KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Andrius Pakaušis

AUTOMATIZUOTO ONTOLOGIJŲ UŽPILDYMO EGZEMPLIORIAIS INFORMACINĖ SISTEMA

Baigiamasis magistro projektas

Vadovas
prof. dr. L. Nemuraitė

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

AUTOMATIZUOTO ONTOLOGIJŲ UŽPILDYMO EGZEMPLIORIAIS INFORMACINĖ SISTEMA

Baigiamasis magistro projektas

Informacinių sistemų inžinerijos studijų programa (kodas 621E15001)

Vadovas

prof. dr. L. Nemuraitė

2015-05-

Recenzentas

dr. Audronė Janavičiūtė

2015-05-

Projektą atliko

Andrius Pakaušis

2015-05-

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

(Fakultetas)

(Studento vardas, pavardė)
Informacinių sistemų inžinerijos studijų programa, 621E15001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Automatizuoto ontologijų užpildymo egzemplioriais informacinė sistema“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.
Kaunas

Patvirtinu, kad mano, **Andriaus Pakaušio**, baigiamasis projektas tema „Automatizuoto ontologijų užpildymo egzemplioriais informacinė sistema“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

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

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

Andrius Pakaušis. Information System for Automated Ontology Population with Individuals. *Final Degree Project of Master of Information Systems Engineering* / Supervisor Prof. Lina Nemuraitė; Kaunas University of Technology, Faculty of Informatics.

Kaunas, 2015. 52 p.

SUMMARY

In this thesis, the Information System prototype was developed for automated populating OWL 2 ontology with individuals. For using ontologies in semantic search, ontology individuals are created via semantically annotating Web sites and linking fragments of Web content with ontology individuals and their property assertions. However, such individuals often are unreliable. For improvement of semantic search, reliable individuals should be created in ontology. In this thesis, the database model is created on the base of Semantics of Business Vocabulary and Business Rules (SBVR) for storing reliable ontology data. Three alternatives for creating reliable data about ontology individuals are proposed: manual input; extraction of textual documents; extraction from reliable Internet sources. Extraction from textual documents or Internet is performed using templates and scripts, which automatically insert data into the database. Ontology population with data, stored in the created database, is automatically performed with scripts for inserting individuals and property assertions. The experimental investigation has shown the effectiveness of the proposed approach with respect to time resources and avoiding errors. Although the created prototype is only the proof of the approach, the database model and the idea can be used for further development.

Keywords: Information System, database, ontology, SBVR, OWL 2.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Ontologija - tam tikros srities sąvokų visumos specifikavimas.

Individas (ontologijoje) – konkretus dalykinės srities objektas, klasės egzempliorius.

SBVR - yra standartas skirtas modeliuoti veiklos žodyną ir taisykles natūralia kalba ir yra pagrįstas prasmės ir jos vaizdavimo atskyrimu. SVBR žodynas leidžia specifikuoti konceptus, jų ryšius, apibrėžimus ir įvairias jų vaizdavimo formas natūralia kalba. Šios specifikacijos suprantamos žmogui ir gali būti apdorojamos kompiuteriu, kadangi yra pagrįstos formalia logika.

ĮVADAS

Saityne atliekama semantinė paieška vykdoma ontologijose, kurių egzemplioriai sukuriami semantinio anotavimo metu. Šie egzemplioriai dažnai yra nepatikimi. Semantinės paieškos sistemų darbas galėtų būti pagerintas, jei jos turėtų patikimus ontologijų duomenis, užpildytus iš patikimų šaltinių.

Šiame darbe buvo tiriami galimi ontologijų užpildymo egzemplioriais metodai, siekiant užtikrinti pildomų ontologijų korektiškumą ir patikimumą. Pildymas ontologijų redaktoriais neužtikrina įvedamų ontologijos egzempliorių korektiškumo ir reikalauja daug pastangų iš vartotojo pusės.

Darbo tikslas ir uždaviniai:

Tikslas: Užtikrinti korektišką patikimų ontologijos egzempliorių įvedimą, sukuriant tam skirtą metodiką ir informacinę sistemą, kuri leistų pildyti ontologijos duomenis rankiniu ar automatizuotu būdu, taikant šablonus, ir įkelti juos į ontologiją.

Uždaviniai:

1. Išanalizuoti:
   1.1. Ontologijų ir duomenų bazių modelius, palyginti jų savybes;
   1.2. Ontologijos duomenų užpildymo metodus ir įrankius;
   1.3. Semantinei paieškai taikomų ontologijų reikalavimus ir kokybės kriterijus.
2. Sudaryti semantinei paieškai taikomų ontologijų duomenų modelį ir jo pildymo egzemplioriais metodiką
3. Suprojektuoti šią metodiką palaikančią programinę įrangą
4. Realizuoti ontologijų pildymo duomenimis informacinę sistemą
5. Atlikti eksperimentą, kuris leistų įvertinti metodikos ir sukurtos sistemos tinkamumą.

Darbo struktūra:

- Pirmame skyriuje analizuojama ontologija ir jos užpildymo metodai.
- Antrame skyriuje pateikiama ontologijų užpildymo sistema.
- Trečiame skyriuje pateikiama ontologijų užpildymo sistemos realizacija.
- Ketvirtame skyriuje pateikiamas realizacijos aprašymas.
- Penktame skyriuje aprašomas eksperimentas.
- Šeštame skyriuje pateikiamas rezultatų apibendrinimas ir išvados

1. PROBLEMINĖS SRITIES ANALIZĖ

1.1. Analizės tikslas

Šiame skyriuje bus analizuoja ontologija ir ontologijų užpildymo egzemplioriais metodai

1.2. Tyrimo objektas, sritis ir problema

Objektas: Ontologijų užpildymo egzemplioriais procesas, leidžiantis įvesti ontologijos duomenis iš duomenų bazės ar tekstinių dokumentų, parengtų pagal specialius šablonus.

Sritis: Ontologijų kūrimo ir palaikymo metodai, leidžiantys korektiškai užpildyti ir saugoti patikimą ontologijų informaciją

Problema: Semantinės paieškos sistemose tikslinga turėti patikimų ontologijos duomenų, kurie leistų pagerinti paieškos procesą ir pasitarnautų semantinio anotavimo priemonių apmokymui. Šiuos duomenis galima įvesti rankiniu būdu ar išgauti iš esamų duomenų šaltinių. Be to, šiuos duomenis tikslinga papildyti natūralios kalbos žodyno elementais, kurie reikalingi formuluojant semantines užklausas. Tačiau egzempliorių įvedimas ontologijų redaktoriais yra sunkus procesas, kur galima padaryti klaidų, nes ontologijų redaktoriai tikrina tik ontologijos egzempliorių neprieštaringumą, bet netikrina įvedamų duomenų korektiškumo ir vientisumo, kurį užtikrina duomenų bazės.

1.3. Tyrimo objekto (veiklos proceso, modelio, metodo, darinio, sistemos ar jos dalies ir pan.) analizė

1.3.1. Ontologija

Ontologijų kalba skiriasi nuo koncepcinio modeliavimo kalbų, tokių kaip *ER* arba *UML* klasių diagramos, nes turi daugiau galimybių apibūdinti klases ir susidoroti su nepilna informacija.

Ontologijoje neįmanoma užtikrinti vientisumo apribojimų (angl. *Integrity constraints*), kokie yra duomenų bazėse. Dėl to įrašant nepilnus duomenis į ontologijas, trūkstami duomenys laikomi nežinomais, tokie duomenys duomenų bazėse būna atmetami [1].

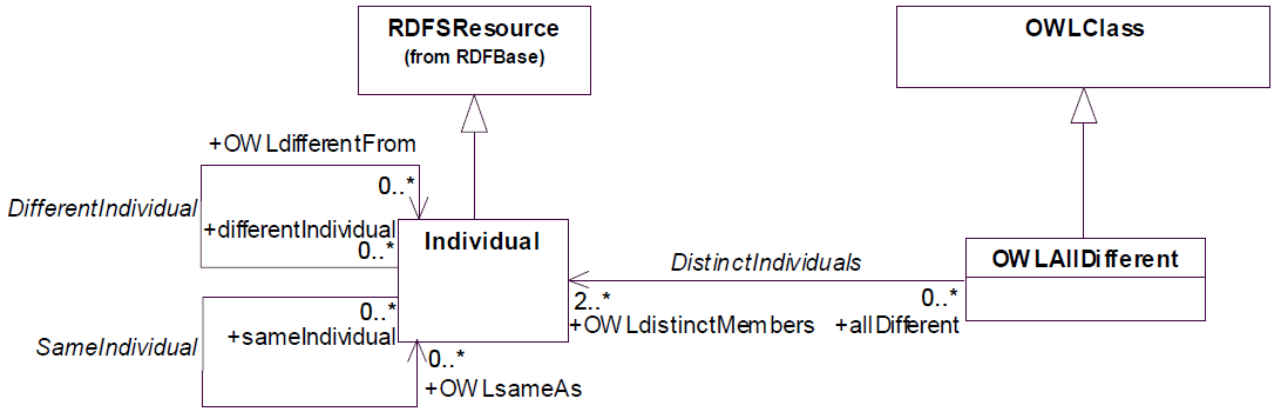
1.3.2. Web Ontology kalbų analizė

Šiuo metu ontologijų kalbos yra pagrįstos išteklių aprašymo karkasu (angl. *Resource Description Framework (RDF)*), naudojamu informacijos apsikeitimui internete. *RDF* yra panašus į esybių diagrama.

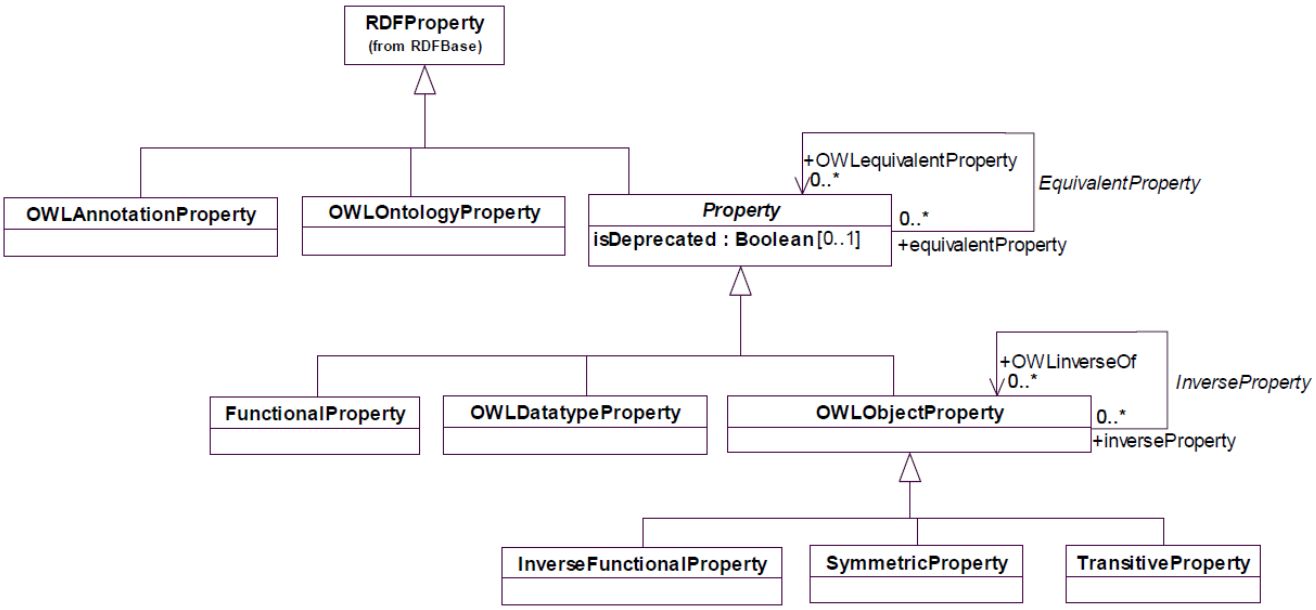
Išplečiant *RDF* semantines galimybes buvo sukurtos dvi kalbos: *RDF* Schema (*RDFS*) ir *Web Ontology Language* (*OWL),* kuri naudojama ontologijose. *OWL* susideda iš klasių, savybių ir individų. Individai vaizduoja objektus norimoje srityje. Klasės yra individų rinkiniai, kurie priskiriami atitinkamoms klasėms. Savybės vaizduoja ryšius tarp individų porų.

Individai (4.1 pav.) ontologijose gali būti aprašomi nurodant [2]:

- tipą, kuris nurodo, kokiai klasei individas priklauso;
- objektines savybes (angl. *object property*)(4.2 pav), kurios nusako ryšius su kitais individais;
- duomenų tipų savybes (angl. *datatype property*)(4.2 pav.), kurios aprašo individų duomenis *XML* schemos duomenų tipais;
- anotacijas, kurios aprašo papildomą informaciją ir metaduomenis.



1.1 pav. *OWL* individų metamodelis [3]



1.2 pav. *OWL* savybių metamodelis [3]

OWL2 yra ontologijos kalba, naudojama ontologijų redaktoriuje *Protégé* ir analizės įrankiuose (angl. *Reasoner*) *RacerPro*, *FaCT++* ir *HermiT*. *OWL2* turi tris variantus: *OWL Lite*, *OWL DL*, *OWL Full*[4].

OWL Lite – naudojama klasifikavimo herarchijoms su paprastais apribojimais. Labai ribota palyginus su *OWL DL* ir *OWL Full*. Leidžia naudoti tik 1 arba 0 kardinalumus.

OWL DL – naudojama norint turėti didžiausia išraiškingumą, prieinamą praktiniams ontologijų analizės (angl. *reasoning*) algoritmams.

OWF Full – turi pilną išraiškingumą ir sintaksinę *RDF* laisvę. Pagristas skirtinga semantika nei *OWL Lite* arba *OWL DL*. *OWL Full* klasės gali būti interpretuojamos kaip individų rinkiniai ir kaip patys individai.

1.3.3. Semantics of business Vocabulary and Business Rules

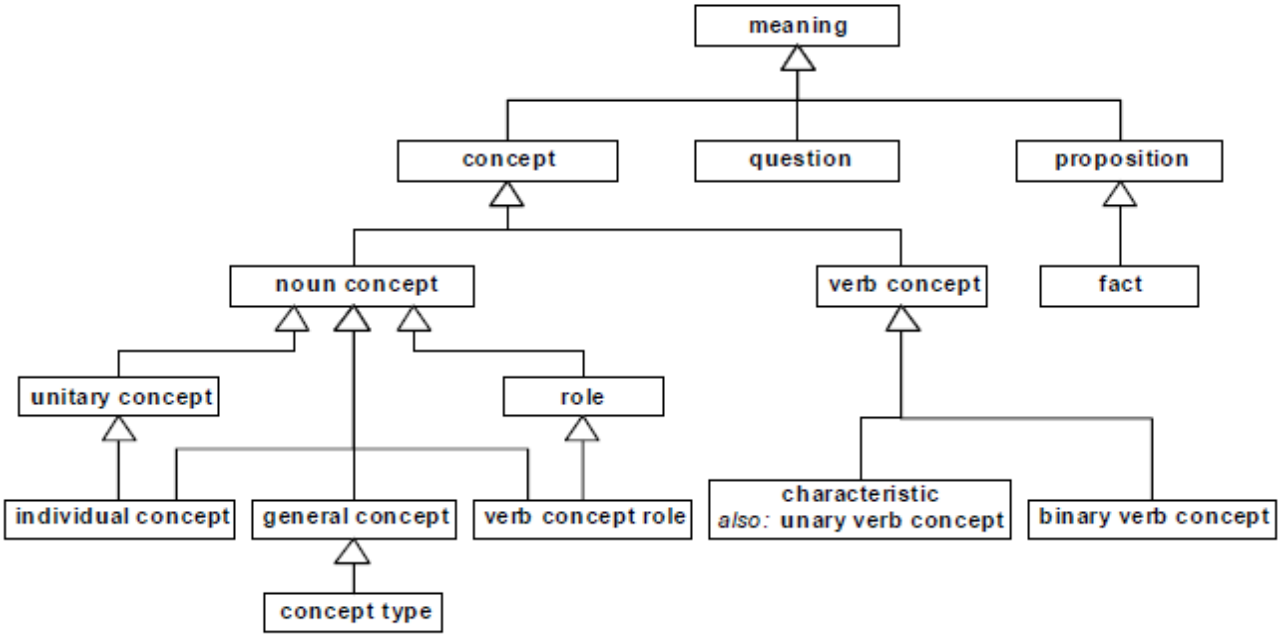
Semantics of business Vocabulary and Business Rules (SBVR) yra galima alternatyva vaizduoti specialių sričių ontologijas (klases bei jų ryšius). *SVBR* nusako žodyną ir taisykles, skirtas dokumentuoti veiklos žodynų, faktų ir taisyklių semantiką.

SBVR yra skirtas modeliuoti veiklą struktūrizuota natūralia kalba. Remiantis *SBVR*, veiklos

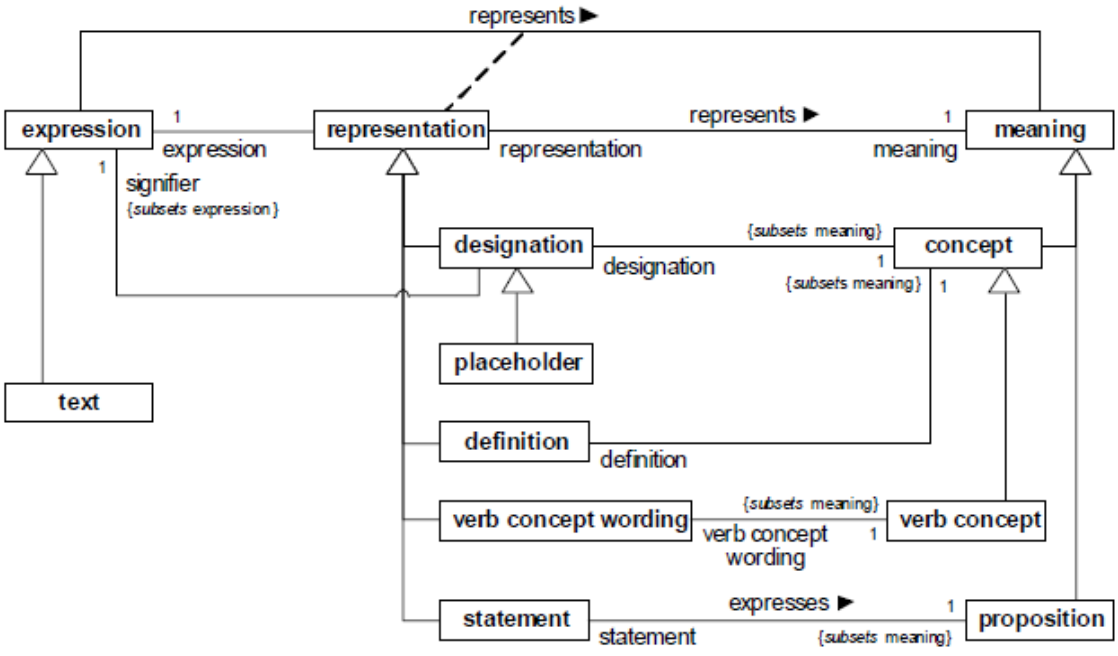
žodynai ir taisyklės išreiškiami struktūrizuota kalba. *SBVR* modelis pagrįstas prasmės (angl. *meaning*) (4.3 pav.) ir jų vaizdavimų (angl. *representation*) (4.4 pav) atskyrimu [5]. *SVBR* žodynas leidžia specifikuoti konceptų, jų ryšių ir apibrėžimų struktūrizuota natūralia kalba vaizdavimus.

SVBR galima laikyti ontologijų specifikavimo kalba nes[6]:

- Apibūdina sritis individais, konceptais ir savybėmis.
- Modeliai vaizduoja aiškiai išreikštas specifikacijas (angl. *explicit specification*).
- Modelis yra skaitomas kompiuterių (angl. *machine readable*).
- Paremtas pirmos eilės logika (angl. *first order logic*).
- Modeliai yra bendro naudojimo, skirti bendruomenėms, ne atskiriems individams.
- Modeliai numatyti turėti universalią apimtį.



1.3 pav. *SBVR* prasmės metamodelis [5]



1.4 pav. *SVBR* vaizdavimo metamodelis [5]

1.1 lentelė. *SBVR* ir *OWL2* konceptai

	SBVR prasmės konceptas	*SBVR* vaizdavimo konceptas	*OWL2* konceptas
1.	Bendrinis konceptas (angl. *General concept*)	Terminas (angl. *Term*)	Klasė (angl. *Class*)
2.	Individualus konceptas	Vardas (angl. *Name*)	Individas
3.	Veiksmažodinis konceptas (angl. *Verb concept*)	Veiksmažodinio koncepto formuluotė (angl. *Verb concept wording*)	Objektinė savybė (angl *Object property*) su apibrėžimo ir kitimo sritimis (angl. *domain* ir *range*) arba jų atliekamais vaidmenimis
4.	Vaidmuo (angl. *Role*)	Terminas	Duomenų savybė (angl. *Data property*)
5.	Veiksmažodinio koncepto vaidmuo (angl. *Verb concept role*)	Terminas	Objektinė savybė su kitimo sritimi (angl. *range*) arba jos atliekamu vaidmeniu
6.	Elementarus konceptas	Terminas	Duomenų tipas

SBVR ir *OWL2* apima tuos pačius prasmės elementus. Ontologijoje laikoma, kad prasmė ir jos vaizdavimo forma sutampa, o *SBVR* ta pati prasmė gali turėti daug formų ir ta pati vaizdavimo forma gali turėti keletą prasmių. Darbe bus siekiama sukurti duomenų bazės schemą, kuri leistų saugoti tuos pačius objektus vieną kartą. *SBVR* turi papildomos informacijos palyginus su *OWL2*, dėl to duomenų bazės schema turėtų būti kuriama pagal *SBVR* metamodelį.

