KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Ernestas Vyšniauskas

OWL ONTOLOGIJŲ TRANSFORMAVIMAS Į RELIACINIŲ DUOMENŲ BAZIŲ SCHEMAS

Magistro darbas

Vadovė
prof. dr. L. Nemuraitė

KAUNAS, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas
prof. dr. R. Butleris

OWL ONTOLOGIJŲ TRANSFORMAVIMAS Į RELIACINIŲ DUOMENŲ BAZIŲ SCHEMAS

Informatikos inžinerijos magistro baigiamasis darbas

Recenzentas
doc. dr S. Maciulevičius

Vadovė
prof. dr. L. Nemuraitė

Atliko
IFM 1/4 gr. stud.
E. Vyšniauskas

KAUNAS, 2007

SUMMARY

Transforming OWL Ontologies To Relational Database Schemas

The current work has arisen with respect to the growing importance of ontology modelling in Information Systems development. Due to emerging technologies of Semantic Web, it is desirable to use for this purpose the Web Ontology Language OWL. From the other side, the relational database technology has ensured the best facilities for storing, updating and manipulating the information of problem domain.

This work covers analysis of the process how ontology of a particular domain described in OWL may be transformed and stored in a relational database. The algorithms for transformation of domain ontology, described in OWL, to relational database are proposed. According this algorithm, ontology classes are mapped to relational tables, properties to relations and attributes, and constraints – to metadata. The proposed algorithm is capable to transform all OWL Lite and part of OWL DL syntax. The further expansion of the algorithm to cover more capabilities of OWL should be based on the same principles. The prototypical tool, performing transformations, has been implemented as add-in for the ontology development tool “Protégé”. The methodology of transformation is illustrated with an example.

Key words: information system, ontology, transformation, relational database, knowledge base, OWL.

TURINYS

Lentelių sąrašas

Paveikslėlių sąrašas

1. Įvadas

Didėjant programinės įrangos galimybėms ir sudėtingumui, informacinių sistemų koncepciniai modeliai įgauna vis tobulesnį pavidalą ir virsta žinių modeliais – ontologijomis. Ontologijos sąvoka yra kilusi iš filosofijos. Ontologija nagrinėja esminių tam tikros srities sąvokų egzistencijos, jų savybių ir sąryšių su kitomis esminėmis sąvokomis klausimus. Ontologijos sąvoka tapo populiari ryšium su Semantinio žiniatinklio idėjomis, tačiau šiandien ontologijos tampa reikalingos ir įprastose įmonių informacinėse sistemose.

Paimkime pavyzdžiu įmonės gaminamų produktų koncepcinį modelį. Spartėjant technologijų pažangai, augant rinkos pasiūlai ir jos kokybei, keičiasi ir paklausa. Šiuolaikiniame pasaulyje vartotojai pageidauja produktų, kuo labiau išsiskiriančių iš bendros produktų masės, labiau atitinkančių jų norus, skonį ar interesus. Klientai reikalauja tik jiems sukurtų produktų ar atliktų sprendimų, kažkuo ypatingų ir besiskiriančių nuo kitų. Vienas iš būdų, padedančių kurti ir pateikti asmeninius produktus yra produktų konfigūracijos sistemos [1]. Produktų konfigūracijos sistemos arba konfigūratoriai yra informaciniai įrankiai, kurie įgalina produktų užsakymų priėmimo proceso automatizavimą, surinkdami kliento poreikius ir reikalavimus nedalyvaujant žmogiškiems tarpininkams, taip pat leidžia lengvai konstruoti naujus gaminius bei atlikti produktų semantinę paiešką.

Visiems šiems veiksmams atlikti konfigūravimo sistemoms reikalinga informacija ir žinios. Tai žinios apie pačius produktus, jų sudedamąsias dalis, įvairios konstravimo taisyklės bei apribojimai. Šios žinios saugomos konfigūravimo sistemų žinių bazėje.

Tačiau sukurti šią informacijos ir žinių bazę nėra paprasta. Produktų aprašymo kataloguose yra daugybė žinių ir taisyklių apie pačius produktus ir jų tarpusavio ryšius, tačiau visa tai būna paslėpta duomenyse, kuriuos sunku ištraukti ir panaudoti programiniuose įrankiuose [1], [22]. Dar sudėtingiau yra surinktas žinias klasifikuoti ir suvesti į informacinėms sistemoms tinkamą formą.

