
<owl:DatatypeProperty rdf:about="#Variklis_Tipas">
<rdfs:domain rdf:resource="#Variklis"/>

```
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!--
http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Va&#382;iuokl&#279;_Amortizatori&#371;_tipas -->
    <owl:DatatypeProperty rdf:about="#Va&#382;iuokl&#279;_Amortizatori&#371;_tipas">
        <rdfs:domain rdf:resource="#Va&#382;iuokl&#279;"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Va&#382;iuokl&#279;_Ratai -->
    <owl:DatatypeProperty rdf:about="#Va&#382;iuokl&#279;_Ratai">
        <rdfs:domain rdf:resource="#Va&#382;iuokl&#279;"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Va&#382;iuokl&#279;_Tipas -->
    <owl:DatatypeProperty rdf:about="#Va&#382;iuokl&#279;_Tipas">
        <rdfs:domain rdf:resource="#Va&#382;iuokl&#279;"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#&#302;ranga_Garso_&#303;ranga -->
    <owl:DatatypeProperty rdf:about="#&#302;ranga_Garso_&#303;ranga">
        <rdfs:domain rdf:resource="#&#302;ranga"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#&#302;ranga_Pagalbin&#279;s_sistemos -->
    <owl:DatatypeProperty rdf:about="#&#302;ranga_Pagalbin&#279;s_sistemos">
        <rdfs:domain rdf:resource="#&#302;ranga"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <!--
    ///////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Gamintojas -->
    <owl:Class rdf:about="#Gamintojas">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
      <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Komplektas -->
    <owl:Class rdf:about="#Komplektas">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#K&#279;bulas -->
    <owl:Class rdf:about="#K&#279;bulas">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Modelis -->
    <owl:Class rdf:about="#Modelis">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Salonas -->
    <owl:Class rdf:about="#Salonas">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Spalva -->
    <owl:Class rdf:about="#Spalva">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    </owl:Class>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Transmisija -->
    <owl:Class rdf:about="#Transmisija">
        <rdfs:subClassOf rdf:resource="&owl;Thing"/>
```

```
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis -->
<owl:Class rdf:about="#Variklis">
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Va&#382;iuokl&#279; -->
<owl:Class rdf:about="#Va&#382;iuokl&#279;">
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#&#302;ranga -->
<owl:Class rdf:about="#&#302;ranga">
   <rdfs:subClassOf rdf:resource="&owl;Thing"/>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing
<owl:Class rdf:about="&owl;Thing"/>
-->
<!--
////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
////////////////////////////////////////////////////////////////////////////////
 -->
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6 -->
<Modelis rdf:about="#Golf_6">
   <rdf:type rdf:resource="&owl;Thing"/>
   <Modelis_Klasė>Vidutin&#279; klas&#279;</Modelis_Klasė>
   <Modelis_Serijinis_numeris>VWG6</Modelis_Serijinis_numeris>
   <Modelis_Pavadinimas>Golf</Modelis_Pavadinimas>
   <kas_komplektuojamas_kuo rdf:resource="#Golf_6_Comfortline"/>
   <kas_komplektuojamas_kuo rdf:resource="#Golf_6_Highline"/>
   <kas_komplektuojamas_kuo rdf:resource="#Golf_6_Trendline"/>
   <kas_gaminamas_ko rdf:resource="#VW"/>
</Modelis>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6_Comfortline -->
<owl:Thing rdf:about="#Golf_6_Comfortline">
   <rdf:type rdf:resource="#Komplektas"/>
   <Komplektas_Tipas>Comfortline</Komplektas_Tipas>
   <kas_komplektuoja_ką rdf:resource="#Golf_6"/>
   <kas_sudarytas_iš_ko rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
   <kas_sudarytas_iš_ko rdf:resource="#K&#279;bulas_Universal"/>
   <kas_sudarytas_iš_ko rdf:resource="#Salonas_Elite"/>
   <kas_sudarytas_iš_ko rdf:resource="#Salonas_Liuks"/>
   <kas_sudarytas_iš_ko rdf:resource="#Transmisija_Liuks"/>
   <kas_sudarytas_iš_ko rdf:resource="#Variklis_Benzinas_1.4"/>
   <kas_sudarytas_iš_ko rdf:resource="#Variklis_Dyzelinis_2.0"/>
   <kas_sudarytas_iš_ko rdf:resource="#Va&#382;iuokl&#279;_Comfort"/>
   <kas_sudarytas_iš_ko rdf:resource="#Va&#382;iuokl&#279;_Sport"/>
   <kas_sudarytas_iš_ko rdf:resource="#&#302;ranga_Alpine"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6_Highline -->