1.4. Esamų problemos sprendimo metodų analizė (Lietuvos ir tarptautiniu mastu)

1.4.1. Ontologijų užpildymo egzemplioriais metodai

1.4.1.1. Metodai, grindžiami tekstų analize

WEB->KB [7]: užpildo ontologijas naršant internetą ir naudojantis *URL* sąrašu kaip pradiniu tašku. Sistemai pateikiama ontologija su sąvokomis ir ryšiais reikalingais užpildymui, bei mokymosi pavyzdžiai apibūdinti sąvokoms ir ryšiams. Naudoja dokumentų klasifikavimo sistema ištisų interneto puslapių atpažinimui ir klasifikavimui.

Adaptiva [7]: vartotojo orientuotų veiksmų sekų mokymosi sistema. Sistema iš rinkinio išrenka pavyzdinius sakinius, kuriuose yra sąvokų pavyzdžiai ir jų tarpusavio ryšiai. Šie pavyzdžiai būna patvirtinami ontologijų eksperto. Patvirtinti pavyzdžiai paverčiami į mokymosi rinkinį.

Artequakt [7]: sistema, kuri automatiškai renka informaciją apie atlikėjus iš interneto su viešos informacijos išrinkimui skirtu *GATE* įrankiu. Ontologijos užpildymas vykdomas dviem etapais. Pirmojo etapo metu visi egzemplioriai, kurie mini tą patį asmenį, bet turi skirtingus faktus, yra sujungiami į vieną išlaikant faktų *URL* adresus. Antrojo etapo metu visi egzemplioriai, kurie turi tas pačias savybes (asmenį, gimimo ir mirties datas) yra sujungiami į vieną.

SOBA [7]: sistema, kuri išrenka informaciją į žinių bazę iš futbolo rungtynių ataskaitų, esančių internete. Išrinkti egzemplioriai tada paverčiami į semantines struktūras, aprašomas ontologijomis. Vertimas taip pat atsižvelgia į tai, kad informacija jau gali būti ontologijoje.

1.4.1.2. Ontologijų užpildymas ontologijų redaktoriais

Protégé: atviro kodo ontologijų redaktorius ir žinių rinkimo sistema, kuriama Stanfordo universiteto kartu su Mančesterio universitetu. *Protégé* turi darbalaukio ir tinklinius klientus.

Redaktorius gali buti pritaikytas kaip žinių rinkimo sistema, naudojanti formų laukus, susietus su ontologijų savybėmis, kas leidžia srities ekspertams tvarkyti ontologijos egzempliorius. Redaktorius palaiko bendradarbiavimą, sekdamas ir rodydamas pakeitimus vartotojams; jiems pranešdamas, kai pakeitimai įvyksta, bei palaikydamas kontekstualizuotas diskusijas [8]. *Protégé* palaiko *RDF/XML, Turtle, OWL/XML, OBO* formatus ir 2 ontologijų modeliavimo būdus [9]:

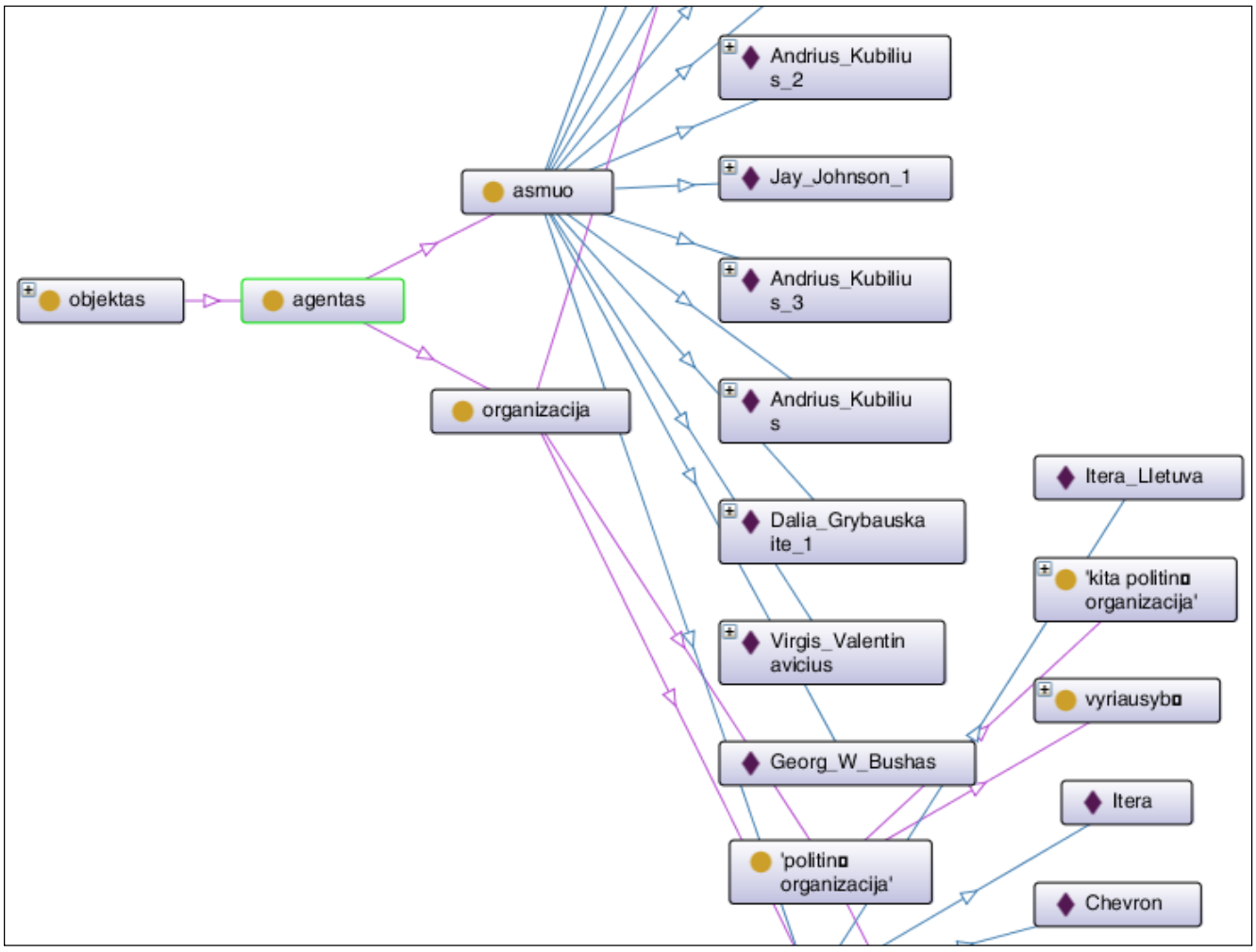
Protégé-Frames: leidžia kurti *frame-based* ontologijas pagal *Open Knowledge Base Connectivity (OKBC)* protokolą.

Protégé-OWL: leidžia kurti ontologijas semantiniam tinklui.

Readaktoriaus vartotojo sąsaja palaiko įprastinius modeliavimo ir vaizdavimo metodus(4.5 pav), bet leidžia ją pritaikyti prie vartotojo poreikių.

Protégé palaiko nemažai pluginų, vienas iš jų yra *Data Master*. Šis puginas atlieka reliacinių duomenų bazių schemų importavimą į *Protégé-OWL* ir *Protégé-Frames* ontologijas. *Protégé-Frames* importavimas apibrėžia duomenų bazės lenteles kaip ontologijos klases, kurios yra lentelių metaklasių egzemplioriai, o stulpelius kaip šablono įrašus. *Protégé-OWL* importavimas skiriasi nuo *Protégé-Frames* tuo, kad šblonų įrašai apibrėžiami kaip klasių anotacijų savybės [10].

Dar vienas pluginas yra *Mapping Master*, skirtas skaityti *Exel* failų duomenis į *OWL* ontologijas. Šis pluginas yra pagrįstas *Manchester* sintakse ir veikia trimis etapais: pirmojo etapo metu sistema apdoroja visas jai duotas išraiškas, aktualus pagal šias išraiškas turinys yra nuskaitomas iš *Excel* failo, šis turinys apkeičiamas su nuorodomis į išraiškas. Antrojo etapo metu paskelbiamos visos nepaskelbtos *OWL* išraiškos norimoje ontologijoje. Trečiojo etapo metu visos išraiškos perduodamos į *Manchester* sintaksės procesorių, kuris užpildo norimą ontologiją[11].



1.5 pav. Klasių hierarchija su individais *Protégé* redaktoriuje

TopBraid [12]: ontologijų redaktorius, atitinkantis *W3C* standartus. Redaktorius palaiko išvedimo ir nuoseklumo tikrinimo mechanizmus ir *OWL* aprašymų logiką su *OWL-DL* analizatoriais. *TopBraid* palaiko *SPARQL* sąsajos notaciją (angl. SPARQL *Inference Notation* (*SPIN*)), kuri gali būti naudojama aprašyti vientisumo apribojimus, padedančius aptikti netinkamus duomenis redagavimo metu.

1.4.1.3. Metodai, grindžiami struktūrizuotais ir pusiau struktūrizuotais duomenų šaltiniais

Metodai, išgaunantys egzempliorius iš duomenų bazių

DB2OWL [13]: Sukuria naujas ontologijas iš duomenų bazių, naujos ontologijos aprašomos *OWL-DL* kalba. Kūrimo metu kiekviena duomenų bazės lentelė paverčiama į klasę, kiekvienas stulpelis į savybę, o apribojimas į ryšį tarp klasių.

Metodai, išgaunantys egzempliorius iš šabloninių dokumentų (*Word, Excel*)

OntoPop[14]: Sistema skirta pildyti ontologijas iš dalinai struktūrizuotų dokumentų. Sistema naudoja *Mondeca's Intelligent Topic Manager* įrankį atvaizduoti ontologijai ir *Temis' Insight Discoverer Extractor* įrankį išrinkti informacijai iš dokumentų. Pirmiausia sistema sulygina duomenų išrinkimo priemonių sukurtas semantines žymes su žinių surinkimo taisyklėmis, kurios aprašo, kaip ontologijos konceptai bus naudojami dokumentui anotuoti. Po to sistema išverčia taisykles į kompiuterių nuskaitomą kalbą. Baigus vertimą, atliekamas validavimas naudojantis pateiktais testiniais dokumentais.

Metodai, paremti įvardytų esybių analize

IBOP[15]: algoritmas kuris kiekvienam kandidatiniam egzemplioriui iš interneto išrenka fragmentus su egzemplioriais. Kiekvienam fragmentui sukuriama hipotetinė frazė sukeičiant egzempliorių su mokomąja reikšme iš duotosios ontologijos, tad hipotetinėms frazėms priskiriamas balas. Pagal priskirtuosius balus egzempliorius priskiriamas prie kategorijos

1.4.2. Ontologijų užpildymo metodų palyginimas

1.2 lentelė. Ontologijų užpildymo metodų palyginimas

Metodas	Egzempliorių patikimumas	Egzempliorių prieinamumas	Automatizavimo laipsnis	Žmogaus darbo sąnaudos įgyvendinus metodą
Metodai, išgaunantys ontologijas remiantis tekstų analize	Nepakankamas	Atviri duomenys	Automatinis	Labai mažos
Metodai, paremti įvardytų esybių (*Named Entity*) analize (Leksinės substitucijos metodas (*IBOP* algoritmas))	Nepakankamas (Apie 60-70%)	Atviri duomenys	Automatinis	Labai mažos
Išgavimas iš reliacinių DB (*RDBToOnto*)[16]	Pakankamas (apie 100%)	Duomenys neprieinami viešam naudojimui	Dalinai automatinis	Vidutinės
Užpildymas taikant ontologijų redaktorių (pvz.,	Gali būti pakankamas, priklauso nuo	Priklauso nuo kūrėjo	Rankinis	Didelės

Metodas	Egzempliorių patikimumas	Egzempliorių prieinamumas	Automatizavimo laipsnis	Žmogaus darbo sąnaudos įgyvendinus metodą
Metodai, išgaunantys ontologijas remiantis tekstų analize	Nepakankamas	Atviri duomenys	Automatinis	Labai mažos
Protégé)	kūrėjo, nes redaktorius neužtikrina ontologijos korektiškumo			
Metodai, išgaunantys individus iš struktūrizuoto teksto	Pakankamas (apie 100%)	Duomenys prieinami ir neprieinami viešam naudojimui	Dalinai automatinis	Vidutinės
Hibridinis metodas (kuriamas šiame darbe)	Pakankamas	Atviri duomenys	Dalinai automatinis	Vidutinės

Leksinės substitucijos metodas naudoja iš anksto suklasifikuotas esybes kaip mokymosi duomenis naujų esybių klasifikavimui.

Išgauti iš reliacinių DB galima naudojantis deklaratyvia sąsaja tarp ontologijos ir duomenų šaltinio arba sukuriant ontologiją iš duomenų bazės koncepcinio modelio.

1.5. Darbo tikslas, uždaviniai ir siekiami privalumai

Tikslas: Užtikrinti korektišką patikimų ontologijos egzempliorių įvedimą, sukuriant tam skirtą metodiką ir informacinę sistemą, kuri leistų pildyti ontologijos duomenis rankiniu ar automatizuotu būdu, taikant šablonus, ir įkelti juos į ontologiją.

Uždaviniai:

6. Išanalizuoti:
   6.1. Ontologijų ir duomenų bazių modelius, palyginti jų savybes;
   6.2. Ontologijos duomenų užpildymo metodus ir įrankius;
   6.3. Semantinei paieškai taikomų ontologijų reikalavimus ir kokybės kriterijus.
7. Sudaryti semantinei paieškai taikomų ontologijų duomenų modelį ir jo pildymo egzemplioriais metodiką
8. Suprojektuoti šią metodiką palaikančią programinę įrangą
9. Realizuoti ontologijų pildymo duomenimis informacinę sistemą
10. Atlikti eksperimentą, kuris leistų įvertinti metodikos ir sukurtos sistemos tinkamumą.

1.6. Siekiamo sprendimo apibrėžimas

Siekiama sukurti ontologijos duomenų bazės modelį ir realizuoti programinę įrangą, kuri leistų korektiškai įvesti ontologijos egzempliorius ir transformuoti juos į ontologijos elementus(4.6pav).

Šiam tikslui reikia:

– sudaryti ontologijos individų modelį;

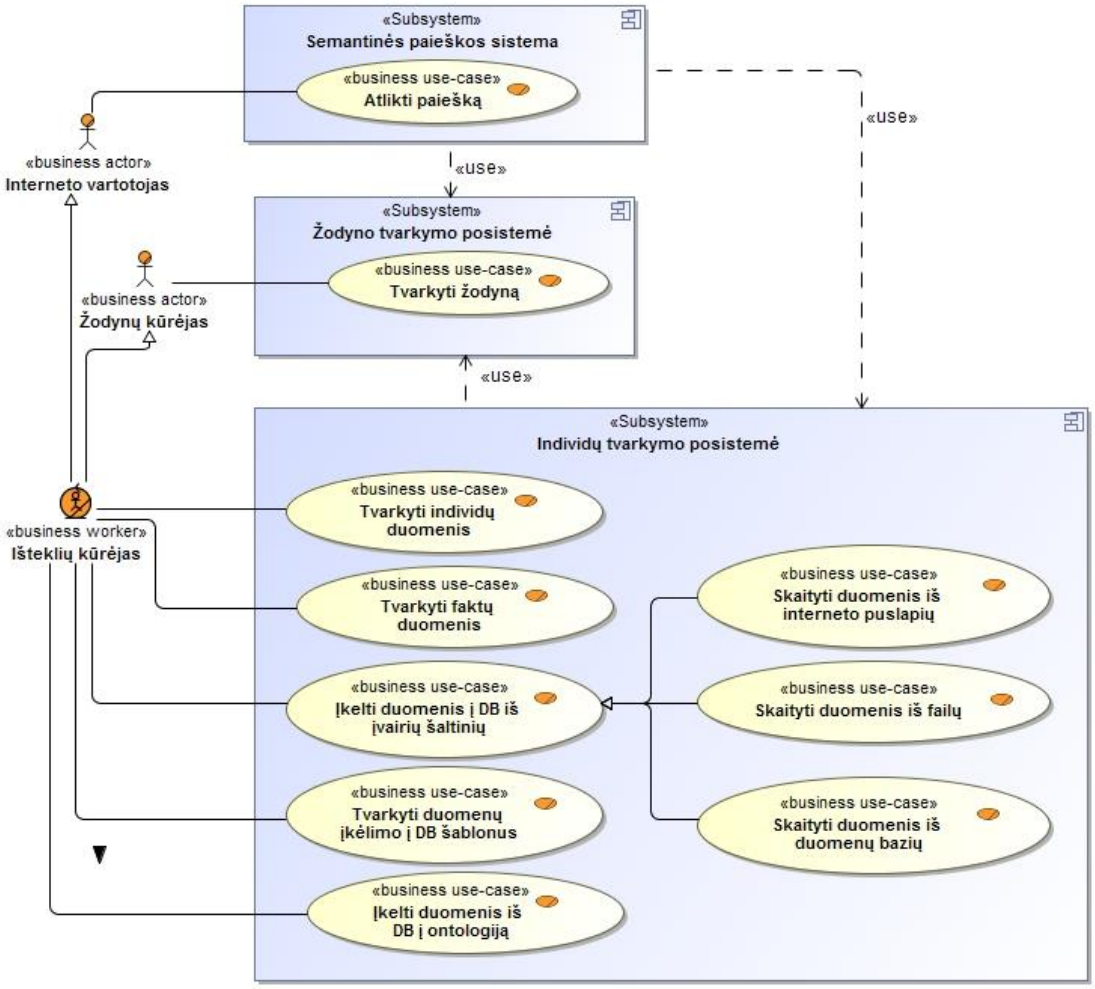
– pasirinkti įėjimo formatų aibę (specialią *RDB* schemą, kuri leistų pildyti egzempliorius rankiniu būdu; tekstinį šabloną);

– sudaryti universalų algoritmą egzemplioriams įkelti į ontologijos schemą;

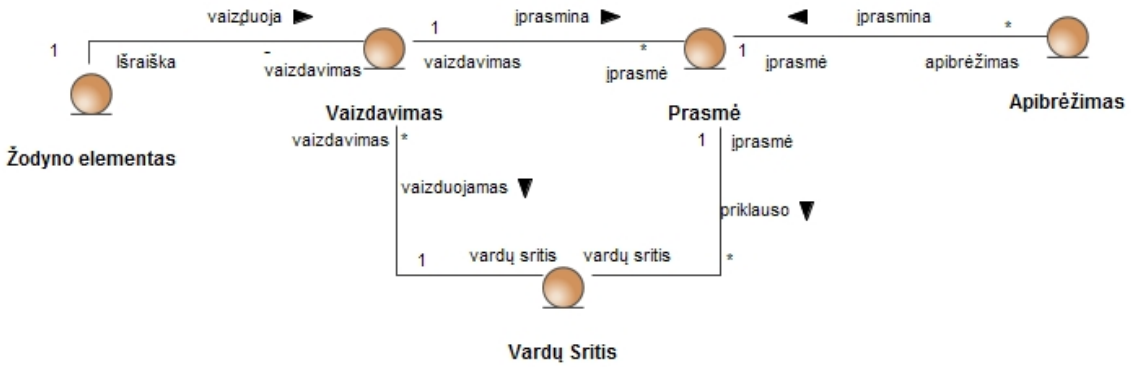
– sukurti taikomąją programą, kuri leistų įvesti egzempliorius rankiniu būdu;

– realizuoti algoritmą individams iš *RDB* ir tekstinio šablono įkelti;

– išbandyti sukurtas programas kelioms pasirinktoms ontologijoms ir įvertinti rezultatus.