Šiame darbe siūloma informacinių sistemų žinių bazės kūrimo procese pritaikyti ontologijas. Ontologijos suteikia pastovią produktų terminologiją viso jų gyvavimo ciklo metu, taip pat galimybę generuoti įvairius tų pačių produktų vaizdus priklausomai nuo esančio konteksto [14], [24]. Ontologija galima aprašyti tam tikrą objektą, jo savybes, ryšius su kitais objektais, taip pat įvairius apribojimus [11], [12]. Tam labiausiai pritaikyta yra „Web Ontology Language“ arba OWL kalba, kuri skirta detaliai aprašyti tam tikros dalykinės srities ontologijai [3].

Taigi, ontologijų pritaikymas konfigūravimo sistemų žinių bazės kūrimo procese yra perspektyvus būdas patobulinti ir supaprastinti produktų katalogų kūrimą ir jų palaikymą [7]. Tačiau tam tikra kalba sudarytą ontologijos aprašą yra sudėtinga pritaikyti taikomosioms programoms [9]. Informacija saugoma tekstinio tipo failuose, kurių naudojimas, esant dideliems duomenų kiekiams, tampa neefektyvus.

Tam, kad išvengti pernelyg sudėtingo ir neefektyvaus ontologijų informacijos naudojimo taikomosiose programose, ontologiniams aprašams saugoti tinkamiausia naudoti reliacines duomenų bazes, kurios skirtos dirbti su dideliais duomenų kiekiais. Šiame darbe pasiūlytas algoritmas, kuris atlieka dalykinės srities ontologijos aprašo transformaciją į reliacinę duomenų bazę. Algoritmo veikimas eksperimentinio tyrimo metu iliustruojamas pavyzdžiu. Nors pateikiami pavyzdžiai yra skirti produktų konfigūravimo žinių bazei, algoritmai tinka bet kokiai dalykinei sričiai, kurios žinių modeliui aprašyti pakanka magistriniame darbe apimtų konceptų, ryšių, savybių ir apribojimų aibės.

Darbo struktūra:

Skyriuje „Ontologijų transformacijų tyrimo sritis ir analizė“ suformuluoti darbo tikslai, apibrėžta tyrimo sritis, objektas ir problema. Pateikiamas ontologijos apibrėžimas ir pagrindinės sąvokos. Tolesniuose skyriuose aprašomas ontologijų taikymas produktų konfigūravimui, atlikta ontologijos aprašymo kalbų analizė, bei literatūros šaltiniuose pateiktų sprendimų problemai spręsti lyginamoji analizė. Nustatytas dalykinės srities žinių bazės kūrimo procesas ir galimas architektūros modelis. Atlikta transformacijos sistemos vartotojų analizė, iškeltas projekto tikslas, nustatyti kokybės kriterijai, pasirinkti projektavimo metodai ir priemonės. Skyriaus pabaigoje suformuluotos analizės išvados.

Skyriuje „Transformacijos sistemos reikalavimų specifikacija“ nustatyti transformacijos įrankio nefunkciniai reikalavimai, išskirti sistemos panaudojimo atvejai ir sudarytos detalios jų specifikacijos.

Skyriuje „Ontologijos aprašo OWL transformavimo į reliacinę duomenų bazę algoritmai“ pateikiami sukurti ontologijos klasių, objektinių ir duomenų tipų savybių transformavimo į RDB algoritmai. Taip pat detalizuota apribojimų transformacija, aprašyti kardinalumo, „allValuesFrom“, „someValuesFrom“ ir „hasValue“ apribojimų atvaizdavimo į RDB algoritmai. Skyriaus pabaigoje pateikiamas ontologijos duomenų egzempliorių įrašymo į RDB algoritmas.

Sekančiuose skyriuose pateikiamas sukurtas ontologijos aprašo transformavimo įrankio programinės realizacijos projektas bei aprašyta transformacijos įrankio prototipo realizacija.

Skyriuje „Eksperimentinis tyrimas ir darbo rezultatų įvertinimas“ sudaryta pavyzdinė dalykinės srities ontologija ir sukonstruotas jos OWL aprašas, kuris transformuojamas į reliacinę duomenų bazę. Skyriaus pabaigoje pateikiama atlikta darbo rezultatų kritinė analizė.

Dalyvauta konferencijose, kuriose skaityti su šio darbo tematika susiję straipsniai (straipsnių medžiaga pateikta priede):

- Informacinės technologijos 2006 (11-toji tarpuniversitetinė doktorantų ir magistrantų konferencija), VU KHF, skaitytas straipsnis „Produkto konfigūravimo žinių bazės ontologinis modeliavimas“;
- Business Informatics Research BIR 2006 (5-toji tarptautinė konferencija), KTU, spausdintas žurnale „Information Technology and Control“ ISSN 1392-124X straipsnis „Transforming Ontology Representation from OWL to Relational Database“.