<owl:Thing rdf:about="#Golf_6_Highline">
   <rdf:type rdf:resource="#Komplektas"/>
   <Komplektas_Tipas>Highline</Komplektas_Tipas>
   <kas_komplektuoja_ką rdf:resource="#Golf_6"/>
   <kas_sudarytas_iš_ko rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
   <kas_sudarytas_iš_ko rdf:resource="#K&#279;bulas_Universal"/>
   <kas_sudarytas_iš_ko rdf:resource="#Salonas_Elite"/>
   <kas_sudarytas_iš_ko rdf:resource="#Salonas_Liuks"/>
   <kas_sudarytas_iš_ko rdf:resource="#Transmisija_Liuks"/>
   <kas_sudarytas_iš_ko rdf:resource="#Variklis_Dyzelinis_1.9"/>
   <kas_sudarytas_iš_ko rdf:resource="#Variklis_Dyzelinis_2.0"/>
```

```
    <kas_sudarytas_iš_ko rdf:resource="#Va&#382;iuokl&#279;_Comfort"/>
    <kas_sudarytas_iš_ko rdf:resource="#Va&#382;iuokl&#279;_Sport"/>
    <kas_sudarytas_iš_ko rdf:resource="#&#302;ranga_Alpine"/>
</owl:Thing>
 <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Golf_6_Trendline -->
<owl:Thing rdf:about="#Golf_6_Trendline">
    <rdf:type rdf:resource="#Komplektas"/>
    <Komplektas_Tipas>Trendline</Komplektas_Tipas>
    <kas_komplektuoja_ką rdf:resource="#Golf_6"/>
    <kas_sudarytas_iš_ko rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
    <kas_sudarytas_iš_ko rdf:resource="#K&#279;bulas_Universal"/>
    <kas_sudarytas_iš_ko rdf:resource="#Salonas_Elite"/>
    <kas_sudarytas_iš_ko rdf:resource="#Transmisija_Sport"/>
    <kas_sudarytas_iš_ko rdf:resource="#Variklis_Benzinas_1.4"/>
    <kas_sudarytas_iš_ko rdf:resource="#Variklis_Dyzelinis_2.0"/>
    <kas_sudarytas_iš_ko rdf:resource="#Va&#382;iuokl&#279;_Comfort"/>
    <kas_sudarytas_iš_ko rdf:resource="#Va&#382;iuokl&#279;_Sport"/>
    <kas_sudarytas_iš_ko rdf:resource="#&#302;ranga_Alpine"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#K&#279;bulas_Het&#269;bekas -->
<Kėbulas rdf:about="#K&#279;bulas_Het&#269;bekas">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Kėbulas_Tipas>He&#269;bek</Kėbulas_Tipas>
    <Kėbulas_Durų_skaičius>5</Kėbulas_Durų_skaičius>
    <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_Geltona"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_Juoda"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_M&#279;lyna"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_Raudona"/>
</Kėbulas>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#K&#279;bulas_Universal -->
<owl:Thing rdf:about="#K&#279;bulas_Universal">
    <rdf:type rdf:resource="#K&#279;bulas"/>
    <Kėbulas_Durų_skaičius>5</Kėbulas_Durų_skaičius>
    <Kėbulas_Tipas>Universal</Kėbulas_Tipas>
    <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_Geltona"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_Juoda"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_M&#279;lyna"/>
    <kas_dažomas_kuo rdf:resource="#Spalva_Raudona"/>
</owl:Thing>
  <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Salonas_Elite -->
<owl:Thing rdf:about="#Salonas_Elite">
    <rdf:type rdf:resource="#Salonas"/>
    <Salonas_Spalva>Baltas</Salonas_Spalva>
    <Salonas_Tipas>Veliuras</Salonas_Tipas>
    <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
</owl:Thing>
 <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Salonas_Liuks -->
<Salonas rdf:about="#Salonas_Liuks">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Salonas_Spalva>Baltai - juodas</Salonas_Spalva>
    <Salonas_Tipas>Odinis</Salonas_Tipas>
    <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
</Salonas>
```

```
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Spalva_Geltona -->
<Spalva rdf:about="#Spalva_Geltona">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Spalva_Spalva>Geltona</Spalva_Spalva>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Universal"/>
</Spalva>
 <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Spalva_Juoda -->
<Spalva rdf:about="#Spalva_Juoda">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Spalva_Spalva>Juoda</Spalva_Spalva>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Universal"/>
</Spalva>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Spalva_M&#279;lyna -->
<Spalva rdf:about="#Spalva_M&#279;lyna">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Spalva_Spalva>M&#279;lyna</Spalva_Spalva>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Universal"/>