1.6 pav. Kuriamos sistemos aplinka ir veiklos panaudojimo atvejų modelis



1.7 pav. Kuriamos sistemos veiklos esybių klasių modelis

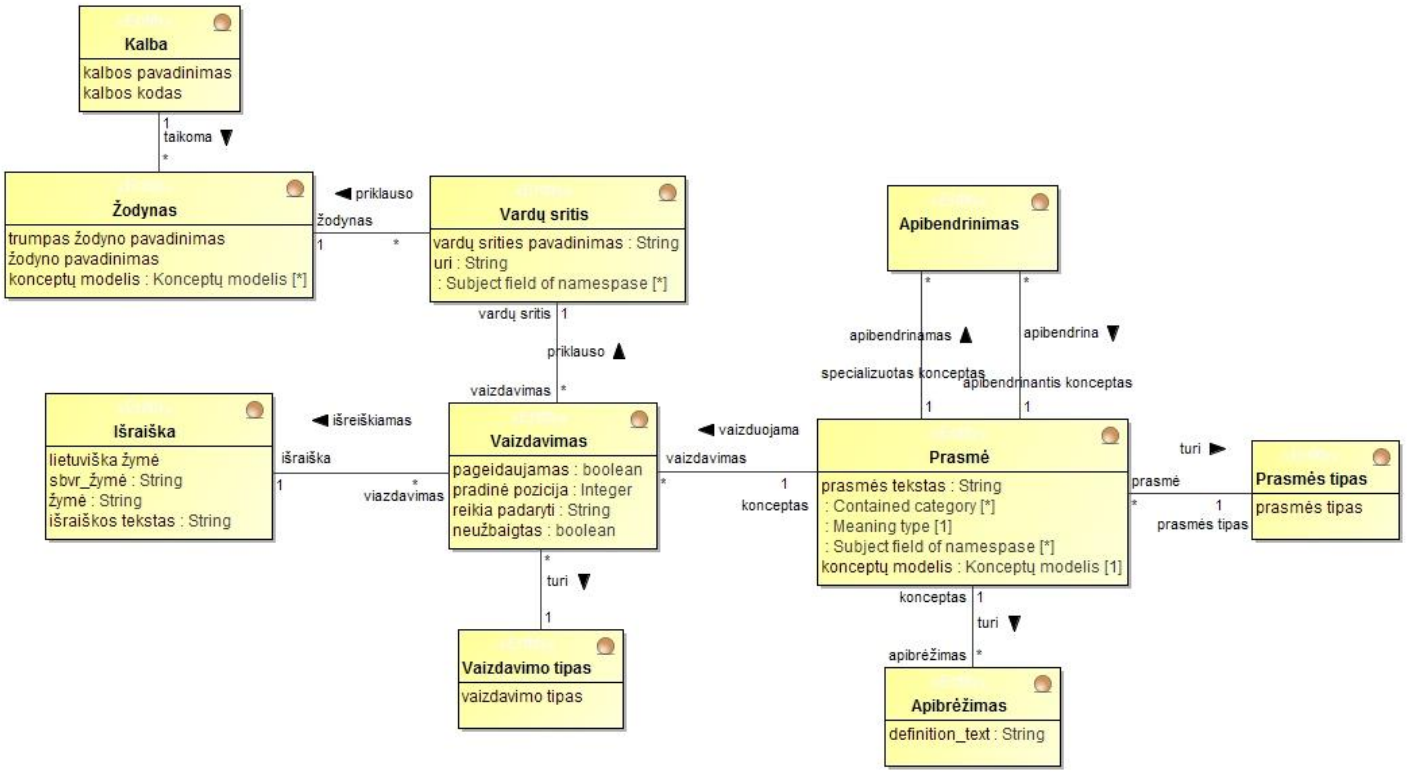
1.7. Analizės išvados

1. Norint pagerinti semantinę paiešką, reikia sukaupti patikimų ontologijų egzempliorių, kurie leidžia gauti geresnius paieškos rezultatus.
2. Ontologijų užpildymo egzemplioriais metodų analizė parodė, kad šiuos metodus galima skirstyti į keturias grupes:
    1) tiesioginis įvedimas per ontologijų redaktorių;
    2) išgavimas iš reliacinių duomenų bazių;
    3) egzempliorių išgavimas iš nestruktūrizuoto teksto;
    4) egzempliorių įvedimas iš dalinai struktūrizuotų dokumentų, taikant šablonus.
2. Naudojant ontologijų redaktorių, sukuriami patikimi duomenys, tačiau toks įvedimas reikalauja daug pastangų.
3. Metodai, išgaunantys egzempliorius iš nestruktūrizuoto teksto, leidžia tą padaryti automatiškai, tačiau lietuvių kalbai pritaikytų metodų nėra, be to, taip sukurti egzemplioriai nėra patikimi.
4. Dėl šių priežasčių šiame darbe bus siekiama sukurti ontologijų užpildymo patikimais egzemplioriais metodiką ir įrankį, kuris leistų kaupti ontologijų egzempliorius pusiau automatizuotu būdu, įvedant juos iš dalinai struktūrizuotų dokumentų ir reliacinių duomenų bazių, bei perkelti į ontologijų dokumentus.
5. *SBVR* metamodelio analizė parodė, kad tokį įrankį tikslinga kurti sudarant duomenų bazės schemą *SBVR* metamodelio pagrindu. Šioje schemoje individai ir jų savybės bus saugomi pagal vieną schemą, iš kurios juos bus galima perkelti į ontologiją taikant tam sukurtą programinį komponentą.

2. INDIVIDŲ TVARKYMO POSISTEMĖS SPRENDIMO REIKALAVIMŲ SPECIFIKACIJA IR PROJEKTAS, FORMALUS APRAŠAS

2.1. Formalus sprendimo aprašas

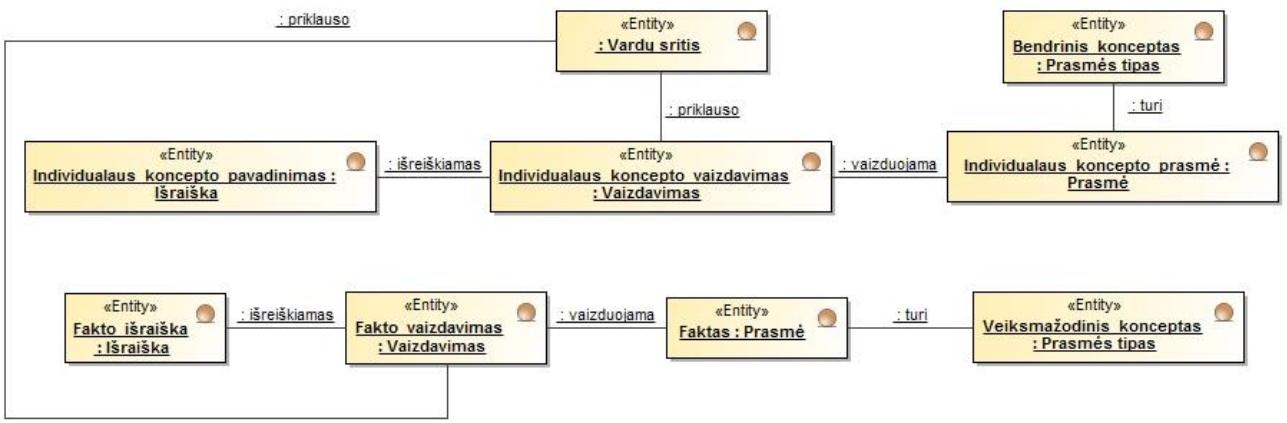
Kuriama sistema duomenis saugos duomenų bazėje sukurtoje pagal *SVBR* metamodelį, kuriame duomenų vaizdavimas yra atskiriamas nuo jų prasmės. Šio metamodelio dalis, kuria remiasi formalus darbo aprašas, parodyta (1.3 pav), (1.4 pav). Šios duomenų bazės koncepcinis modelis pavaizduotas (2.14 pav.). Individų ir faktų egzempliorių modelis pavaizduotas (2.15 pav.).



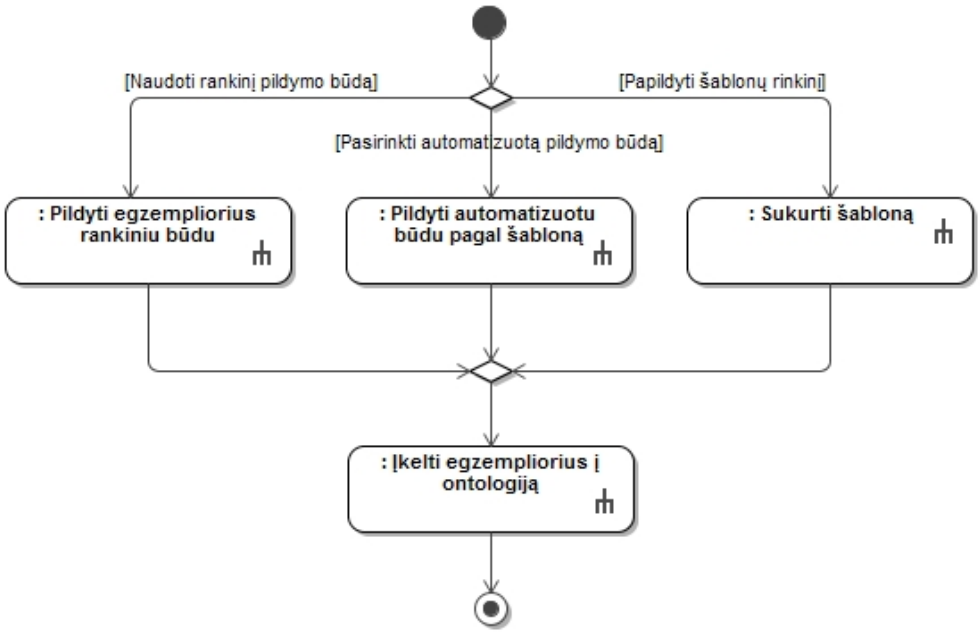
2.14pav Dalykinės srities esybių prasmės ir vaizdavimo modelis

Dalykinės srities modelis atitinka tyrimo srities esybių prasmės ir vaizdavimo formalų aprašą.

Sistema dirbs su žodyno elementu, vaizdavimu, prasme. Žodyno elementai galės būti vardai arba faktai, vaizdavimai galės būti žymėjimai arba veiksmažodinės formos, prasmės galės būti bendriniai, individualūs, veiksmažodiniai konceptai.



2.15 pav. Formalus individualių konceptų ir faktų modelis



2.16 pav. Ontologijos užpildymo patikimais egzemplioriais metodika

Principinė ontologijos užpildymo patikimais egzemplioriais metodika kur parodyti pagrindiniai užpildymo proceso etapai.

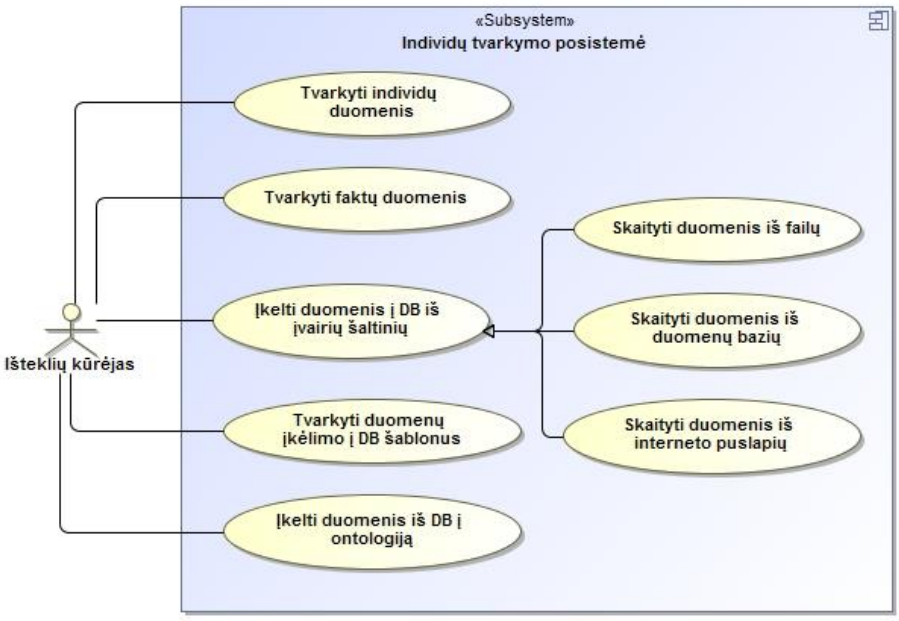
2.2. Reikalavimų specifikacija

Pagrindiniai panaudojimo atvejai sistemoje bus individų duomenų tvarkymas, bei faktų duomenų tvarkymas (3.1 pav.).

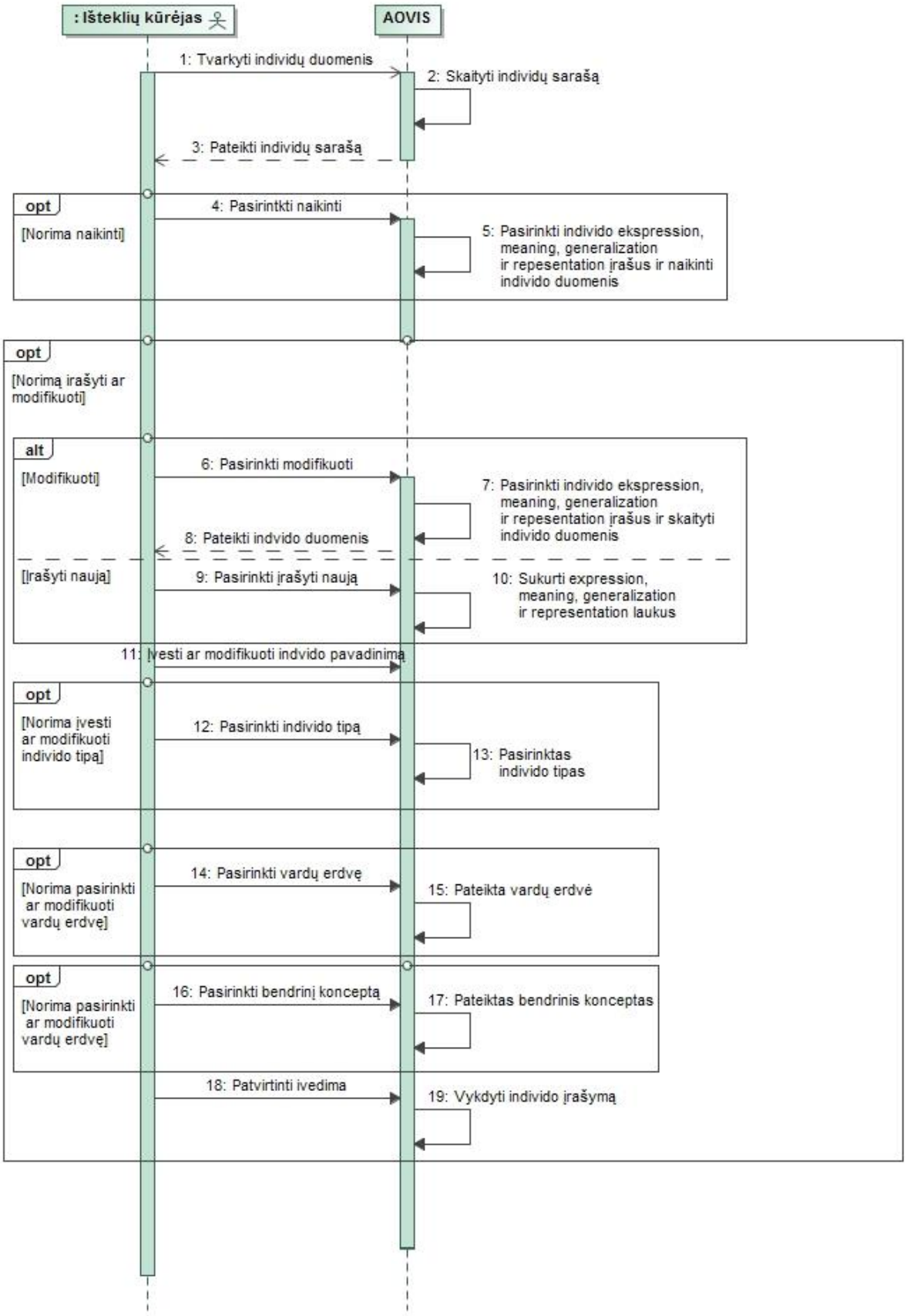
Tvarkant individų duomenis (3.2 pav.), sistema pateiks individų sąrašą ir leis vartotojui įkelti naują individą, modifikuoti arba ištrinti esamą. Įkeliant arba modifikuojant vartotojas nurodys individą, bei vardų erdvę. Sistema pagal duotą informaciją sukurs papildomus duomenis ir įkels į duomenų bazę, jei vartotojas papildomai nurodė individo tipą ar vardų erdvė, sistema taip pat juo įrašys į duomenų bazę, jei ne, bus įvedami esami duomenys arba sukuriami nauji pagal nutylėjimą.

Tvarkant faktų duomenis (3.3 pav.), sistema pateiks faktų sąrašą ir leis vartotojui įkelti naują faktą, modifikuoti arba ištrinti esamą. Įkeliant arba modifikuojant vartotojas įrašys faktą ir nurodys jo tipą, taip pat papildomai gali keisti preffered reikšmę. Sistema pagal duotą informaciją sukurs papildomus duomenis ir įkels į duomenų bazę.

Įkeliant duomenis iš šaltinių (3.5pav), vartotojas nurodys šaltinio tipą, su papildomais duomenimis, šabloną ir pasirinks ar įvedami tik individai ar individai su faktais. Atlikus pasirinkimą sistema nuskaitys individus iš šaltinio ir tikrins ar jie jau neįvesti, jei individas neįvestas, sistema įrašys individą su papildomais duomenimis pagal nutylėjimą, jei įvestas, sistema praleis įvedimą. Po individo tikrinimo sistema tikrina ar vykdomas faktų įvedimas, jei vykdomas, sistema nuskaitys vartotojo įvesta fakto tekstą ir įrašys faktą su individų bei papildomais duomenis pagal nutylėjimą, jei nevykdomas fakto įvedimas, sistema pereis prie sekančio individo. Sistema baigs darbą kai bus perskaitytas paskutinis individas. Tvarkant šablonus sistema pateiks esamų šablonų sąrašą ir leis vartotojui įkelti naują šabloną, modifikuoti arba ištrinti esamą. Įkeliant arba modifikuojant esamą vartotojas nurodys šablono pavadinimą ir sukurs šablono kūną.