</Spalva>
 <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Spalva_Raudona -->
<owl:Thing rdf:about="#Spalva_Raudona">
    <rdf:type rdf:resource="#Spalva"/>
    <Spalva_Spalva>Raudona</Spalva_Spalva>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Het&#269;bekas"/>
    <kuo_dažomi_kas rdf:resource="#K&#279;bulas_Universal"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Transmisija_Liuks -->
<Transmisija rdf:about="#Transmisija_Liuks">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Transmisija_Sankabos_tipas>Plaukiojanti</Transmisija_Sankabos_tipas>
    <Transmisija_Pavarų_dežės_tipas>Mechanin&#279;</Transmisija_Pavarų_dežės_tipas>
    <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
</Transmisija>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Transmisija_Sport -->
<owl:Thing rdf:about="#Transmisija_Sport">
    <rdf:type rdf:resource="#Transmisija"/>
    <Transmisija_Pavarų_dežės_tipas>Automatin&#279;</Transmisija_Pavarų_dežės_tipas>
    <Transmisija_Sankabos_tipas>Sportin&#279;</Transmisija_Sankabos_tipas>
    <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#VW -->
<owl:Thing rdf:about="#VW">
    <rdf:type rdf:resource="#Gamintojas"/>
    <Gamintojas_Centrinė_būstinė>Volfsburgas</Gamintojas_Centrinė_būstinė>
    <Gamintojas_Įkūrimo_data>1937-01-01</Gamintojas_Įkūrimo_data>
    <Gamintojas_Pavadinimas>Volksvagen</Gamintojas_Pavadinimas>
    <kas_gamina_ką rdf:resource="#Golf_6"/>
</owl:Thing>
<!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Benzinas_1.4 -->
<Variklis rdf:about="#Variklis_Benzinas_1.4">
    <rdf:type rdf:resource="&owl;Thing"/>
    <Variklis_Kuro_tipas>Benzinas</Variklis_Kuro_tipas>
    <Variklis_Galia>59</Variklis_Galia>
    <Variklis_Darbinis_tūris>1400</Variklis_Darbinis_tūris>
    <Variklis_Tipas>S</Variklis_Tipas>
    <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
    <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
</Variklis>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Dyzelinis_1.9 -->
```

```
    <owl:Thing rdf:about="#Variklis_Dyzelinis_1.9">
      <rdf:type rdf:resource="#Variklis"/>
      <Variklis_Kuro_tipas>Dyzelis</Variklis_Kuro_tipas>
      <Variklis_Tipas>TDI</Variklis_Tipas>
      <Variklis_Galia>85</Variklis_Galia>
      <Variklis_Darbinis_tūris>1896</Variklis_Darbinis_tūris>
      <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
    </owl:Thing>
     <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Variklis_Dyzelinis_2.0 -->
    <owl:Thing rdf:about="#Variklis_Dyzelinis_2.0">
      <rdf:type rdf:resource="#Variklis"/>
      <Variklis_Kuro_tipas>Dyzelis</Variklis_Kuro_tipas>
      <Variklis_Tipas>TD</Variklis_Tipas>
      <Variklis_Darbinis_tūris>2000</Variklis_Darbinis_tūris>
      <Variklis_Galia>103</Variklis_Galia>
      <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
    </owl:Thing>
     <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Va&#382;iuokl&#279;_Comfort -->
    <owl:Thing rdf:about="#Va&#382;iuokl&#279;_Comfort">
      <rdf:type rdf:resource="#Va&#382;iuokl&#279;"/>
      <Važiuoklė_Amortizatorių_tipas>&#302;prasti</Važiuoklė_Amortizatorių_tipas>
      <Važiuoklė_Tipas>Comfort</Važiuoklė_Tipas>
      <Važiuoklė_Ratai>Michelin R16</Važiuoklė_Ratai>
      <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
    </owl:Thing>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#Va&#382;iuokl&#279;_Sport -->
    <Važiuoklė rdf:about="#Va&#382;iuokl&#279;_Sport">
      <rdf:type rdf:resource="&owl;Thing"/>
      <Važiuoklė_Ratai
         >Michelin R17 Summer</Važiuoklė_Ratai>
      <Važiuoklė_Tipas>Sport</Važiuoklė_Tipas>
      <Važiuoklė_Amortizatorių_tipas>Su&#382;eminti</Važiuoklė_Amortizatorių_tipas>
      <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
    </Važiuoklė>
    <!-- http://www.semanticweb.org/ontologies/2012/0/Auto.owl#&#302;ranga_Alpine -->
    <owl:Thing rdf:about="#&#302;ranga_Alpine">
      <rdf:type rdf:resource="#&#302;ranga"/>
      <Įranga_Pagalbinės_sistemos>Alpine</Įranga_Pagalbinės_sistemos>
      <Įranga_Garso_įranga>Alpine</Įranga_Garso_įranga>
      <kas_sudaro_ką rdf:resource="#Golf_6_Comfortline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Highline"/>
      <kas_sudaro_ką rdf:resource="#Golf_6_Trendline"/>
    </owl:Thing>
  </rdf:RDF>
```

9.2.Programiniai kodai

9.2.1. Užklausos formavimo programinis kodas

```
//-------------------------------------------------
  // Suformuojama uzklausa, atliekanti paieška ontologijoje
  protected void GautiRezultata_Click(object sender, EventArgs e)
  {
     PapildomosFunkcijos.SPARQL sparql = new SPARQL();
     string[,] uzklMas = (string[,])HttpContext.Current.Session["uzklMas"];// Masyvas skirtas užklausos duomenims saugoti
     string nezinomasis = (string)HttpContext.Current.Session["nezinomasis"];
     string uzklausa = "";
     if (uzklMas != null)
     {
        for (int i = 0; i < uzklMas.Length; i++)
        {
           if (uzklMas[i, 4] != null)
           {
              uzklausa += uzklMas[i, 4] + " ";
           }
           else
              break;
        }
        if (uzklausa != "")
        {
           string rezultatas = sparql.GautiAtsakyma(nezinomasis, uzklausa);
           string[] atsakymas = rezultatas.Split(';');
           tekst.InnerHtml = "<table >";
           tekst.InnerHtml += "<tr><th >Atsakymas</th></tr>";
           foreach (string eilute in atsakymas)
           {
              tekst.InnerHtml += String.Format("<tr><td >{0}</td></tr>", eilute.Replace("_", " "));
           }
           tekst.InnerHtml += "</table>";
        }
     }
  }
  //-------------------------------------------------
  //Formuojama užklausos duomenys
  [System.Web.Services.WebMethodAttribute(), System.Web.Script.Services.ScriptMethodAttribute()]
  public static string[] GautiZodziuSarasa(string prefixText, int count, string contextKey)
  {
     int maxKiek = 50;
     PapildomosFunkcijos.SPARQL sparql = new SPARQL();
     bool naujasZodis = false;
     // tikrinama ar klientas nori įvesti naują žodį
     if (prefixText.EndsWith(" "))
        naujasZodis = true;
     string pradzia = prefixText.Replace("  ", " ").TrimEnd(' ');
     prefixText = prefixText.Remove(0, prefixText.LastIndexOf(" ") + 1);
     string[] sakinys = pradzia.Split(' ');
     //-------------------------------------------------
     string rezultatas;
     //0 - V; 1 - K; 2 - N; 3 - G; 4 - In; 5 - Vt; 6 - Š
     int[] linksnis = { -1, -1, -1, -1, -1, -1, -1 };
     string objectoTipas = "";
     string[] zodziai = new string[50];
```

```
        string[] atsakymas;
        string uzklausa = ""; // skirta saugoti generuojamai sparql uzklausai
        int zodzKiek = 0;// zodPasi esančių žodžių kiekis
        //*******************************
        // priskiriami sesijos kintamieji
        string[,] uzklMas = (string[,])HttpContext.Current.Session["uzklMas"];// Masyvas skirtas užklausos duomenims
saugoti
        string[,] zodPasi = new string[maxKiek, 5]; // Masyvas skirtas galimiems pasirinkimams saugoti
        zodPasi = (string[,])HttpContext.Current.Session["duoMas"];
        atsakymas = (string[])HttpContext.Current.Session["atsakymas"];
        string nezinomasis = (string)HttpContext.Current.Session["nezinomasis"];
        if (uzklMas == null)// Jei masyvas neegzistuoja, jį sukuria
            uzklMas = new string[maxKiek, 5];
        if (zodPasi == null)
            zodPasi = new string[maxKiek, 5];
        if (atsakymas == null)
            atsakymas = new string[maxKiek];
        if (nezinomasis == null)
            nezinomasis = "";
        //-------------------------------------------------
        // Jei tai pradinis žodis
        // jei įvedant paskutinis simbolis buvo įvestas tarpas
        if (naujasZodis)
        {
            //sakinio strukturos saugojimas
            bool priskirtiDuom = false;
            for (int i = 0; i < maxKiek; i++)
            {
                if (zodPasi[i, 1] == pradzia)
                {
                    for (int q = 0; q < maxKiek; q++)
                        if (uzklMas[q, 0] == null)
                        {
                            uzklMas[q, 0] = zodPasi[i, 0];
                            uzklMas[q, 1] = zodPasi[i, 1];
                            uzklMas[q, 2] = zodPasi[i, 2];
                            uzklMas[q, 3] = zodPasi[i, 3];
                            string[] veiksm;
                            if (zodPasi[i, 4].Contains("*"))
                            {
                                veiksm = zodPasi[i, 4].Split('*');
                                int grizt = Convert.ToInt32(veiksm[0]);
                                grizt--;
                                while (grizt > 0)
                                {
                                    uzklMas[q - grizt, 4] = "";
                                    grizt--;
                                }
                                if (Convert.ToInt32(veiksm[1]) == 2)
                                {
                                    grizt = Convert.ToInt32(veiksm[0]);
                                    string veiksmas = uzklMas[q - grizt, 4].Remove(uzklMas[q - grizt, 4].LastIndexOf("?"), uzklMas[q
- grizt, 4].Length - uzklMas[q - grizt, 4].LastIndexOf("?"));
                                    uzklMas[q - grizt, 4] = veiksmas + veiksm[2];
                                }
                                else
                                    uzklMas[grizt, 4] = veiksm[2] + uzklMas[q - grizt, 4].Remove(0, uzklMas[q - grizt,
4].IndexOf("?"));
                                uzklMas[q, 4] = "";
                            }
                            else
                                uzklMas[q, 4] = zodPasi[i, 4];
```

```
                priskirtiDuom = true;
                break;
            }
            else
                if (uzklMas[q, 1] == pradzia)
                    break;
    } if (priskirtiDuom)
        break;
    if (zodPasi[i, 1] == null)
    {
        break;
    }
}
HttpContext.Current.Session["uzklMas"] = uzklMas;
//**********************************
// jeigu buvo istrinti žodžiai iš užklausos, tai visas galas nukerpamas
if (!priskirtiDuom)
{
    bool baigti = false;
    string sak = "";
    for (int i = 0; i < maxKiek; i++)
    {
        sak = uzklMas[i, 1];
        if (sak == null)
            break;
        if (!pradzia.StartsWith(sak))
        {
            while (i < maxKiek)
            {
                uzklMas[i, 0] = null;