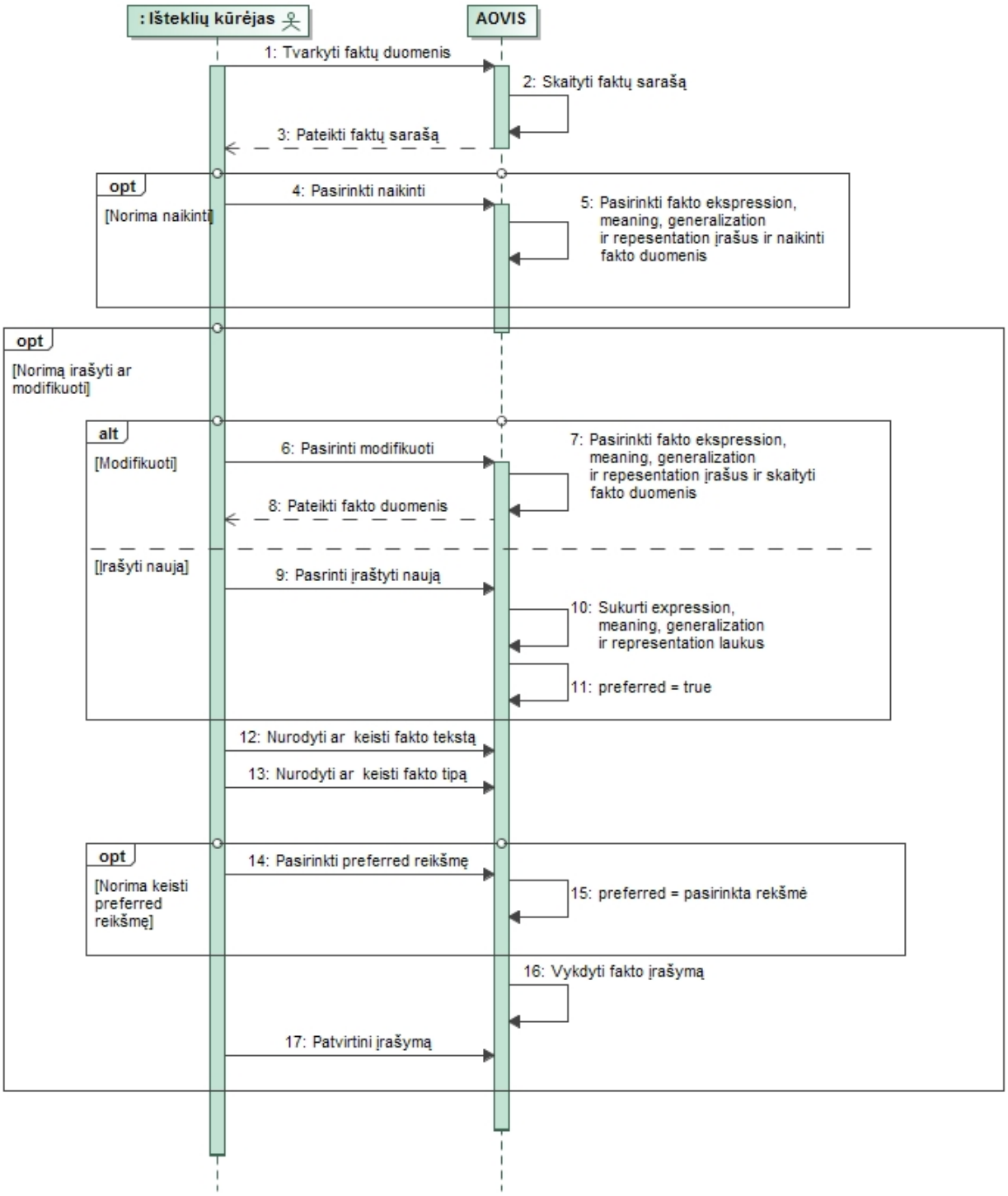


2.1pav Reikalavimų etapo panaudojimo atveju modelis



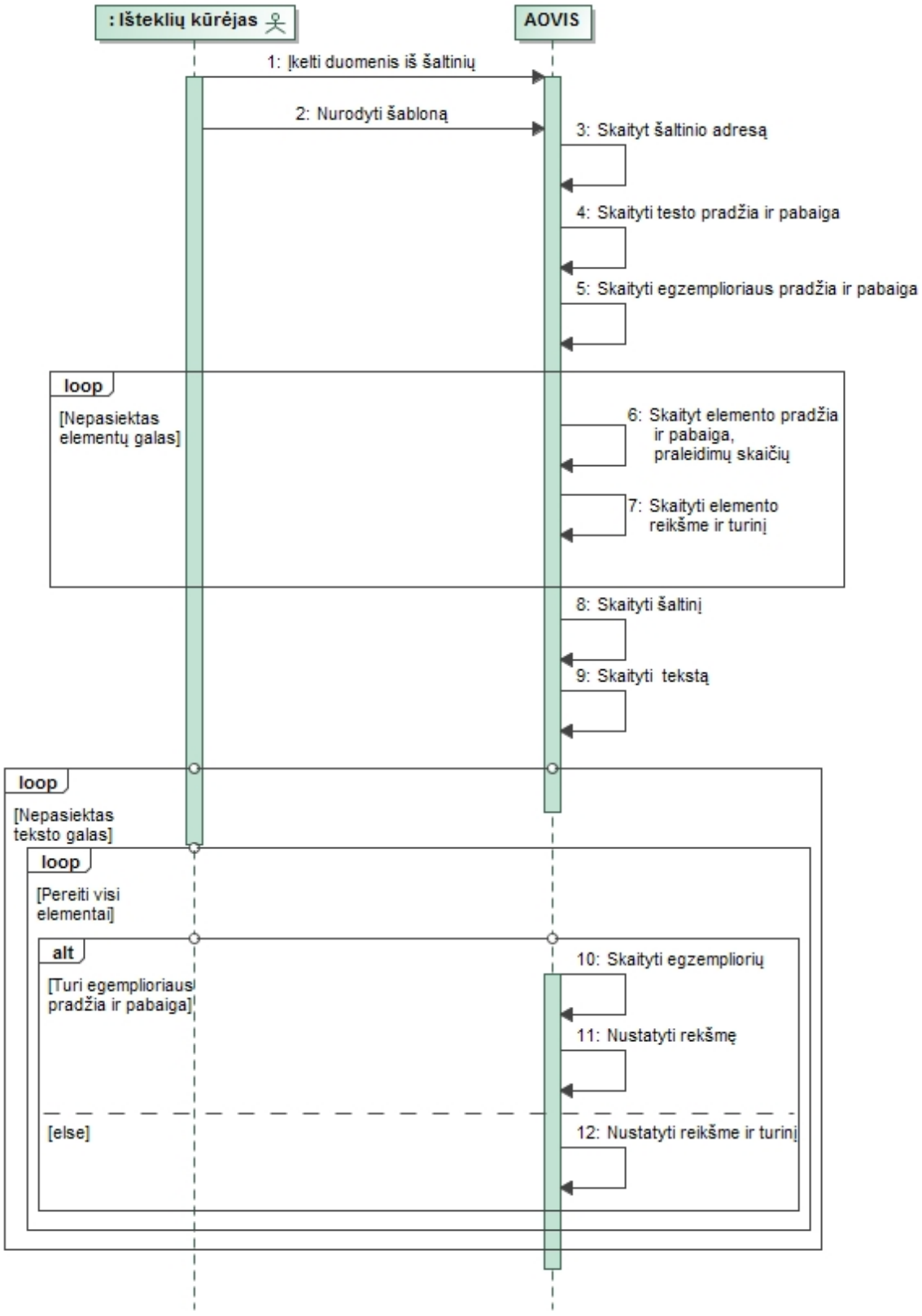
2.2pav Panaudojimo atvejo "Tvarkyti individų duomenis" sekų diagrama

Tvarkant individus sistema pateikia vartotojui esamų individų sąrašą ir leidžia vartotojui trinti, redaguoti arba kurti naujus. Kuriant arba redaguojant vartotojas gali keisti individo vardą, pasirinkti jo tipus, bendrinius konceptus ir vardų erdvę.



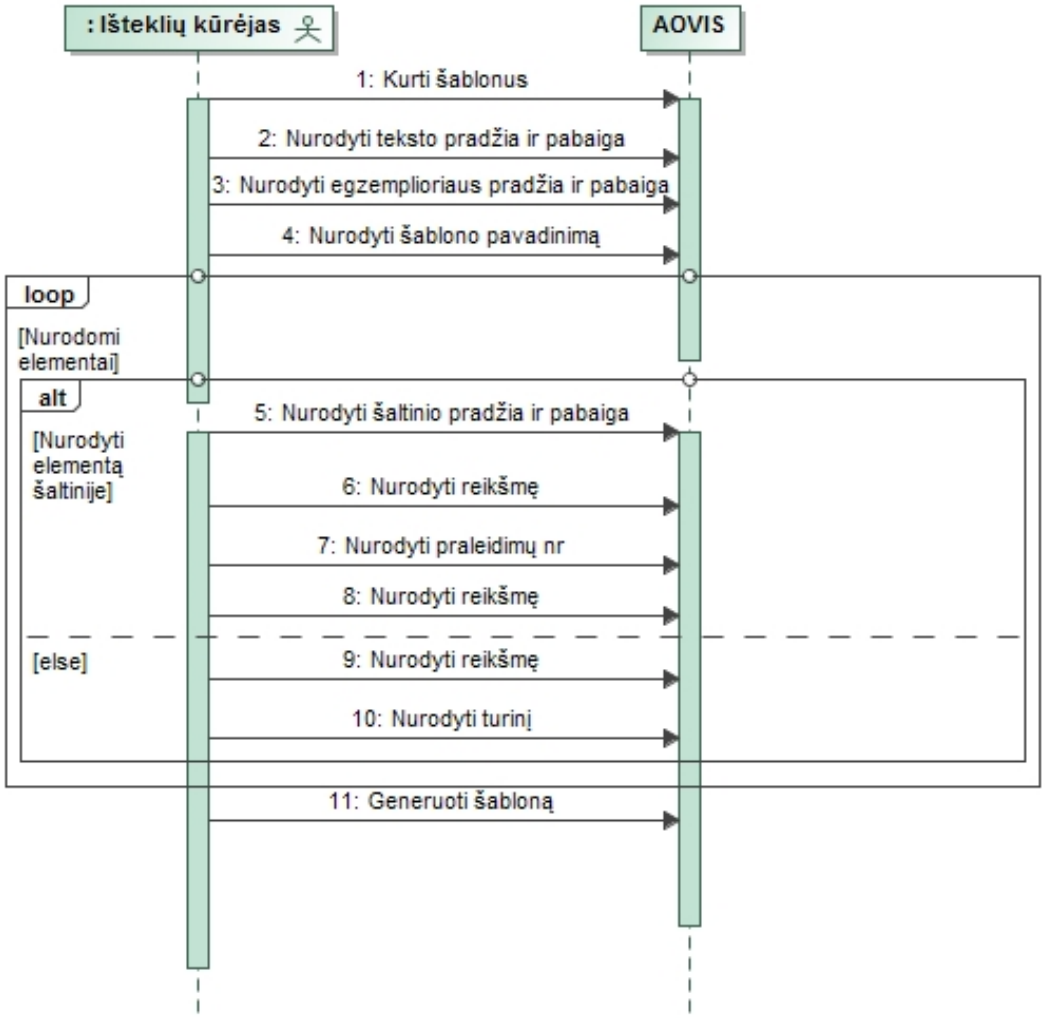
2.3pav Panaudojimo atvejo "Tvarkyti faktų duomenis" sekų diagrama

Tvarkant faktus sistema pateikia vartotojui esamų faktų sąrašą ir leidžia vartotojui trinti, redaguoti arba kurti naujus. Kuriant arba redaguojant vartotojas gali pasirikti fakto tesktą, jo tipus ir pageidautiną reikšmę.



2.4pav Panaudojimo atvejo "Įkelti duomenis iš šaltinių" sekų diagrama

Įkeliant duomenis vartotojas nurodo šabloną. Sistema iš šablono nuskaito šaltinio adresą, teksto pradžia ir pabaiga, egzempliorių pradžia ir pabaiga, ir visus elementus. Nuskaičius visus elementus, sistema nuskaito šaltinį ir kol nepasiekia teksto galo, pereina per visus elementus, jei elementas turi nurodyta elemento pradžia ir pabaiga, sistema nuskaito tekstą iš šaltinio, jei ne, sistema naudoja šablono duomenis



2.5pav Panaudojimo atvejo „Kurti šablonus“ sekų diagrama

Kuriant šablonus vartotojas nurodo teksto pradžia ir pabaiga, egzemplioriaus pradžia ir pabaiga, šablono pavadinimą. Kol reikia nurodyti elementus vartotojas nurodo žodžio pradžia ir pabaiga, praleidimų skaičių ir reikšme, jei renkamas iš šaltinio, arba reikšme ir turinį, jei statinis.

2.1 lentelė. Panaudojimo atvejo „Tvarkyti individų duomenis“ specifikacija

PA „Tvarkyti individų duomenis“		
Tikslas. Leisti vartotojui įkelti individus rankiniu būdu, modifikuoti ir naikinti		
Aprašymas. Šis PA apima individų duomenų įkėlimą, modifikavimą ir naikinimą		
Prieš sąlyga		
Aktorius		Išteklių kūrėjas
Sužadinimo sąlyga		Vartotojas atidaro individų duomenų tvarkymo langą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas atidaro individų duomenų tvarkymo langą		Sistema nuskaito individų duomenis iš duomenų bazės ir pateikia juos vartotojui
2. Vartotojas įveda individo pavadinimą		

3. Vartotojas prasmės tipą	Sistema pateikia esamų tipų sąrašą pasirinkimui
4. Vartotojas parsikreka vardų erdvę	Sistema pateikia esamų vardų erdvių sąrašą pasirinkimui
5. Vartotojas pasirenka bendrinį konceptą	Sistemą pareikia esamų konceptų sąrašą
6.Vartotojas patvirtina įvedimą	Sistema įrašo duomenis į duomenų bazę
Po sąlyga:	
Alternatyvūs scenarijai	
Vartotojas nori modifikuoti individų duomenis	Sistemą pateikia individo įvedimo formą su užpildytais pasirinkto individo duomenimis
Vartotojas nori naikinti individo duomenis	Sistema ištrina pasirinkto individo duomenis iš duomenų bazės

2.2 lentelė. Panaudojimo atvejo „Tvarkyti faktų duomenis“ specifikacija

PA „Tvarkyti faktų duomenis“		
Tikslas. Leisti vartotojui įkelti faktus rankiniu būdu, modifikuoti ir naikinti		
Aprašymas. Šis PA apima faktų duomenų įkėlimą, modifikavimą ir naikinimą		
Prieš sąlyga		
Aktorius		Išteklių kūrėjas
Sužadinimo sąlyga		Vartotojas atidaro faktų duomenų tvarkymo langą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas atidaro faktų duomenų tvarkymo langą		Sistema nuskaito faktų duomenis iš duomenų bazės ir pateikia juos vartotojui
2. Vartotojas nurodo fakto tekstą		
3. Vartotojas pasirenka fakto tipą		Sistema pateikia esamų fakto tipų sąrašą pasirinkimui
4. Vartotojas pasirenka pageidautiną reikšmę		
5.Vartotojas patvirtina įvedimą		Sistema įrašo duomenis į duomenų bazę
Po sąlyga:		
Alternatyvūs scenarijai		
Vartotojas nori modifikuoti faktą		Sistemą pateikia fakto įvedimo formą su užpildytais pasirinkto fakto duomenimis
Vartotojas nori naikinti fakto duomenis		Sistema ištrina pasirinkto fakto duomenis iš duomenų bazės

2.3 lentelė. Panaudojimo atvejo „Įkelti duomenis iš šaltinių“ specifikacija

PA „Įkelti duomenis iš interneto dokumentų“		
Tikslas. Įrašyti pateiktus duomenis iš šaltinio		
Aprašymas. Šis PA vykdomas jei vartotojas nori įrašyti duomenis iš šaltinio		
Prieš sąlyga		Sistemoje turi buti šablonų
Aktorius		Vartotojas
Sužadinimo sąlyga		Vartotojas nori įrašyti duomenis iš šaltinio
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	

Pagrindinis įvykių srautas	Sistemos reakcija ir sprendimai
1.Vartotojas atidaro duomenų įkėlimo langą	
2. Nurodomas šablonas	
Po sąlyga:	
Alternatyvūs scenarijai	

2.4 lentelė. Panaudojimo atvejo „Kurti šablonus“ specifikacija

PA „Kurti šablonus“		
Tikslas. Leisti vartotojui kurti šablonus rankiniu būdu, modifikuoti ir naikinti		
Aprašymas. Šis PA apima šablonų kūrimą, modifikavimą ir naikinimą		
Prieš sąlyga		
Aktorius		Išteklių kūrėjas
Sužadinimo sąlyga		Vartotojas atidaro šablonų tvarkymo langą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykių srautas		Sistemos reakcija ir sprendimai
1. Vartotojas atidaro šablonų tvarkymo langą		
2. Vartotojas įveda teksto pradžią ir pabaigą.		
3. Vartotojas įveda egzemplioriaus pradžia ir pabaiga		
4. Vartotojas įveda adresą		
5. Vartotojas įveda šablono pavadinimą		
6. Vartotojas įveda žodžio pradžia ir pabaigą		
7. Vartotojas pasirenka reikšmę		Sistema pateikia galimų reikšmių sąrašą
8. Vartotojas patvirtina parametrus		Sistema išsaugo teksto pradžia, pabaigą ir reikšmę
9. Vartotojas patvirtina generavimą		Sistema sukuria šabloną
Po sąlyga:		
Alternatyvūs scenarijai		
8a. Vartotojas tęsia parametrų įvedimą		Pereinama į 8-ajį žingsnį
8b Vartotojas baigia parametrų įvedimą		Pereinama į 9-ajį žingsnį

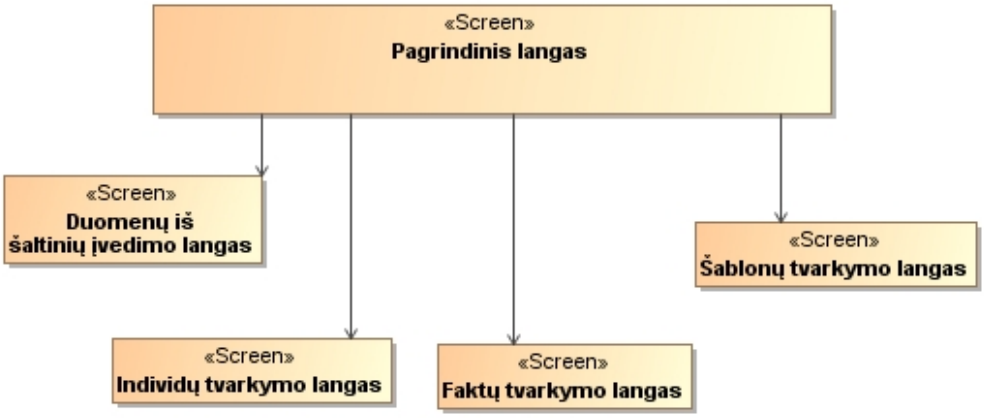
2.3. Dalykinės srities modelis

Dalykinės srities modelis atitinka tyrimo srities esybių prasmės ir vaizdavimo formalų aprašą, pavaizduotą 2.14 paveiksle.

Sistema dirbs su žodyno elementu, vaizdavimu, prasme. Žodyno elementai galės būti vardai arba faktai, vaizdavimai galės būti žymėjimai arba veiksmažodinės formos, prasmės galės būti bendriniai, individualūs, veiksmažodiniai konceptai.

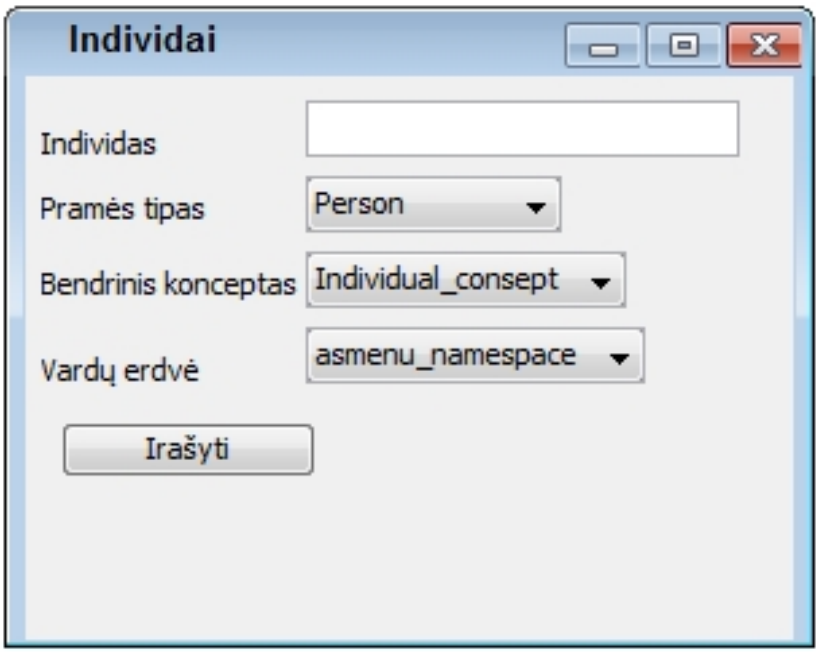
2.4. Naudotojų sąsajos modelis

Prisijungus prie sistemos, vartotojui pateikiamas pradinis langas iš kurio galima pereiti į duomenų iš šaltinių įvedimą, individų tvarkymą, faktų tvarkymą ir šablonų tvarkymą (2.7 pav.).



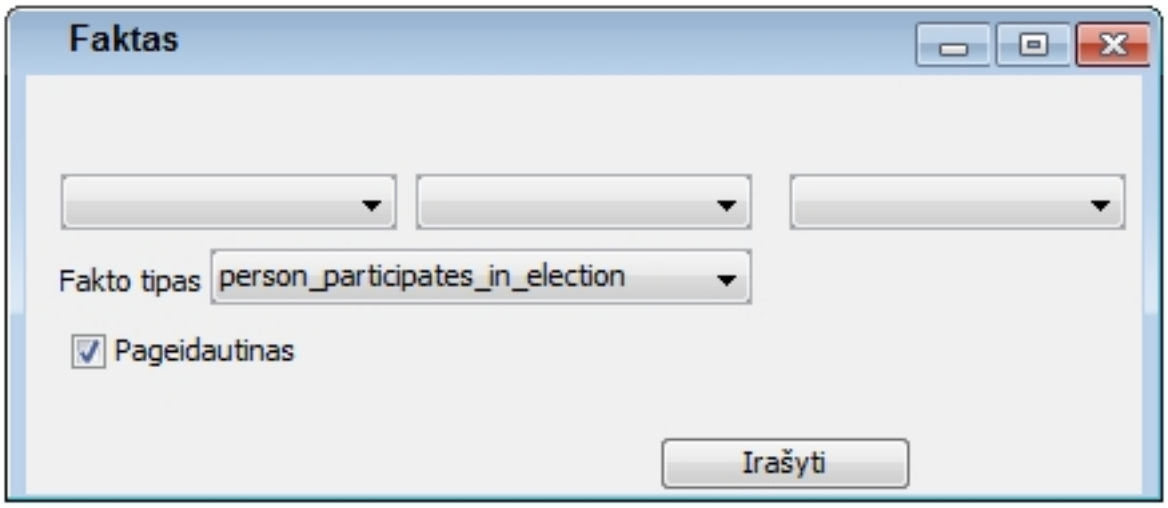
2.7 pav. Vartotojo navigavimo planas

Individų tvarkymo lange vartotojas įrašo individo vardą, pasirenka prasmės tipą, bendrinį konceptą ir vardų erdvę (2.8 pav.).