                uzklMas[i, 1] = null;
                uzklMas[i, 2] = null;
                uzklMas[i, 3] = null;
                uzklMas[i, 4] = null;
                if (uzklMas[i + 1, 0] == null)
                {
                    baigti = true;
                    break;
                }
                i++;
            }
        }
        if (baigti)
            break;
    }
}
HttpContext.Current.Session["uzklMas"] = uzklMas;
//**********************************
//isrenkamas naujas žodis
//išvalomas išvalomas išrenkamų žodžių masyvas
zodPasi = null;
zodPasi = new string[maxKiek, 5];
bool linksniuoti = false;
bool uzklParam = false;
for (int i = 0; i < maxKiek; i++)
{
    if (uzklMas[i, 0] == null)
    {
        zodzKiek = i;
        objectoTipas = uzklMas[i, 2];
        break;
    }
```

```
    if (uzklMas[i, 2] == "PagalbinisParametram")
        uzklParam = true;
}
if (!uzklParam)
{
    if (sakinys.Count() == 1)
    {
        // Sukuriamas sarasas
        rezultatas = sparql.gautiKlases();
        linksnis[3] = 3;
        linksnis[1] = 1;
        objectoTipas = "Class";
        zodziai = rezultatas.Split(';');
        //**
        uzklausa = "?x rdf:type Auto:xxx.";
        nezinomasis = " ?x ";
        //**
        uzklMas[0, 0] = sakinys[0];
        uzklMas[0, 1] = sakinys[0];
        uzklMas[0, 2] = "Pagalbinis";
        uzklMas[0, 3] = "";
        uzklMas[0, 4] = "";
        //**
        linksniuoti = true;
    }
    //*********************************
    //isrenkami visos klases property
    if (zodzKiek == 2 && uzklMas[1, 3] == "1")
    {
        rezultatas = sparql.gautiKlasesDataParametrus(uzklMas[1, 0]);
        linksnis[3] = 3;
        objectoTipas = "Data property";
        zodziai = rezultatas.Split(';');
        //**
        uzklausa = "?x Auto:xxx ?g.";
        nezinomasis = " ?g ";
        //**
        linksniuoti = true;
    }
    //*********************************
    if (((zodzKiek == 2 && uzklMas[1, 3] == "3") || uzklMas[2, 3] == "3") && !uzklParam)
    {
        atsakymas = null;
        atsakymas = new string[maxKiek];
        atsakymas[0] = pradzia + ", kuris";
        /*   uzklMas[zodzKiek, 0] = "kuris";
          uzklMas[zodzKiek, 1] = ", kuris";
          uzklMas[zodzKiek, 2] = "PagalbinisParametram";
          uzklMas[zodzKiek, 3] = "";
          uzklMas[zodzKiek, 4] = "";
         */
        zodPasi[0, 0] = ", kuris";
        zodPasi[0, 1] = pradzia + ", kuris";
        zodPasi[0, 2] = "PagalbinisParametram";
        zodPasi[0, 3] = "";
        zodPasi[0, 4] = "";

    }
    //*********************************
}
else
{  //-----------------------------------------------
```

```
            // Pirmas žodis po klases pavadinimo, parametrų įvedimas
            int rysioLinks = 0;
            if (uzklMas[zodzKiek - 1, 2] == "PagalbinisParametram")
            {
                rezultatas = sparql.gautiKlasesRysius(uzklMas[1, 0]);
                uzklausa = "?x Auto:xxx ?c .";
                linksnis[3] = 3;
                objectoTipas = "Object properties";
                zodziai = rezultatas.TrimEnd(';').Split(';');
                linksniuoti = true;
            }
            else
            {
                objectoTipas = uzklMas[zodzKiek - 1, 2];
                HttpContext.Current.Session["duoMas"] = zodPasi;
                HttpContext.Current.Session["uzklMas"] = uzklMas;
                switch (objectoTipas)
                {
                    //******************
                    case "Object properties":
                        rezultatas = sparql.gautiRysiuKlases(uzklMas[zodzKiek - 1, 0]);
                        rysioLinks = gautiLinksnioID(uzklMas[zodzKiek - 1, 0]);
                        linksnis[rysioLinks] = rysioLinks;
                        objectoTipas = "Class";
                        uzklausa = uzklMas[zodzKiek - 1, 4];
                        uzklausa = uzklausa.Remove(0, uzklausa.LastIndexOf("?")).Replace(".", "") + "rdf:type Auto:xxx.";
                        zodziai = rezultatas.TrimEnd(';').Split(';');
                        linksniuoti = true;
                        // ieskom ryšio klases
                        break;
                    //******************
                    case "Class":
                        rezultatas = sparql.GautiKlasesEsybiuPavadinimus(uzklMas[zodzKiek - 1, 0]);
                        zodziai = rezultatas.TrimEnd(';').Split(';');
                        objectoTipas = "Individual about";
                        linksnis[0] = 0;
                        uzklausa = "2*2*Auto:xxx* ";
                        atsakymas = GautiSulinksniuotus(maxKiek, zodziai, objectoTipas, zodzKiek, linksnis, pradzia,
uzklausa);
                        //**
                        rezultatas = sparql.gautiKlasesDataParametrus(uzklMas[zodzKiek - 1, 0]);
                        linksnis[0] = 0;
                        objectoTipas = "Data property";
                        zodziai = rezultatas.Split(';');
                        string[] atsakymas2 = GautiSulinksniuotus(maxKiek, zodziai, objectoTipas, zodzKiek, linksnis, pradzia
+ ", kurio", uzklausa);
                        // sutraukia į 1 masyvą
                        int i2 = 0;
                        for (int i = 0; i < atsakymas.Count(); i++)
                        {
                            if (atsakymas[i] == null)
                            {
                                atsakymas[i] = atsakymas2[i2];
                                i2++;
                            }
                            if (atsakymas2[i2] == null || atsakymas2.Count() < i2)
                                break;
                        }
                        linksniuoti = false;
                        //      rezultatas = sparql.gautiKlasesRysius(uzklMas[zodzKiek-1, 0]);
                        // ieskom klases rysiu bei duomenu parametrus
                        break;
```

```
                //*******************
                case "Data property":
                    // ieskom duomenu individual
                    rezultatas = sparql.GautiKlasesDataPropertyReiksme(uzklMas[zodzKiek - 2, 0], uzklMas[zodzKiek - 1,
0]);
                    zodziai = rezultatas.TrimEnd(';').Split(';');
                    objectoTipas = "Individual about";
                    linksnis[0] = 0;
                    uzklausa = "";
                    atsakymas = GautiSulinksniuotus(maxKiek, zodziai, objectoTipas, zodzKiek, linksnis, pradzia,
uzklausa);
                    linksniuoti = false;
                    break;
                //*******************
                case "Individual":
                    // ieskom duomenu individual
                    break;
            }
            zodPasi = (string[,])HttpContext.Current.Session["duoMas"];
            uzklMas = (string[,])HttpContext.Current.Session["uzklMas"];
        }
    }
    //-------------------------------------------------
    // Linksniavimas
    HttpContext.Current.Session["nezinomasis"] = nezinomasis;
    HttpContext.Current.Session["duoMas"] = zodPasi;
    HttpContext.Current.Session["uzklMas"] = uzklMas;
    if (linksniuoti)//(sakinys.Count() == 1) || (zodzKiek == 2 && uzklMas[1, 3] == "1"))
    {
        atsakymas = GautiSulinksniuotus(maxKiek, zodziai, objectoTipas, zodzKiek, linksnis, pradzia, uzklausa);
    }
}
//-------------------------------------------------
#region  masyvo minimizavimas, tuščių elementų masyve panaikinimas
string[] NEWatsakymas;
int a = 0;
string[] laikAtsakymas = atsakymas;
foreach (string eil in laikAtsakymas)
{
    if (eil != null)
        if (eil.StartsWith(pradzia))
        {
            a++;
        }
}
NEWatsakymas = new string[a];
laikAtsakymas = atsakymas;
a = 0;
foreach (string eil in laikAtsakymas)
{
    if (eil != null)
        if (eil.StartsWith(pradzia))
        {
            NEWatsakymas[a] = eil;
            a++;
        }
}
atsakymas = NEWatsakymas;
#endregion
//-------------------------------------------------
//0- pradine forma; 1 - sulinksniuota; 2- objectoTipas; 3 - linksnis;
HttpContext.Current.Session["atsakymas"] = atsakymas;
```

```
        return (from m in atsakymas where m.StartsWith(pradzia, StringComparison.CurrentCultureIgnoreCase) select
m).Take(10).ToArray();
    }
    //-------------------------------------------------
    //Linksniuojamas žodžių masyvas
    static string[] GautiSulinksniuotus(int maxKiek, string[] zodziai, string objectoTipas, int zodzKiek, int[] linksnis, string
pradzia, string uzklausa)
    {
        PapildomosFunkcijos.Linksniavimas linksn = new PapildomosFunkcijos.Linksniavimas();
        int AtsakEilNr = 0;
        string[,] uzklMas = (string[,])HttpContext.Current.Session["uzklMas"];
        string[,] zodPasi = (string[,])HttpContext.Current.Session["duoMas"];
        string[] atsakymas = new string[maxKiek];
        for (int i = 0; i < zodziai.Length; i++)
        {
            string zodisLinksniav = "";
            //***************
            if (objectoTipas == "Object properties")
            {
                if (zodziai[i].Contains("_yra_"))
                    zodisLinksniav = zodziai[i].Replace("yra_", "");
                else
                    zodisLinksniav = zodziai[i];
                zodisLinksniav = zodisLinksniav.Remove(0, zodziai[i].IndexOf('_') + 1);

                zodisLinksniav = zodisLinksniav.Remove(zodisLinksniav.LastIndexOf('_'), zodisLinksniav.Length -
zodisLinksniav.LastIndexOf('_')).Replace("_", " ");
            }
            else
                zodisLinksniav = zodziai[i].Replace("_", " ");
            //***************
            string[] zodSudet;
            if (zodisLinksniav.ToString().IndexOf(' ') > 0)
            {
                zodSudet = zodisLinksniav.ToString().Split(' ');
                foreach (string vienasZ in zodSudet)
                {
                    if (uzklMas[zodzKiek - 1, 0].ToLower().Contains(linksn.linksniuoti(vienasZ.ToLower(), 0)))
                    {
                        zodisLinksniav = zodisLinksniav.Replace(vienasZ, "");
                    }
                }
                zodisLinksniav = zodisLinksniav.TrimStart(' ').TrimEnd(' ');
            }
            //***************
            for (int f = 0; f <= 6; f++)
            {
                if (linksnis[f] >= 0 && zodziai[i] != "")
                {
                    string sulinskniuota = "";
                    if (linksnis[f] == 3 && zodisLinksniav.Contains(' '))
                    {
                        zodSudet = zodisLinksniav.Split(' ');
                        foreach (string zod in zodSudet)
                        {
                            sulinskniuota += " " + linksn.linksniuoti(zod.ToLower(), linksnis[f]);
                        }
                    }
                    else
                        sulinskniuota = " " + linksn.linksniuoti(zodisLinksniav.ToLower(), linksnis[f]);
                    //***************
                    string laikinas = pradzia + sulinskniuota;
```

```
                atsakymas[AtsakEilNr] = laikinas.Replace("  ", " ");
                zodPasi[AtsakEilNr, 0] = zodziai[i];
                zodPasi[AtsakEilNr, 1] = laikinas.Replace("  ", " ");
                zodPasi[AtsakEilNr, 2] = objectoTipas;
                zodPasi[AtsakEilNr, 3] = linksnis[f].ToString();
                zodPasi[AtsakEilNr, 4] = uzklausa.Replace("xxx", zodziai[i]);
                AtsakEilNr++;
            }
        }
    }
    //-----------------------------------------------
    return atsakymas;
}
//-----------------------------------------------
//Gaunamas ryšyje nurodytas linksnio ID
static int gautiLinksnioID(string risys)
{
    int linksnioID = -1;
    risys = risys.Remove(0, risys.LastIndexOf('_') + 1);
    switch (risys)
    {
        //0 - V; 1 - K; 2 - N; 3 - G; 4 - In; 5 - Vt; 6 - Š
        case "kas":
            linksnioID = 0;
            break;
        case "ko":
            linksnioID = 1;
            break;
        case "kam":
            linksnioID = 2;
            break;
        case "ką":
            linksnioID = 3;
            break;
        case "kuo":
            linksnioID = 4;
            break;
        case "kur":
            linksnioID = 5;
            break;
    }
    return linksnioID;
}

//-----------------------------------------------
```