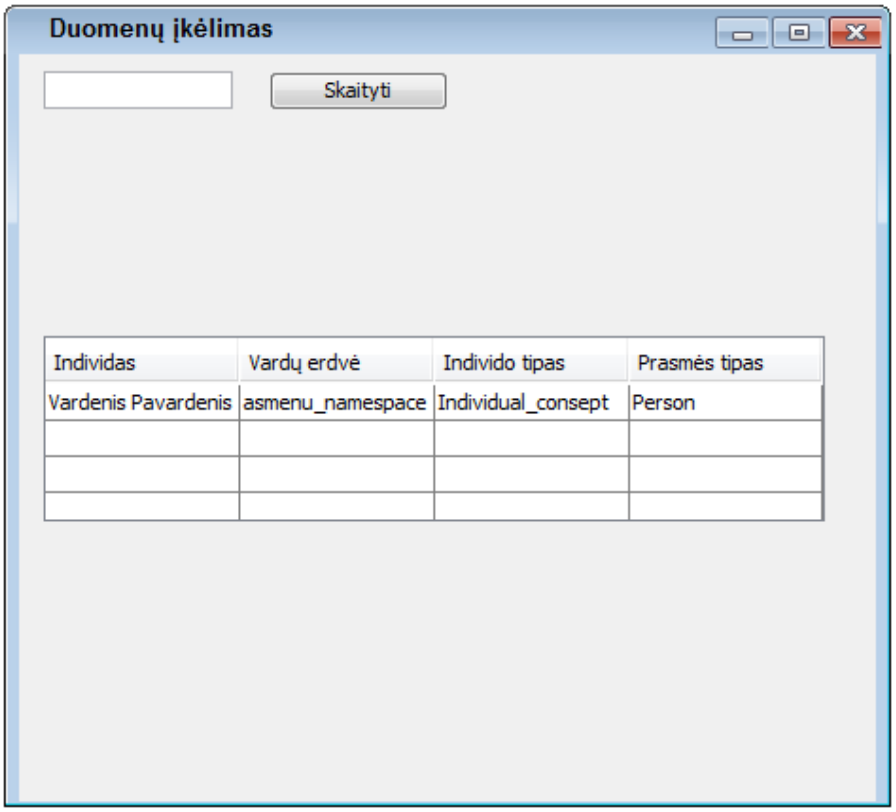
2.8 pav. Individų tvarkymo lango eskizas

Faktų tvarkymo lange vartotojas nurodo fakto tekstą, tipą ir prasmę (2.9 pav.).



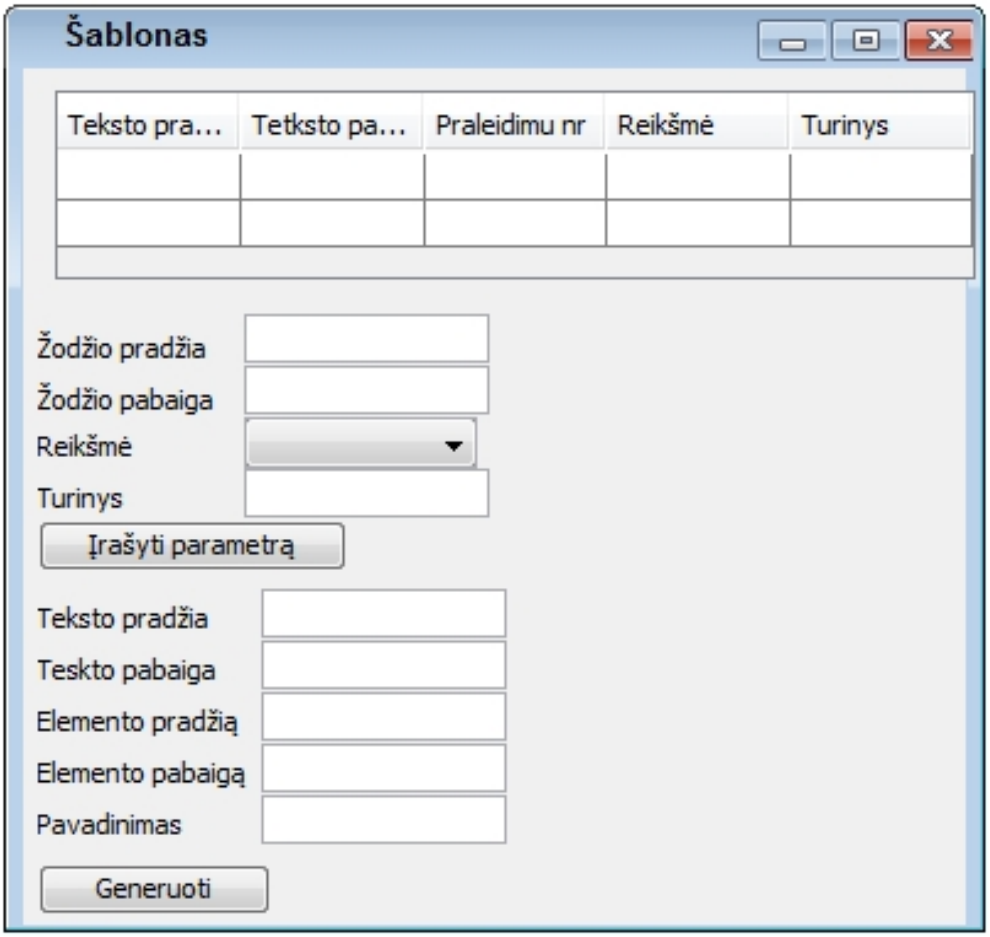
2.9 pav. Faktų tvarkymo lango eskizas

Duomenų įkėlimo lange vartotojas nurodo šabloną, pagal kuri sistema skaitys duomenis ir pateiks rezultatus (2.10 pav.).



2.10 pav. Duomenų įkėlimo iš šaltinių lango eskizas

Šablonu tvarkymo lange vartotojas įrašo teksto pradžią ir pabaigą, elemento pradžią ir pabaigą, šablono pavadinimą, taip pat pildo parametrus: žodžio pradžią ir pabaigą, elemento pradžią ir pabaigą, reikšmę ir turinį (2.11 pav.).



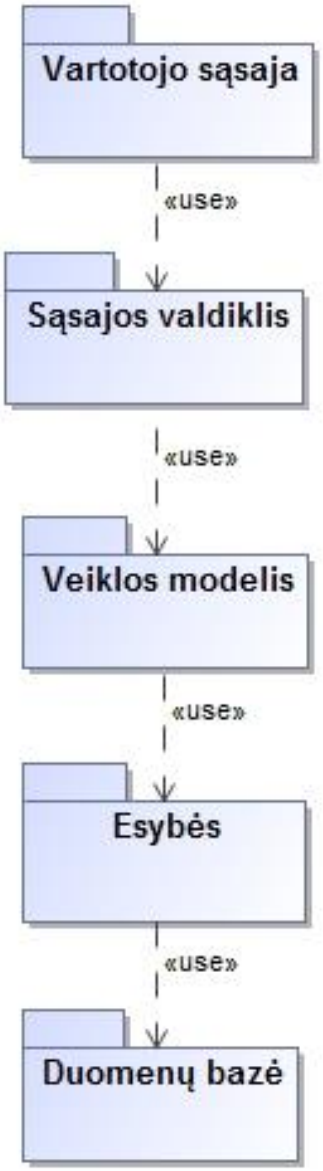
.

2.11 pav. Šablonų tvarkymo lango eskizas

3. INDIVIDŲ TVARKYMO POSISTEMĖS SPRENDIMO REALIZACIJOS PROJEKTAS

3.1. Sistemos architektūra

Sistema bus realizuojama pagal *MVC* architektūrą su *Entity* karkasu. Loginė architektūros schema pateikta 3.1 pav.

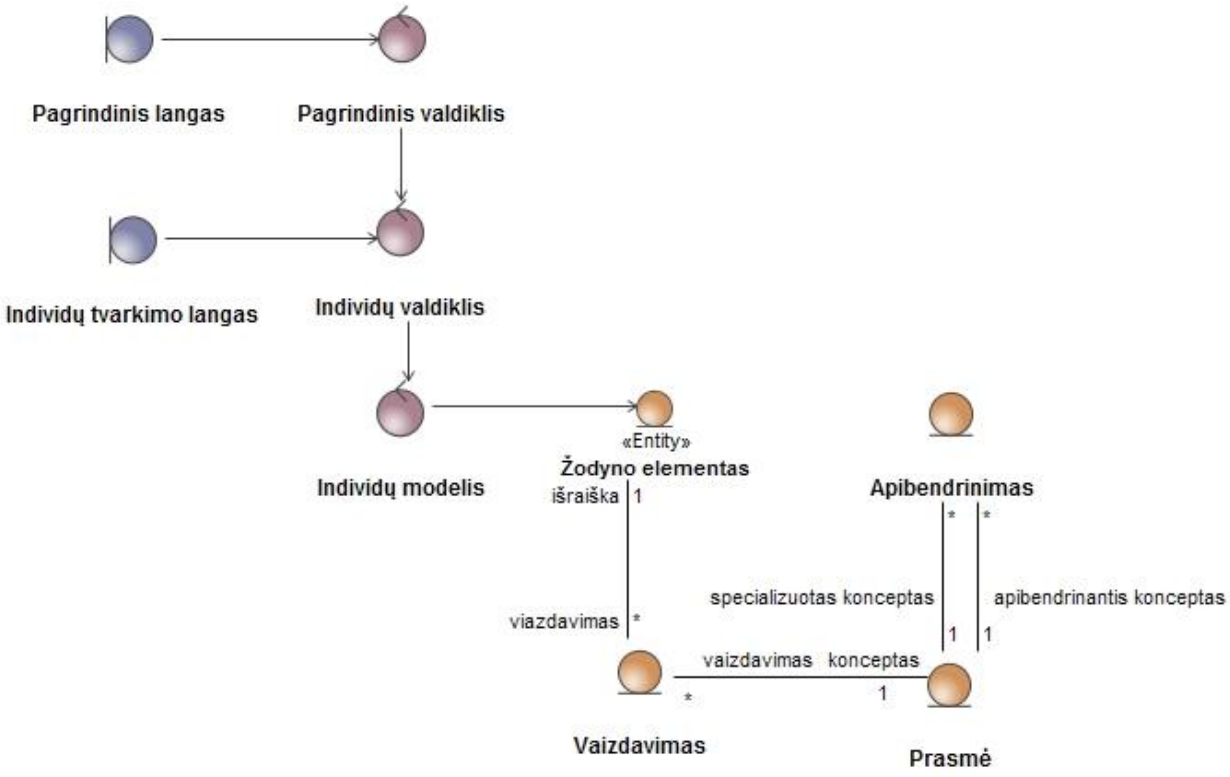


3.1 pav. Sistemos loginė architektūra

.

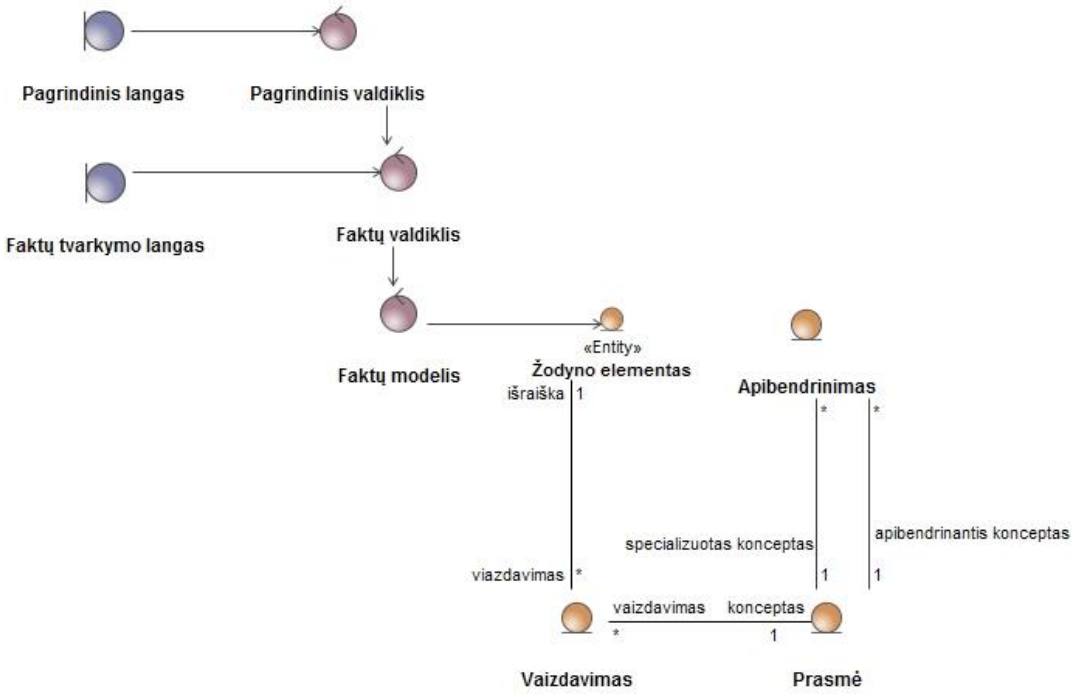
3.1.1. Reikalavimų analizė

Tvarkant individų duomenis naudojamas individų valdiklis, modelis ir žodyno elemento, vaizdavimo, prasmės, apibendrinimo esybės (3.2 pav.).



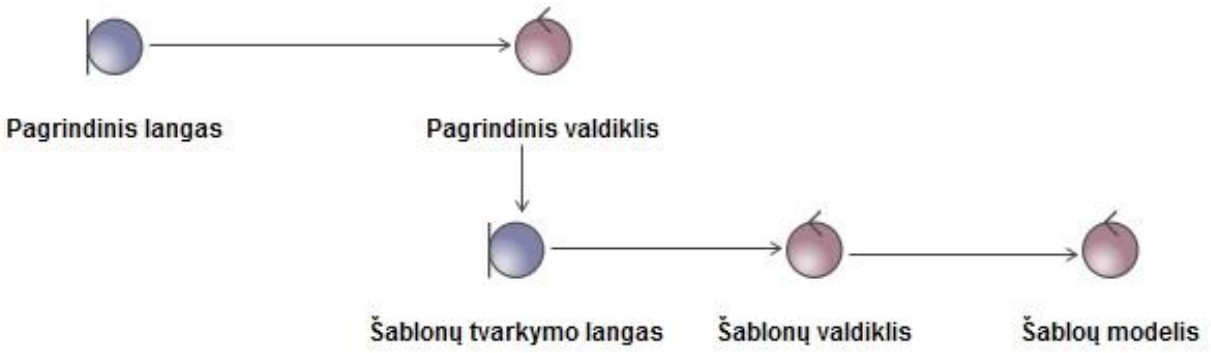
3.2 pav. „Tvarkyti individų duomenis“ analizės klasės

Tvarkant faktų duomenis, naudojamas faktų valdiklis, modelis ir žodyno elemento, vaizdavimo, prasmės, apibendrinimo esybės (3.3 pav.)..



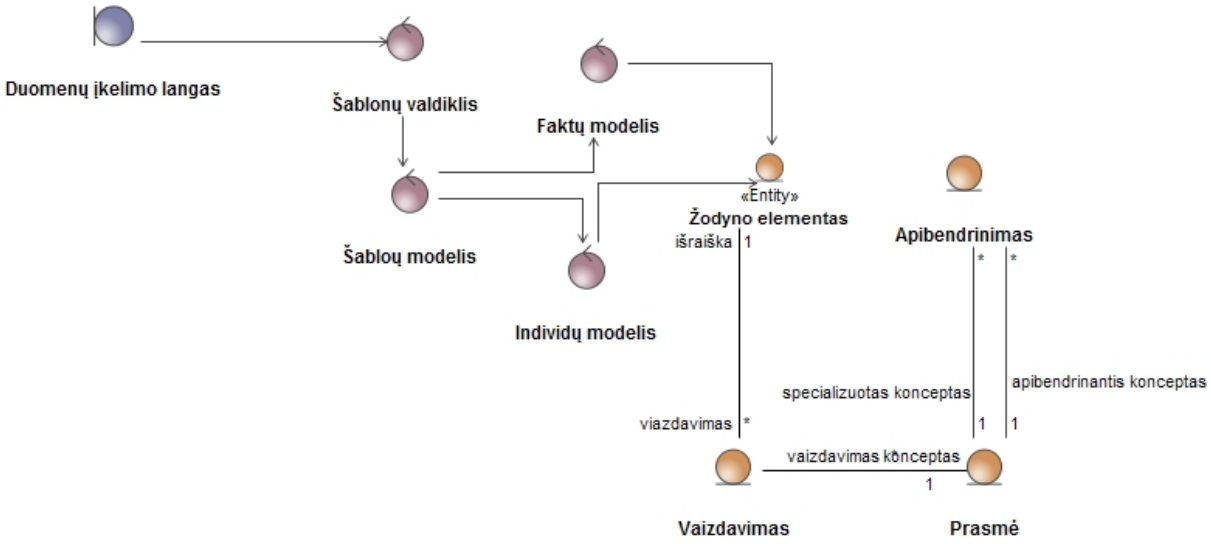
3.3 pav. „Tvarkyti faktų duomenis“ analizės klasės

Tvarkant šablonus naudojamas šablonų modelis ir valdiklis (3.4 pav.).



3.4 pav. „Tvarkyti šablonus“ analizės klasės

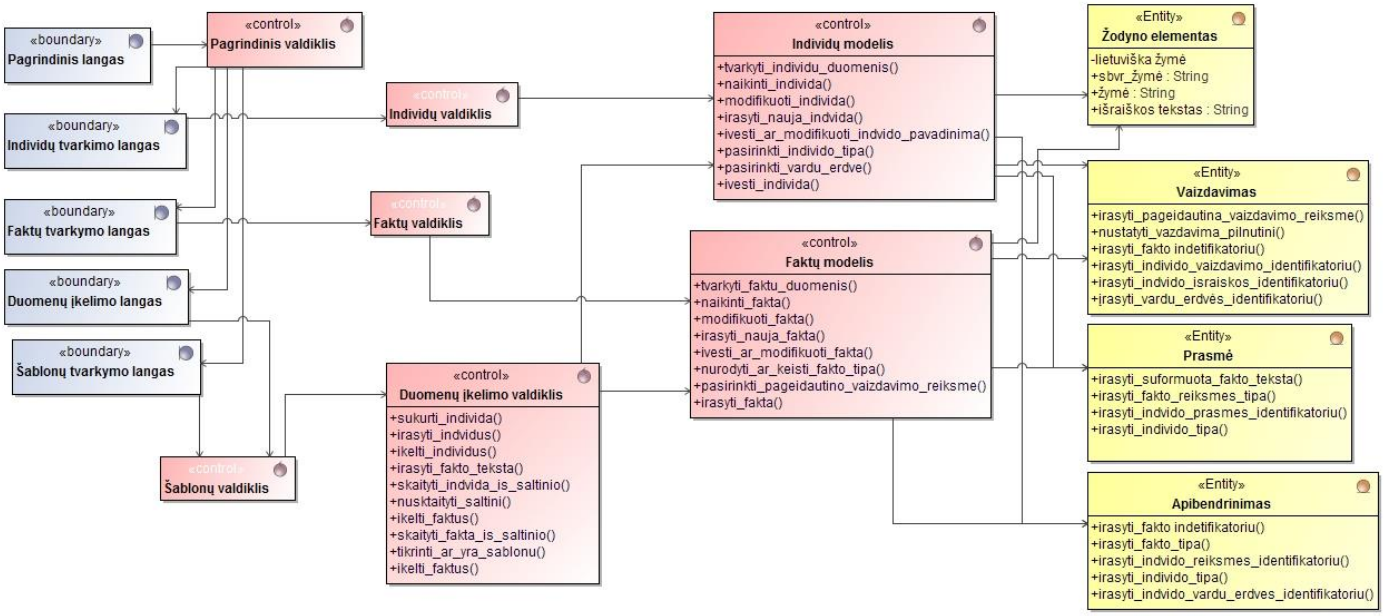
Įkeliant individų duomenis iš šaltinių, naudojamas šablonų modelis, šablonų valdiklis, faktų modelis, individų modelis ir žodyno elemento, vaizdavimo, prasmės, apibendrinimo esybės (3.5 pav.).