9.2.2. SPARQL užklausų sudarymo šablono programinis kodas

```
public class SPARQL
    {
        string prefix = "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> \n PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> \n  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> \n
PREFIX owl: <http://www.w3.org/2002/07/owl#> \n  PREFIX Auto:
<http://www.semanticweb.org/ontologies/2012/0/Auto.owl#> \n";
        PagalbinisWebService.Service SPARQL_uzklausos = new PagalbinisWebService.Service();
        public SPARQL()
        {

        }
        //-----------------------------------------------
```

```
        //Gaunamos visos ontologijos klases
        public string gautiKlases()
        {
            string rez = SPARQL_uzklausos.GautiSPARQL(prefix + " select ?isrinktaklase where{?isrinktaklase rdf:type
owl:Class; }");
            return rez;
        }
        //-----------------------------------------------
        public string gautiKlasesRysius(string klase)
        {
            string uzklausa = prefix + "select ?isrinktipar where{?isrinktipar  rdf:type owl:ObjectProperty; rdfs:domain Auto:"
+ klase + ".}";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(uzklausa);
            return rez1;
        }
        //-----------------------------------------------
        // isrenka klases kurios eina po ryšio
        public string gautiRysiuKlases(string risys)
        {
            string uzklausa = prefix + "select ?isrRysKla where{Auto:" + risys + " rdfs:range ?isrRysKla; }";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(uzklausa);
            return rez1;
        }
        //-----------------------------------------------
        public string gautiKlasesDataParametrus(string klase)
        {
            string uzklausa = prefix + "select ?x \n where{?x  rdf:type owl:DatatypeProperty .\n ?x rdfs:domain Auto:" +
klase + ".}";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(uzklausa);
            return rez1;
        }
        //-----------------------------------------------
        public string gautiKlasesElementus(string klase)
        {
            string uzklausa = prefix + " select * where{ ?isr rdf:type owl:NamedIndividual. rdfs:resource Auto:" + klase + "
.}";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(uzklausa);
            return rez1;
        }
        //-----------------------------------------------
        public string GautiKlasesEsybiuPavadinimus(string klase)
        {
            string uzklausa = prefix + " select * where{ ?x rdf:type owl:NamedIndividual. ?x   rdf:type Auto:" + klase + " .}";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(uzklausa);
            return rez1;
        }
        //-----------------------------------------------
        public string GautiKlasesDataPropertyReiksme(string klase, string dataProperty)
        {
            string uzklausa = prefix + " select ?b where{ ?x rdf:type owl:NamedIndividual. ?x   rdf:type Auto:" + klase + " .?x
Auto:" + dataProperty + " ?b.}";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(uzklausa);
            return rez1;
        }
        //-----------------------------------------------
        public string GautiAtsakyma(string ieskomasis, string uzklausa)
        {
            string duomenys = prefix + " select distinct " + ieskomasis + " where{" + uzklausa + "}";
            string rez1 = SPARQL_uzklausos.GautiSPARQL(duomenys);
            return rez1;
        }
        //-----------------------------------------------
```

```
}
```