3.5 pav. „Įkelti duomenis iš šaltinio“ analizės klasės

3.1.2. Loginė visos sistemos architektūra

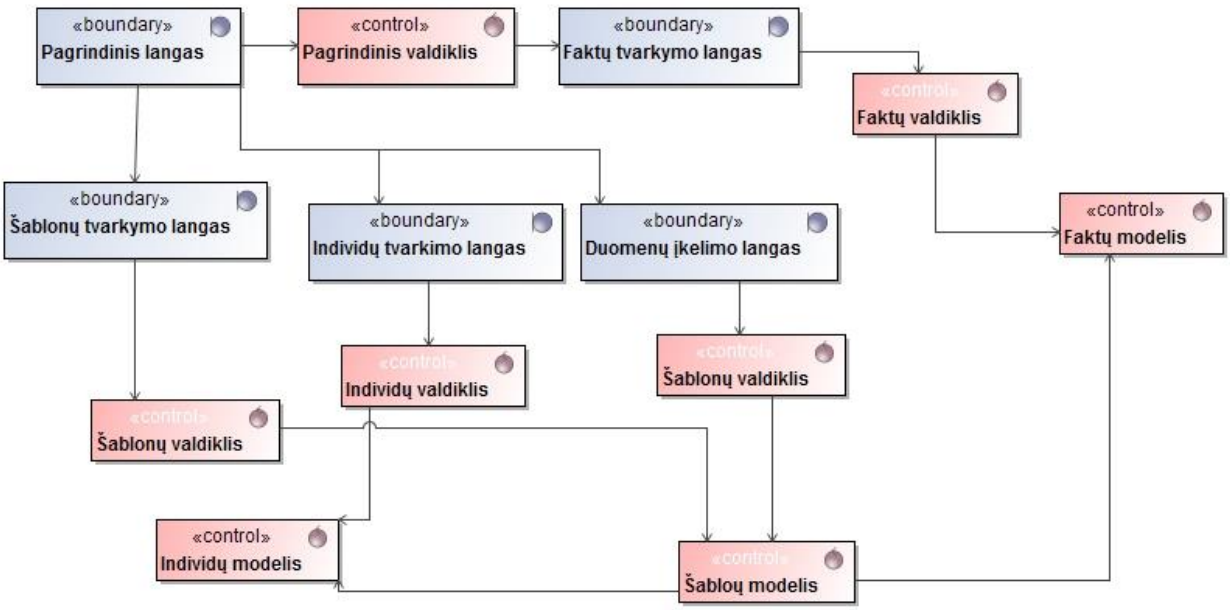
Loginė visos sistemos architektūra pateikta 3.6 pav. Sistema turi 5 langus ir 4 valdiklius (šablonų langai turi bendrą valdiklį), 3 modelius ir dirba su 4 duomenų bazės esybėmis.



3.6 pav. Klasių diagrama

3.1.3. Vartotojo sąsajos klasių modelis

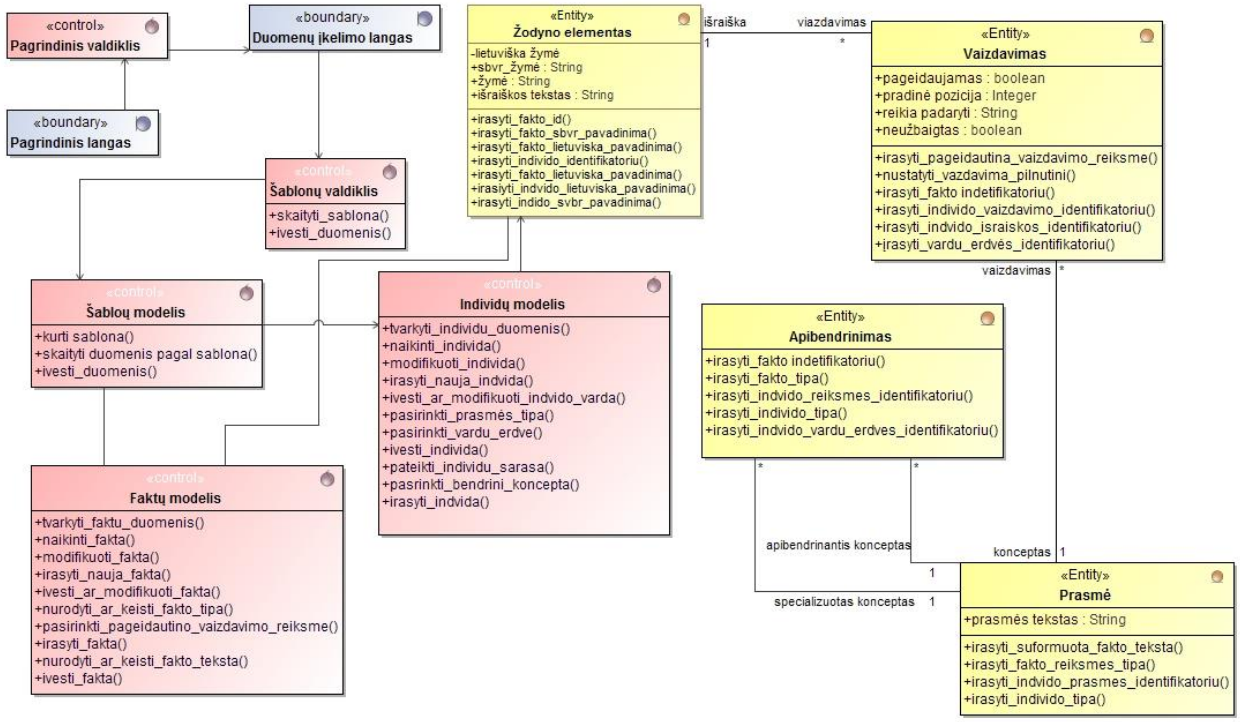
Vartotojo sąsaja susideda iš 5 langų, valdomų 4 valdiklių (šablonų langai turi bendrą valdiklį) dirbančių su 3 modeliais (3.7 pav.):



3.7 pav. Vartotojo sąsajos klasių modelis

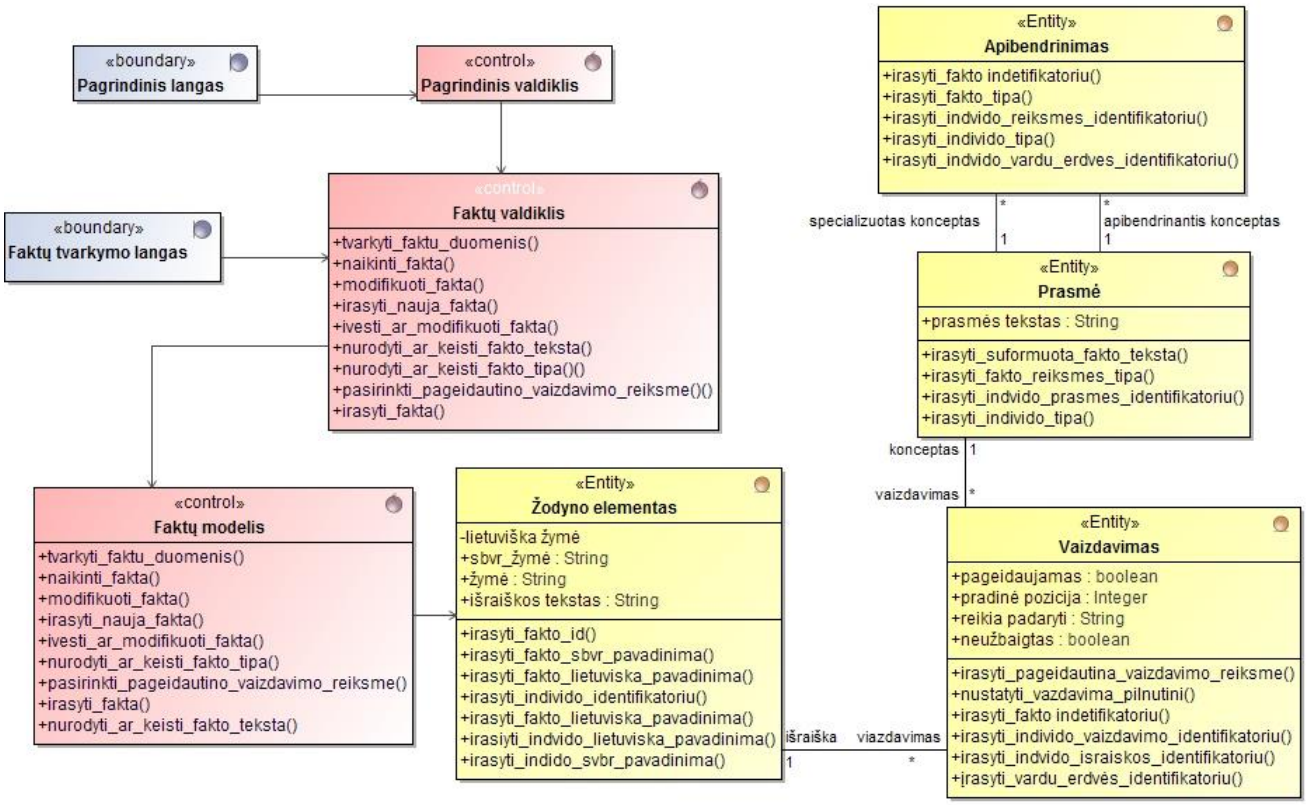
3.1.4. Veiklos logikos (valdymo ir esybių klasių) modelis

Įkeliant individų modelį iš šaltinio naudojamas šablonų modelis, šablonų valdiklis, faktų modelis, individų modelis ir žodyno elemento, vaizdavimo, prasmės, apibendrinimo esybės (3.8 pav.).



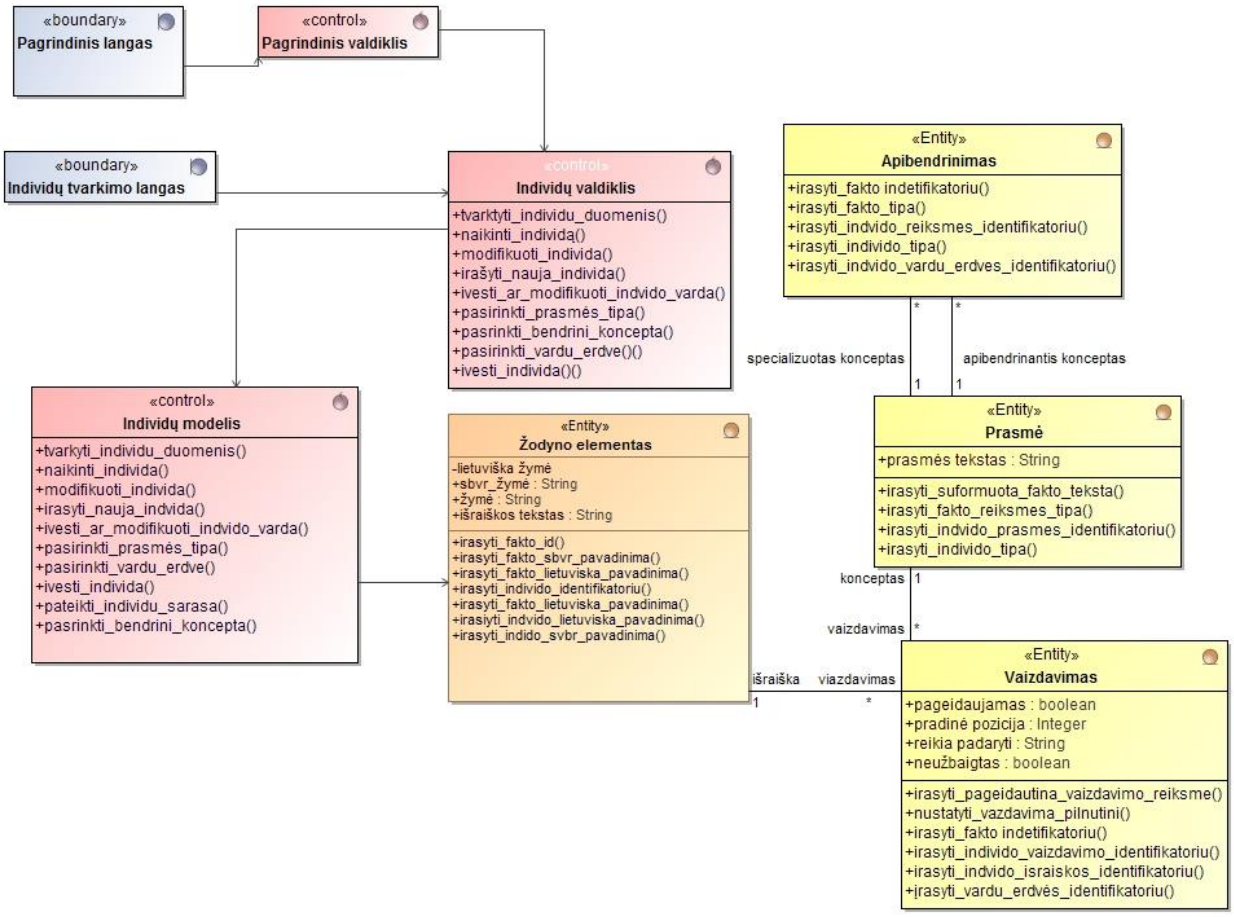
3.8 pav. „Įkelti duomenis iš šaltinių" vartotojo sąsajos ir veiklos logikos klasės

Tvarkant faktų duomenis, naudojamas faktų valdiklis, modelis ir žodyno elemento, vaizdavimo, prasmės, apibendrinimo esybės (3.9 pav.).



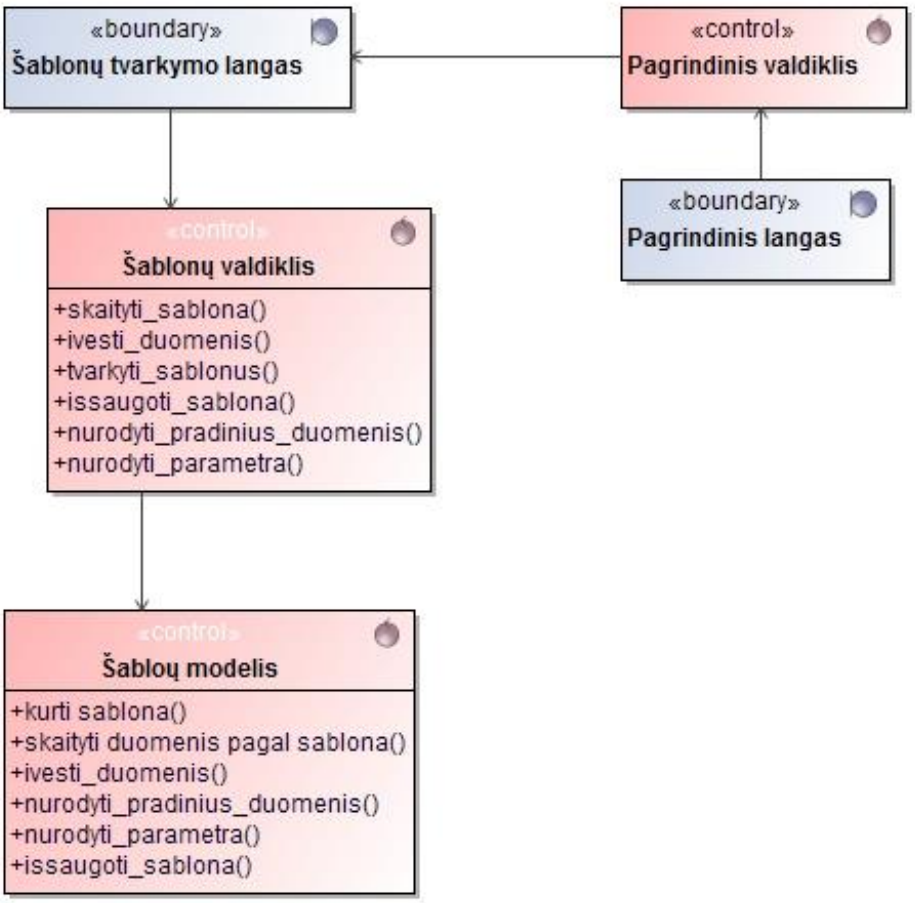
3.9 pav. „Tvarkyti faktų duomenis" vartotojo sąsajos ir veiklos logikos klasės

Tvarkant individų duomenis, naudojamas individų valdiklis, modelis ir žodyno elemento, vaizdavimo, prasmės, apibendrinimo esybės (3.10 pav.)



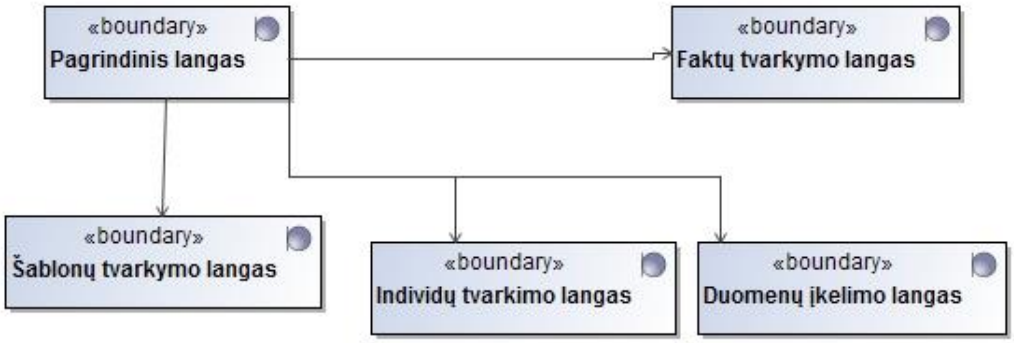
3.10 pav. „Tvarkyti individų duomenis" vartotojo sąsajos ir veiklos logikos klasės

Tvarkant šablonus naudojamas šablonų modelis ir valdiklis (3.11 pav.).



3.11 pav. „Tvarkyti šablonus" vartotojo sąsajos ir veiklos logikos klasės

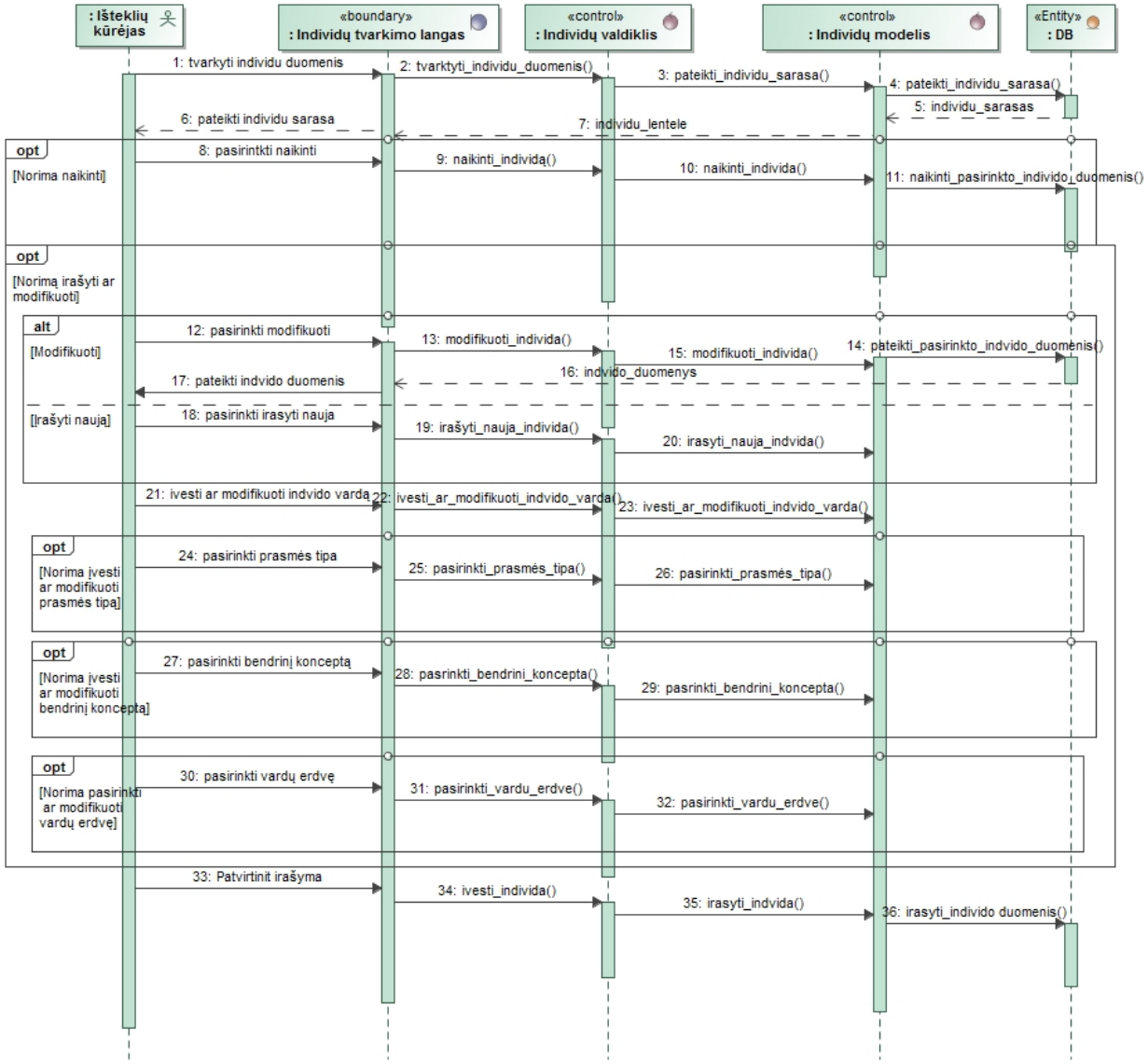
Sistema turi langus individų, faktų ir šablonų tvarkymui bei duomenų iš šaltinių skaitymui, kurie pasiekiami per pagrindini langą (3.12 pav.).