9.2.3. Linksniavimo serviso programinis kodas

```
public string linksniuoti(string zodis, int linksnis)
    {
      //0 - V; 1 - K; 2 - N; 3 - G; 4 - In; 5 - Vt; 6 - Š
      string atsakymas = "";
      string[] doc = linksniavimas(zodis); //Linksn.GautiLinksnius(zodis);
      if (doc.Count() - 1 < linksnis)
        atsakymas = doc[doc.Count() - 1];// jei norimas linksnis yra negalimas, grąžinama pagrindinė forma
      else
        atsakymas = doc[linksnis];
      return atsakymas;
    }
    //--------------------------------------------------
    public string[] linksniavimas(string zodis)
    {
      string[] sakinys;
      WebClient webClient = new WebClient();
      NetworkCredential credentials = new NetworkCredential("wsktu", "lns_test1");
      webClient.Credentials = credentials;
      string json = DownloadString(webClient, "http://linksniavimas.virsmas.lt/linksniai.json?formos=" + zodis,
Encoding.UTF8);

      sakinys = json.Split(new char[] { ',' }); //0 - V; 1 - K; 2 - N; 3 - G; 4 - In; 5 - Vt; 6 - Š
      switch (sakinys.Count())
      {
        case 1:
          if (sakinys[0] == "[]")
            sakinys[0] = zodis;
          else
            sakinys[0] = sakinys[0].Remove(0, sakinys[0].IndexOf(':') + 3).Replace("\"", "").Replace("\n\t}\n]",
"");//forma
          break;
        case 4:
          sakinys[0] = sakinys[0].Remove(0, sakinys[0].IndexOf(':') + 3).Replace("\"", "");//I asmuo
          sakinys[1] = sakinys[1].Remove(0, sakinys[1].IndexOf(':') + 3).Replace("\"", "");//II asmuo
          sakinys[2] = sakinys[2].Remove(0, sakinys[2].IndexOf(':') + 3).Replace("\"", "");//III asmuo
          sakinys[3] = sakinys[3].Remove(0, sakinys[3].IndexOf(':') + 3).Replace("\"", "").Replace("\n\t}\n]",
"");//forma
          break;
        case 8:
          sakinys[0] = sakinys[0].Remove(0, sakinys[0].IndexOf(':') + 3).Replace("\"", "");//Vardininkas
          sakinys[1] = sakinys[1].Remove(0, sakinys[1].IndexOf(':') + 3).Replace("\"", "");//Kilmininkas
          sakinys[2] = sakinys[2].Remove(0, sakinys[2].IndexOf(':') + 3).Replace("\"", "");//Naudininkas
          sakinys[3] = sakinys[3].Remove(0, sakinys[3].IndexOf(':') + 3).Replace("\"", "");//Galininkas
          sakinys[4] = sakinys[4].Remove(0, sakinys[4].IndexOf(':') + 3).Replace("\"", "");//Inagininkas
          sakinys[5] = sakinys[5].Remove(0, sakinys[5].IndexOf(':') + 3).Replace("\"", "");//Vietininkas
          sakinys[6] = sakinys[6].Remove(0, sakinys[6].IndexOf(':') + 3).Replace("\"", "");//Šauksmininkas
          sakinys[7] = sakinys[7].Remove(0, sakinys[7].IndexOf(':') + 3).Replace("\"", "").Replace("\n\t}\n]",
"");//Forma
          break;
      }
      return sakinys;
    }
    //--------------------------------------------------
    public String DownloadString(WebClient webClient, String address, Encoding encoding)
    {
      byte[] buffer = webClient.DownloadData(address);
      byte[] bom = encoding.GetPreamble();
      if ((0 == bom.Length) || (buffer.Length < bom.Length))
```

```
{
    return encoding.GetString(buffer);
}
for (int i = 0; i < bom.Length; i++)
{
    if (buffer[i] != bom[i])
    {
        return encoding.GetString(buffer);
    }
}
return encoding.GetString(buffer, bom.Length, buffer.Length - bom.Length);        }
```

10. TERMINŲ ŽODYNAS

OKBC (ang. *Open Knowledge Base Connectivity protocol*) – atviras žynių bazės protokolas .

OWL (ang. *Web Ontology Language Overview*) – žinių atvaizdavimo kalba ontologijoms kurti.

RDF (angl. *Resource Description Framework*) – žinių atvaizdavimo kalba ontologijoms kurti.

SPARQL (angl. *Simple Protocol and RDF Query Language*) - ontologijų užklausų kalba.

RDQL(ang. *RDF Data Query Language*) - ontologijų užklausų kalba.

NLK – Natūrali Lietuvių Kalba.

NLL (ang. *Natural Lithuanian Language*) – natūrali lietuvių kalba

NK – Natūralioji Kalba.

XML (angl. *Extensible Markup Language*) - yra W3C rekomenduojama bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

IS – informacinė sistema.

IIS (angl. *Internet Information Services*) – internetinio serverio aplikacija.

W3C konsorciumo (angl. *World Wide Web Consortium*) – konsorciumas leidžiantis programinės įrangos standartus.