3.12 pav. Vartotojo sąsajos veiklos logikos klasės

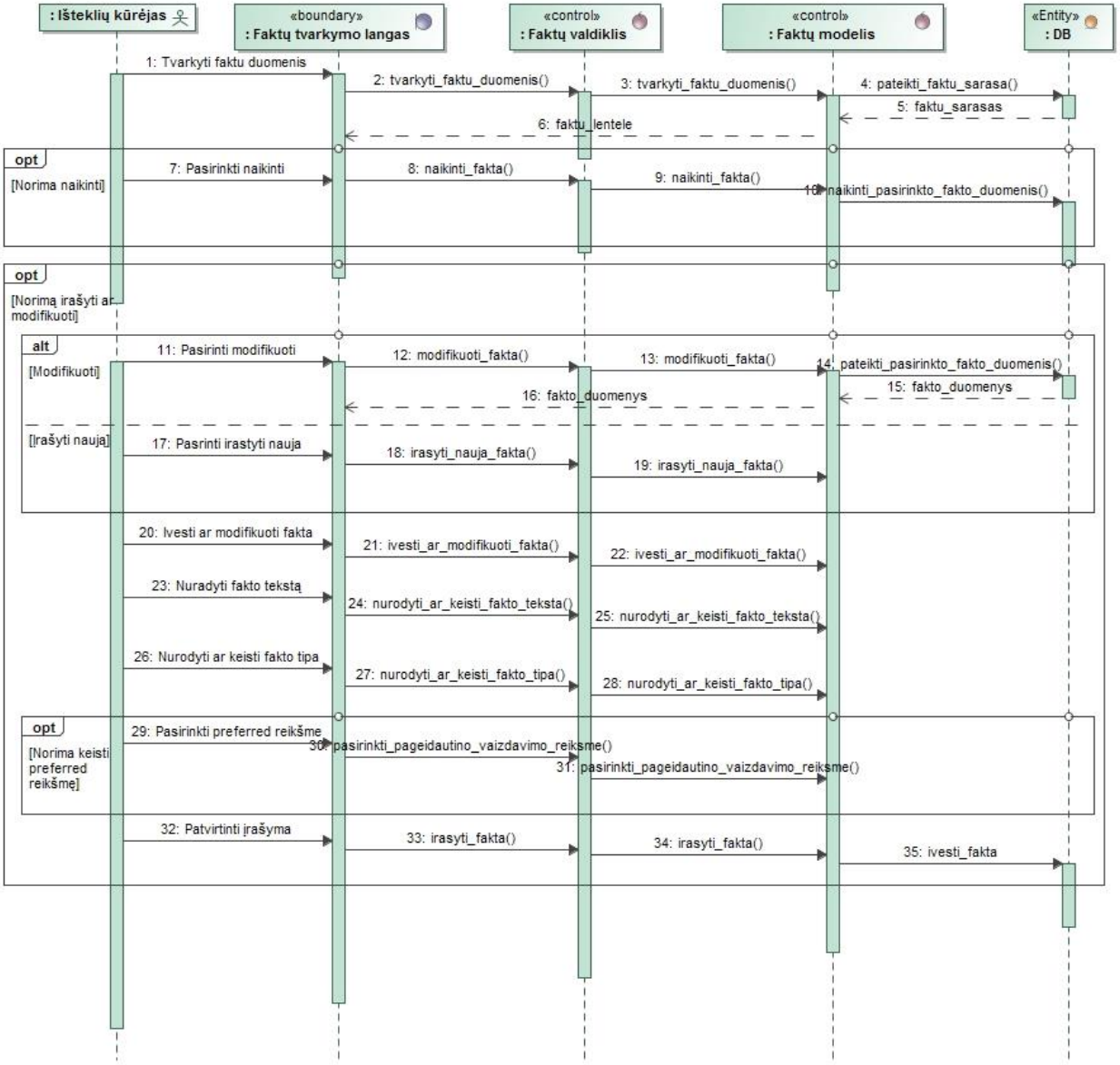
3.2. Sistemos elgsenos modelis

Tvarkant individus sistema pateikia vartotojui esamų individų sąrašą ir leidžia vartotojui trinti, redaguoti arba kurti naujus. Kuriant arba redaguojant vartotojas gali keisti individo vardą, pasirinkti jo tipus, bendrinius konceptus ir vardų erdvę (3.13 pav.).



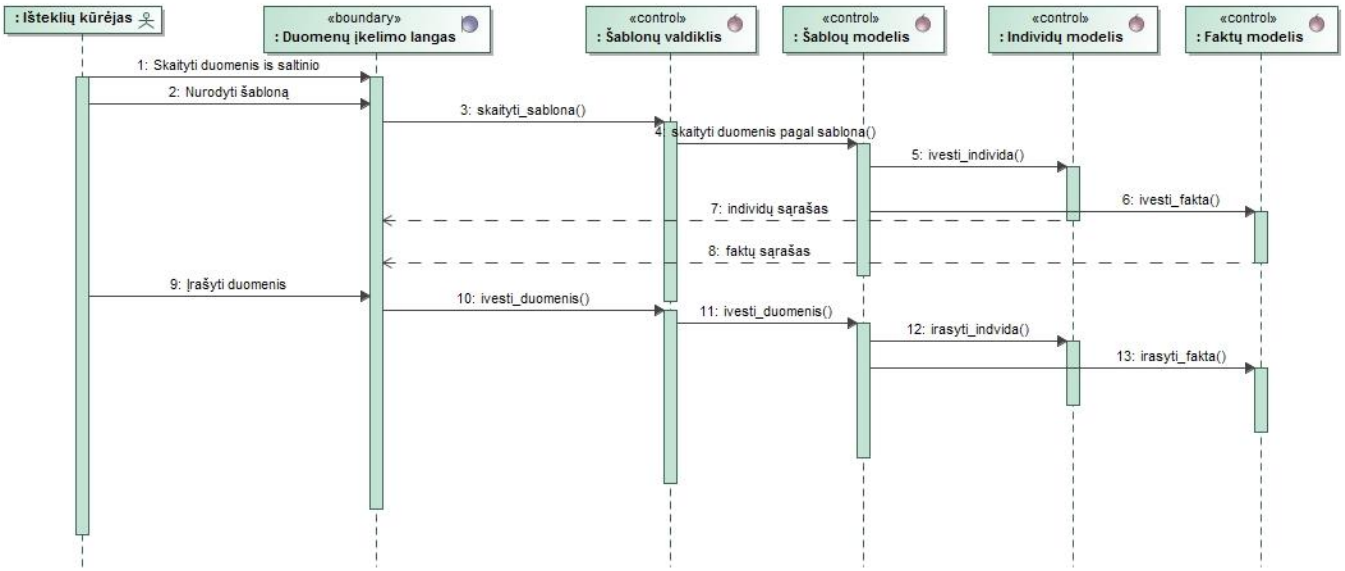
3.13 pav. Panaudojimo atvejo „Tvarkyti individų duomenis“ sekų diagrama

Tvarkant faktus sistema pateikia vartotojui esamų faktų sąrašą ir leidžia vartotojui trinti, redaguoti arba kurti naujus. Kuriant arba redaguojant vartotojas gali pasirikti fakto tesktą, jo tipus ir pageidautiną reikšmę (3.14 pav.).



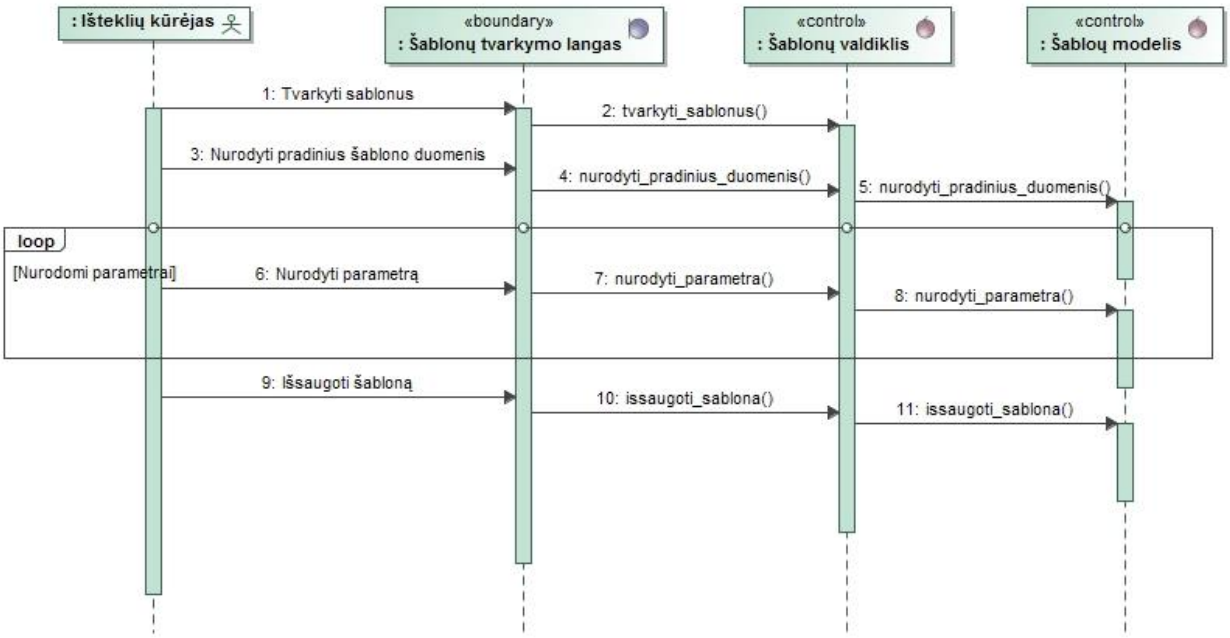
3.14 pav. Panaudojimo atvejo „Tvarkyti faktų duomenis“ sekų diagrama

Įkeliant duomenis vartotojas nurodo šabloną. Sistema iš šablono nuskaito šaltinio adresą, teksto pradžią ir pabaigą, egzempliorių pradžią ir pabaigą, ir visus elementus. Nuskaičius visus elementus, sistema nuskaito šaltinį ir kol nepasiekia teksto galo, pereina per visus elementus, jei elementas turi nurodytą elemento pradžią ir pabaigą, sistema nuskaito tekstą iš šaltinio, jei ne, sistema naudoja šablono duomenis (3.15 pav.).



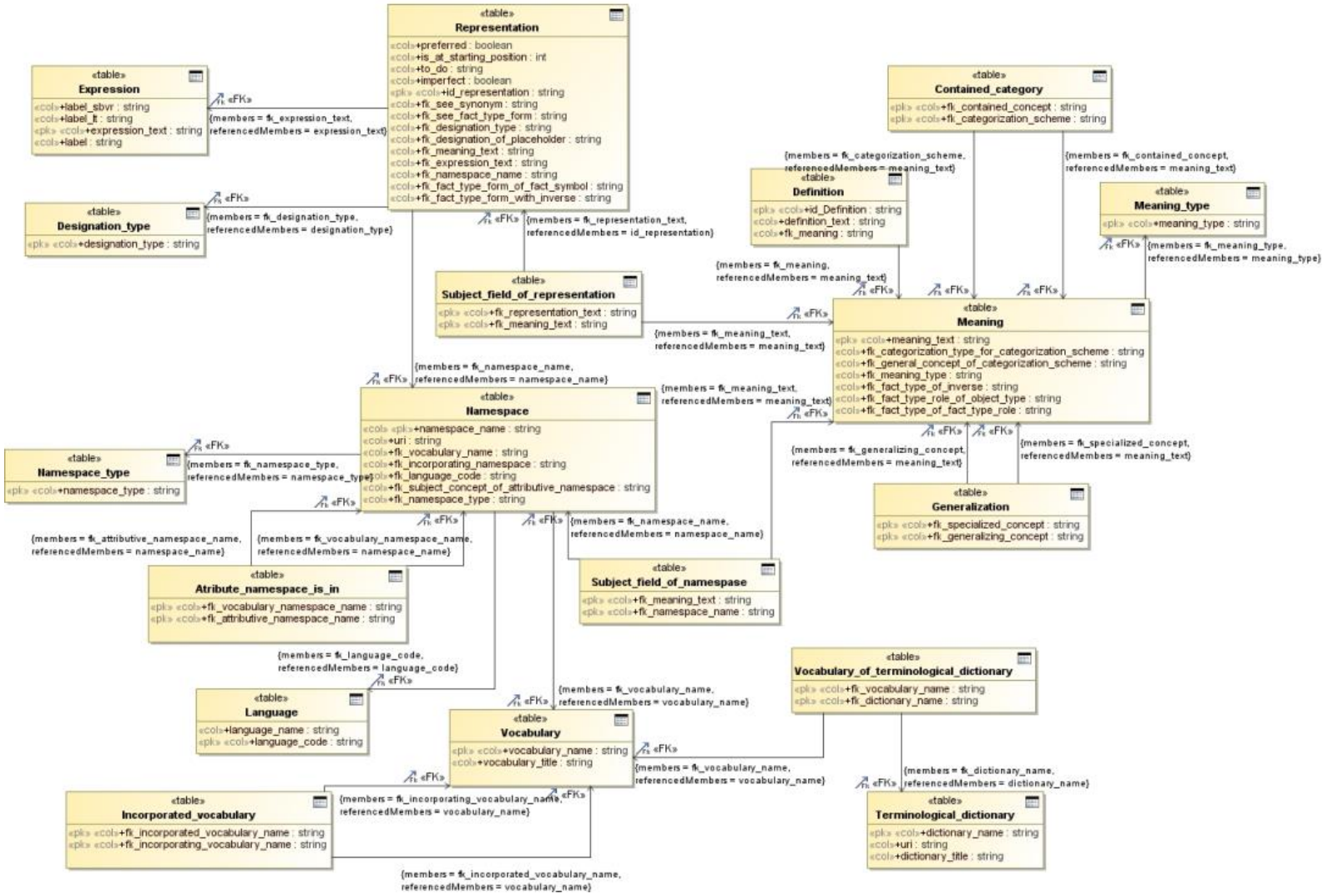
3.15 pav. Panaudojimo atvejo „Įkelti duomenis iš šaltinių“ sekų diagrama

Kuriant šablonus vartotojas nurodo pradinius duomenis (teksto pradžia ir pabaiga, egzemplioriaus pradžia ir pabaiga, šablono pavadinimą). Jei šablonas renkamas iš šaltinio, vartotojas nurodo parametrus (žodžio pradžią ir pabaigą, praleidimų skaičių ir reikšmę). Jei šaltinis statinis, nurodo reikšmę ir turinį (3.16 pav.).



3.16 pav. Panaudojimo atvejo „Tvarkyti šablonus“ sekų diagrama

3.3. Duomenų bazės schema

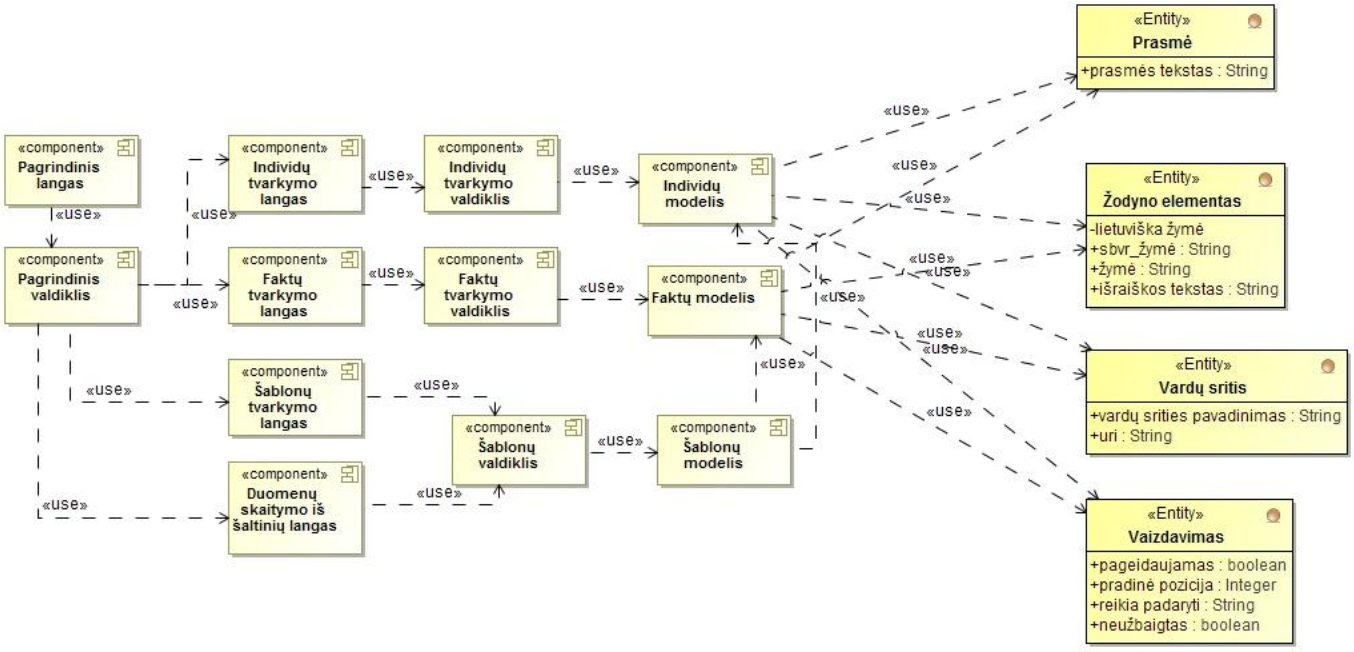


3.17 pav. Duomenų bazės schema

3.4. Realizacijos modelis

3.4.1. Programinių komponentų architektūra

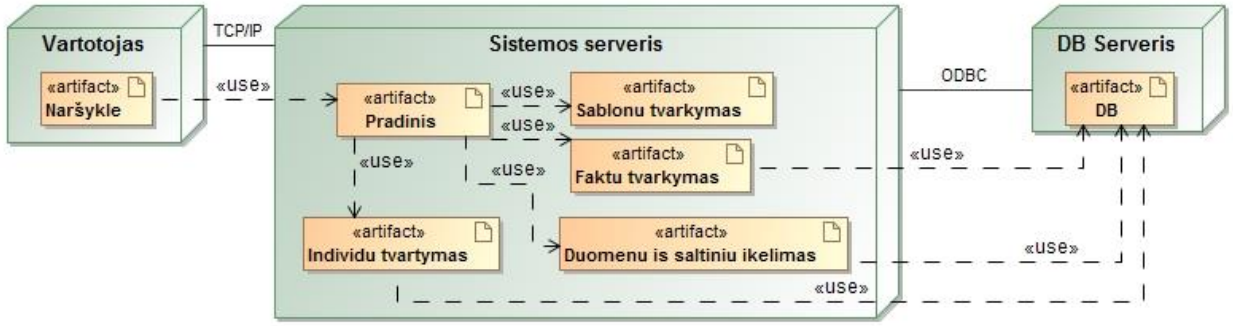
Programinių komponentų architektūra, atitinkanti loginę sistemos architektūrą, pavaizduota 3.17 pav.



3.17 pav. Sistemos realizavimas komponentais

3.4.2. Diegimo modelis

Sistemos diegimo diagrama vaizduoja realizuotų artefaktų išdėstymą techniniuose įrenginiuose (3.18 pav.).



3.18pav Sistemos diegimo modelis

4. SPRENDIMO REALIZACIJA IR TESTAVIMAS

4.1. Sprendimo realizacijos ir veikimo aprašas

Darbo metu buvo kuriama sistema kuri leidžia vartotojui pildyti ir redaguoti ontologijų individus ir faktus, bei rinkti automatizuotai iš duomenų bazių, interneto puslapių ir dokumentų su vartotojų sukurtų šablonų pagalba. Įvesti duomenys yra išsaugomi duomenų bazėje sukurtoje pagal SVBR metamodelį iš kurios sistema gali tuos duomenis eksportuoti i ontologijas.

Sistemos veikimas pailiustruotas nuosekliai atliekamais žingsniais. Pirmuose dviejuose žingsniuose aprašytas atvejis, kai informacija įkeliama į duomenų bazę iš interneto svetainių, tačiau ji gali būti įkeliama ir iš tekstinių dokumentų arba įvedama rankiniu būdu.

1 žingsnis. Šablono sukūrimas.

Pavyzdžiui, reikia perkelti duomenis iš interneto svetainės http://www.lrvalstybe.lt.

Kuriant šabloną puslapiui „LR Valstybė“ (5.1 pav.), į lauką „pavadinimas“ nurodomas norimas failo pavadinimas;

į lauką „adresas“ nurodomas adresas (http://www.lrvalstybe.lt/lsdp-taryba-1001491/).

į lauką „teksto pradžia“ įvedamas ( ),

į lauką „teksto pabaiga“ ( ),

į lauką „egzemplioriaus pradžia“ – ( ),

į lauką „egzemplioriaus pabaiga“ – ( ),

į lauką „žodžio pradžia“ – (">),

į lauką „žodžio pabaiga“ – ( ),

į lauką „praleidimų nr“ – (1),

lauke „reikšmė“ pasirenkamas (Vardas) ir paspaudžiamas mygtukas „išsaugoti parametrą“.

Paspaudus „įrašyti parametrą“, vartotojas lauke „reikšmė“ pasirenka (Bendrinis konceptas), lauke „turinys“ pasirenka (Person), ir paspaudžia „įrašyti parametrą“;

lauke „reikšmė“ pasirenka (Prasmės tipas),

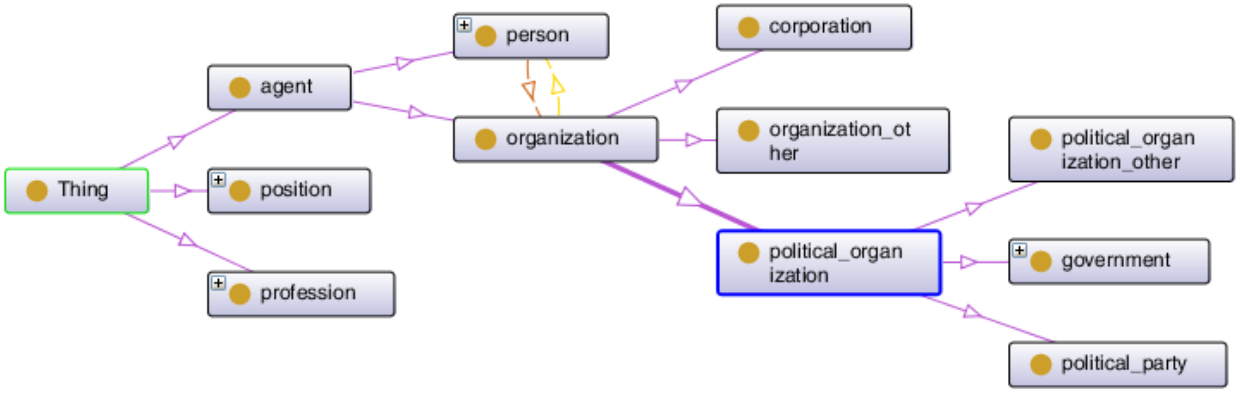
lauke „turinys“ – (individual_concept), paspaudžia „įrašyti parametrą“ ir „generuoti“.

4.1 pav. Puslapis „LR Valstybė“

2 žingsnis. Pagal sukurtą šabloną informacija iš nurodyto puslapio perkeliama į duomenų bazę.

3 žingsnis. Ontologijos užpildymas egzemplioriais iš duomenų bazės

Pavyzdžiui, norime įkelti politikos veikėjus į agentų ontologiją, kurios schema pateikiama 5.2 paveiksle (ontologijos aprašas pateikiamas 8.1 priede).



4.2 pav. Agentų ontologijos schema

Individo aprašas *OWL 2* funkcine sintaksine:

Declaration(NamedIndividual(**prefix**: **named_individual**))
ClassAssertion(**ontology_prefix**: **class prefix: named_individual**)
AnnotationAssertion(prefix:label_sbvr prefix:**named_individual**"**Individual_name in_SBVR_Style**"@lt)
AnnotationAssertion(prefix:label_lt prefix:**named_individual**"**Individual_name_in_Lithuanian**"@lt)
AnnotationAssertion(rdfs:label prefix:**named_individual**"**Individual_name_in_Lithuanian**"@lt)

Pavyzdžiui:
1)
Declaration(NamedIndividual(agents:Andrius_Kubilius))
ClassAssertion(agents:person agents:Andrius_Kubilius)
AnnotationAssertion(rdfs:label agents:Andrius_Kubilius "Andrius Kubilius"@lt)
AnnotationAssertion(agents:label_sbvr agents:Andrius_Kubilius "Andrius_Kubilius"@lt)
AnnotationAssertion(agents:label_lt agents:Andrius_Kubilius "Andrius Kubilius"@lt)
2)
Declaration(NamedIndividual(agents:Lietuvos_Respublikos_Seimas))
ClassAssertion(agents:government agents:Lietuvos_Respublikos_Seimas)
AnnotationAssertion(agents:label_sbvr agents:Lietuvos_Respublikos_Seimas "Lietuvos_Respublikos_Seimas"@lt)
AnnotationAssertion(agents:label_lt agents:Lietuvos_Respublikos_Seimas "Lietuvos Respublikos Seimas"@lt)
AnnotationAssertion(rdfs:label agents:Lietuvos_Respublikos_Seimas "Lietuvos Respublikos Seimas"@lt)

Sukurtas skriptas pagal šį šabloną gali užpildyti egzemplioriais bet kurią ontologiją, įstatydamas duomenis, išgautus iš sukurtos duomenų bazės. Skripto parametruose reikėtų nurodyti atitinkamos klasės vardą.

4 žingsnis. **Ontologijos užpildymas faktais**

Fakto aprašas *OWL 2* funkcine sintaksine:

ObjectPropertyAssertion(**prefix:object_property prefix:named_individual prefix: named_individual**)

Pavyzdžiui:

ObjectPropertyAssertion(agents:works_in__organization agents:Andrius_Kubilius agents:Lietuvos_Respublikos_Seimas)

Sukurtas skriptas pagal šį šabloną gali užpildyti faktais bet kurią ontologiją, įstatydamas duomenis, išgautus iš sukurtos duomenų bazės (atitinkami egzemplioriai turi būti jau įvesti). Skripto parametruose reikėtų nurodyti atitinkamos objektinės savybės pavadinimą, jos apibrėžimo (angl. *Domain*) ir kitimo srities (angl. *Range*) klasių vardus.

5. EKSPERIMENTINIS ONTOLOGIJŲ UŽPILDYMO PATIKIMAIS EGZEMPLIORIAIS SPRENDIMO TYRIMAS

Eksperimento tikslas: Išanalizuoti automatizuotą ontologijos egzempliorių įvedimą siekiant įvertinti automatizuoto įvedimo naudingumą lyginant su rankiniu įvedimu, ontologijų įvedimo spartos ir korektiškumo atžvilgiu iš vartotojo perspektyvos.

5.1. Eksperimento planas

Eksperimento metu vartotojas įves 100 ontologijos egzempliorių į sistemą naudodamas šablonus 3 puslapiams ir 100 ontologijų egzempliorių rankiniu būdu.

Vidutinis 1 egzemplioriaus įvedimo laikas įvedant rankiniu būdu bus skaičiuojamas egzempliorių įvedimo į ontologiją laiką padalinant iš egzempliorių skaičiaus. Įvedant automatizuotu būdu, šablono kūrimo laikas bus dalinamas iš egzempliorių skaičiaus.

5.2. Eksperimento rezultatai

Eksperimento rezultatai pateikiami 5.1 lentelėje.

5.1 lentelė. Eksperimento rezultatai

	Įvedimo į DB laikas	Įvedimo į DB klaidų skaičius	Įvedimo į ontologiją laikas sek	Įvedimo į ontologiją klaidų skaičius	Šablono kūrimo laikas	Egzempliorių skaičius	1 egzemplioriaus įvedimo trukmė	Klaidų procentas
Įvedimas rankiniu būdu								
	16s	0	1510s	4		100	15s	4%
Įvedimas automatizuotu būdu								
	2s	0	32s	0	263s	16	16s	0%
	24s	0	105s	0	315s	149	2s	0%
	7s	0	47s	0	276s	49	5s	0%

Išanalizavus eksperimento rezultatus, buvo prieita išvados, kad šablonai pagreitina didelio individų skaičiaus įvedimą. Tai ypač pasireikštų tais atvejais, kai vienas šablonas tinka dideliam skaičiui individų. Jei reikės daug skirtingų šablonų, pagreitėjimas bus mažesnis, be to, duomenų įvedimo ir šablonų kūrimo sparta priklausys nuo vartotojo sugebėjimų.

Galimos grėsmės eksperimento pagrįstumui yra pokyčiai interneto spartoje, vartotojo gebėjimai įvesti duomenis rankiniu būdu, išanalizuoti puslapius ir kurti šablonus. Šie faktoriai gali paveikti duomenų įvedimo laiką.

6. REZULTATŲ APIBENDRINIMAS IR IŠVADOS

1. Norint pagerinti semantinę paiešką, reikia sukaupti patikimų ontologijų egzempliorių, kurie leidžia gauti geresnius paieškos rezultatus.
2. Ontologijų užpildymo egzemplioriais metodų analizė parodė, kad šiuos metodus galima skirstyti į keturias grupes:
    1) tiesioginis įvedimas per ontologijų redaktorių;
    2) išgavimas iš reliacinių duomenų bazių;
    3) egzempliorių išgavimas iš nestruktūrizuoto teksto;
    4) egzempliorių įvedimas iš dalinai struktūrizuotų dokumentų, taikant šablonus.
3. Naudojant ontologijų redaktorių, sukuriami patikimi duomenys, tačiau toks įvedimas reikalauja daug pastangų.
4. Metodai, išgaunantys egzempliorius iš nestruktūrizuoto teksto, leidžia tą padaryti automatiškai, tačiau lietuvių kalbai pritaikytų metodų nėra, be to, taip sukurti egzemplioriai nėra patikimi.
5. Dėl šių priežasčių šiame darbe siekiama sukurti ontologijų užpildymo patikimais egzemplioriais metodiką ir įrankį, kuris leistų kaupti ontologijų egzempliorius pusiau automatizuotu būdu, įvedant juos iš dalinai struktūrizuotų dokumentų ir reliacinių duomenų bazių, bei perkelti į ontologijų dokumentus.
6. *SBVR* metamodelio analizė parodė, kad tokį įrankį tikslinga kurti sudarant duomenų bazės schemą *SBVR* metamodelio pagrindu. Šioje schemoje individai ir jų savybės bus saugomi pagal vieną schemą, iš kurios juos bus galima perkelti į ontologiją taikant tam sukurtą programinį komponentą.
7. Pagal sukurtus modelius buvo kuriama sistema, kuri pildo duomenų bazę, sukurtą remiantis SBVR metamodeliu, egzemplioriais, įvestais rankiniu būdu ir išrinktais iš įvairių šaltinių.
8. Darbo metu nebuvo realizuota vientisa sistema, apimanti pilną funkcionalumą, bet atskirų jos dalių realizacijos ir pasiekti rezultatai rodo, kad tokią sistemą galima realizuoti.
9. Išanalizavus eksperimento rezultatus, buvo prieita išvados, kad šablonai pagreitina didelio individų skaičiaus įvedimą. Tai pasireikštų tais atvejais, kai vienas šablonas tinka dideliam skaičiui individų. Jei reikės daug skirtingų šablonų, pagreitėjimas bus mažesnis, be to, šablonų kūrimo sparta priklausys nuo vartotojo sugebėjimų.

7. LITERATŪRA

[1] Motik, B., Horrocks I., Sattler, U: Bridging the gap between OWL and relational databases. Web Semantics: Science, Services and Agents on the World Wide Web. Vol. 7(2), April 2009, 74–89.
[2] Horridge, M: A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools Edition 1.3. March 24, 2011;
[3] OMG: Ontology Definition Metamodel. September 2013;
[4] W3C: OWL Web Ontology Language Overview. 10 February 2004, http://www.w3.org/TR/2004/REC-owl-features-20040210/;
[5] OMG: Semantics of Business Vocabulary and Business Rules (SBVR). September 2013;
[6] Gailly, F. „Transforming Enterprise Ontologies into SBVR formalizations“;
[7] Petasis, G., Karkaletsis, V., Paliouras G.: Ontology Population and Enrichment: State of the Art. Springer Berlin Heidelberg 2011, pp 134-166 15/03/2007;
[8] Tudorache, T., Nyulas, C., Noy, F., N., Musen A. M.,:WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. (2011), http://www.semantic-web-journal.net/sites/default/files/swj210_1.pdf;
[9] what is protégé?. http://protege.stanford.edu/overview/index.html;
[10] Nyulas, C., O'Connor, M., Tu, S.,: DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé.(2007)
[11] O'Connor, J., M., Halaschek-Wiener, C., Musen A., M.: Mapping Master: a Flexible Approach for Mapping Spreadsheets to OWL. Springer Berlin Heidelberg, The Semantic Web – ISWC 2010 Lecture Notes in Computer Science Volume 6497, 2010, pp 194-208
[12] Alatrish, S., E.,: Comparison of Ontology Editors. Vol. 4, 2012 http://joc.raf.edu.rs/4/_k_0020.pdf
[13] Ghawi, R., Cullot, N.,: Database-to-Ontology Mapping Generation for Semantic Interoperability. VLDB '07, September 23-28, 2007, Vienna, Austria;
[14] Amardeilh, F,.: OntoPop or how to annotate documents and populate ontologies from texts (2006);
[15] Giuliano, C., Gliozzo, A,.: Instance-Based Ontology Population Exploiting Named-Entity Substitution Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pages 265–272Manchester, August 2008;
[16] Telnarova, Z.: Relational database as a source of ontology creation, PROCEEDINGS OF THE IMCSIT. VOLUME 5, 2010;

8. PRIEDAI

8.1. Agentų ontologija funkcine sintakse:

```
Prefix(:=<http://www.w3.org/2002/07/owl#>)
Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
Prefix(rdf:=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Prefix(xml:=<http://www.w3.org/XML/1998/namespace>)
Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
Prefix(rdfs:=<http://www.w3.org/2000/01/rdf-schema#>)
Prefix(swrl:=<http://www.w3.org/2003/11/swrl#>)
Prefix(time:=<http://isd.ktu.lt/Time#>)
Prefix(semlt:=<http://isd.ktu.lt/SemLT#>)
Prefix(swrlb:=<http://www.w3.org/2003/11/swrlb#>)
Prefix(agents:=<http://isd.ktu.lt/Agents#>)


Ontology(<http://isd.ktu.lt/Agents>

Declaration(Class(agents:agent))
AnnotationAssertion(agents:label_sbvr agents:agent "agentas"@lt)
AnnotationAssertion(agents:label_lt agents:agent "agentas"@lt)
AnnotationAssertion(rdfs:label agents:agent "agentas"@lt)
Declaration(Class(agents:corporation))
AnnotationAssertion(agents:label_sbvr agents:corporation "korporacija")
AnnotationAssertion(agents:label_lt agents:corporation "korporacija")
AnnotationAssertion(rdfs:label agents:corporation "korporacija"@lt)
SubClassOf(agents:corporation agents:organization)
Declaration(Class(agents:government))
AnnotationAssertion(rdfs:label agents:government "vyriausybė"@lt)
AnnotationAssertion(agents:label_sbvr agents:government "vyriausybė"@lt)
AnnotationAssertion(agents:label_lt agents:government "vyriausybė"@lt)
SubClassOf(agents:government agents:political_organization)
Declaration(Class(agents:organization))
AnnotationAssertion(rdfs:label agents:organization "organizacija"@lt)
AnnotationAssertion(agents:label_sbvr agents:organization "organizacija")
AnnotationAssertion(agents:label_lt agents:organization "organizacija")
SubClassOf(agents:organization agents:agent)
Declaration(Class(agents:organization_other))
AnnotationAssertion(agents:label_sbvr agents:organization_other "kita_organizacija"@lt)
AnnotationAssertion(rdfs:label agents:organization_other "kita organizacija"@lt)
AnnotationAssertion(agents:label_lt agents:organization_other "kita organizacija"@lt)
SubClassOf(agents:organization_other agents:organization)
Declaration(Class(agents:person))
AnnotationAssertion(agents:label_sbvr agents:person "asmuo"@lt)
AnnotationAssertion(rdfs:label agents:person "asmuo"@lt)
AnnotationAssertion(agents:label_lt agents:person "asmuo"@lt)
SubClassOf(agents:person agents:agent)
Declaration(Class(agents:political_organization))
AnnotationAssertion(agents:label_sbvr agents:political_organization
"politinė_organizacija"@lt)
AnnotationAssertion(rdfs:label agents:political_organization "politinė organizacija"@lt)
```

AnnotationAssertion(agents:label_lt agents:political_organization "politinė organizacija"@lt)
SubClassOf(agents:political_organization agents:organization)
Declaration(Class(agents:political_organization_other))
AnnotationAssertion(rdfs:label agents:political_organization_other "kita politinė organizacija"@lt)
AnnotationAssertion(agents:label_sbvr agents:political_organization_other "kita_politinė_organizacija"@lt)
SubClassOf(agents:political_organization_other agents:political_organization)
Declaration(Class(agents:political_party))
AnnotationAssertion(rdfs:label agents:political_party "politinė partija"@lt)
AnnotationAssertion(agents:label_lt agents:political_party "politinė partija"@lt)
AnnotationAssertion(agents:label_sbvr agents:political_party "politinė_partija"@lt)
SubClassOf(agents:political_party agents:political_organization)
Declaration(Class(agents:position))
AnnotationAssertion(agents:label_lt agents:position "pareigos"@lt)
AnnotationAssertion(rdfs:label agents:position "pareigos"@lt)
AnnotationAssertion(agents:label_sbvr agents:position "pareigos"@lt)
Declaration(Class(agents:profession))
AnnotationAssertion(agents:label_lt agents:profession "profesija"@lt)
AnnotationAssertion(agents:label_sbvr agents:profession "profesija"@lt)
AnnotationAssertion(rdfs:label agents:profession "profesija"@lt)
Declaration(ObjectProperty(agents:has__position))
AnnotationAssertion(agents:label_lt agents:has__position "asmens pareigos"@lt)
AnnotationAssertion(agents:label_sbvr agents:has__position "asmens pareigos"@lt)
AnnotationAssertion(rdfs:label agents:has__position "asmens pareigos"@lt)
InverseObjectProperties(agents:has__position agents:is_position_of__person)
ObjectPropertyDomain(agents:has__position agents:person)
ObjectPropertyRange(agents:has__position agents:position)
Declaration(ObjectProperty(agents:has__profession))
AnnotationAssertion(agents:label_lt agents:has__profession "asmens profesija"@lt)
AnnotationAssertion(agents:label_sbvr agents:has__profession "asmens profesija"@lt)
AnnotationAssertion(rdfs:label agents:has__profession "asmens profesija"@lt)
InverseObjectProperties(agents:is_profession_of__person agents:has__profession)
ObjectPropertyDomain(agents:has__profession agents:person)
ObjectPropertyRange(agents:has__profession agents:profession)
Declaration(ObjectProperty(agents:is_position_of__person))
AnnotationAssertion(agents:label_sbvr agents:is_position_of__person "pareigos einamos asmens"@lt)
AnnotationAssertion(rdfs:label agents:is_position_of__person "pareigos einamos asmens"@lt)
AnnotationAssertion(agents:label_lt agents:is_position_of__person "pareigos einamos asmens"@lt)
InverseObjectProperties(agents:has__position agents:is_position_of__person)
ObjectPropertyDomain(agents:is_position_of__person agents:position)
ObjectPropertyRange(agents:is_position_of__person agents:person)
Declaration(ObjectProperty(agents:is_profession_of__person))
AnnotationAssertion(rdfs:label agents:is_profession_of__person "profesiją turi asmuo"@lt)
AnnotationAssertion(agents:label_sbvr agents:is_profession_of__person "profesiją turi asmuo"@lt)
AnnotationAssertion(agents:label_lt agents:is_profession_of__person "profesiją turi asmuo"@lt)

InverseObjectProperties(agents:is_profession_of__person agents:has__profession)
ObjectPropertyDomain(agents:is_profession_of__person agents:profession)
ObjectPropertyRange(agents:is_profession_of__person agents:person)
Declaration(ObjectProperty(agents:provides_work_for__person))
AnnotationAssertion(rdfs:label agents:provides_work_for__person "organizacijoje dirba asmuo"@lt)
AnnotationAssertion(agents:label_sbvr agents:provides_work_for__person "organizacijoje dirba asmuo"@lt)
AnnotationAssertion(agents:label_lt agents:provides_work_for__person "organizacijoje dirba asmuo"@lt)
InverseObjectProperties(agents:provides_work_for__person agents:works_in__organization)
ObjectPropertyDomain(agents:provides_work_for__person agents:organization)
ObjectPropertyRange(agents:provides_work_for__person agents:person)
Declaration(ObjectProperty(agents:works_in__organization))
AnnotationAssertion(agents:label_lt agents:works_in__organization "asmuo dirba organizacijoje"@lt)
AnnotationAssertion(rdfs:label agents:works_in__organization "asmuo dirba organizacijoje"@lt)
AnnotationAssertion(agents:label_sbvr agents:works_in__organization "asmuo dirba organizacijoje"@lt)
InverseObjectProperties(agents:provides_work_for__person agents:works_in__organization)
ObjectPropertyDomain(agents:works_in__organization agents:person)
ObjectPropertyRange(agents:works_in__organization agents:organization)
Declaration(NamedIndividual(agents:Andrius_Kubilius))
AnnotationAssertion(rdfs:label agents:Andrius_Kubilius "Andrius Kubilius"@lt)
AnnotationAssertion(agents:label_sbvr agents:Andrius_Kubilius "Andrius_Kubilius"@lt)
AnnotationAssertion(agents:label_lt agents:Andrius_Kubilius "Andrius Kubilius"@lt)
ClassAssertion(agents:person agents:Andrius_Kubilius)
ObjectPropertyAssertion(agents:works_in__organization agents:Andrius_Kubilius agents:Lietuvos_Respublikos_Seimas)
Declaration(NamedIndividual(agents:Lietuvos_Respublikos_Seimas))
AnnotationAssertion(agents:label_sbvr agents:Lietuvos_Respublikos_Seimas "Lietuvos_Respublikos_Seimas"@lt)
AnnotationAssertion(agents:label_lt agents:Lietuvos_Respublikos_Seimas "Lietuvos Respublikos Seimas"@lt)
AnnotationAssertion(rdfs:label agents:Lietuvos_Respublikos_Seimas "Lietuvos Respublikos Seimas"@lt)
ClassAssertion(agents:government agents:Lietuvos_Respublikos_Seimas)
)