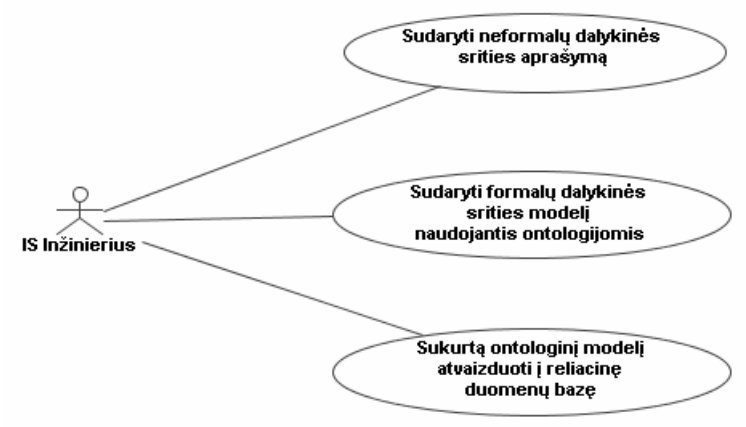
2. Ontologijų transformacijų tyrimo sritis ir analizė

2.1 Darbo tikslas

Darbo tikslas yra sukurti arba panaudoti egzistuojančias priemones dalykinės srities aprašymui tokia forma, kuri supaprastintų ir formalizuotų dalykinės srities žinių inžineriją ir leistų efektyviai panaudoti šias žinias taikomosiose programose. Metamodelis (metaontologija) turi leisti sukurti dalykinės srities žinių modelį, kuris leistų paimti bet kurį jos elementą ir atsakyti į tokius klausimus:

- Kokie yra reikalavimai šio elemento panaudojimui (įjungimui į produktą, į paslaugų paketą, arba bendruoju atveju – į sistemą);
- Kokie yra su šiuo elementu nesuderinami elementai (arba kokiomis savybėmis pasižymintys elementai);
- Kokie yra rekomenduotini kiti elementai (ar kokias savybes turintys kiti elementai);
- Kokie kiti elementai gali tarnauti kaip alternatyva šiam elementui (kokie santykiniai privalumai ir trūkumai);
- Kiti panašūs klausimai.

Taip pat turi būti sukurtas transformacijos algorimtas, leidžiantis išsaugoti dalykinės srities ontologijos metamodelį reliacinėje duomenų bazėje. Siekiami darbo tikslai turėtų leisti informacinių sistemų inžinieriui įgyvendinti 1 pav. pateiktus poreikius, atsirandančius žinių bazės kūrimo proceso metu.



1 pav. IS inžinieriaus darbo tikslai

Tam, kad patenkinti poreikius, atsirandančius žinių bazės kūrimo proceso metu, iškelti šie darbo uždaviniai:

- Sukurti arba praplėsti redaktorių, kuris leistų kurti ontologinį aprašą tenkinančią konfigūraciją;
- Sukurti transformacijos algoritmą, kuris leistų atvaizduoti ontologijos aprašą OWL (angl. *Web Ontology Language*) į DDL (angl. *Data Definition Language*) skriptą;
- Išbandyti metodiką sukuriant produktų konfigūravimo ontologinį aprašą ir transformuojant jį į reliacinę duomenų bazę.

2.2 Tyrimo sritis, objektas ir problema

Šiame darbe nagrinėjama tyrimo sritis yra ontologijų inžinerija, taip pat ontologijų atvaizdavimas į informacinių sistemų inžinerijos modelius. Tyrimo objektas yra dalykinės srities ontologijos kūrimo ir atvaizdavimo į reliacinę duomenų bazę metodika. Problema yra susieti ontologijas su vartotojų taikomosiomis programomis, užtikrinti jose naudojamų sąvokų atitikimą ontologijos apibrėžtiems konceptams.

Sprendžiamos problemos:

- Dalykinės srities semantinio modelio aprašymas ontologijų aprašymo kalba (OWL).
- Dalykinės srities semantinio modelio transformavimas į reliacinėje DBVS saugomos žinių bazės meta-aprašymą.

Eksperimentinis taikymas atliekamas sukuriant produktų konfigūracijos ontologiją ir atvaizduojant ją į reliacinę duomenų bazę.

2.3 Ontologijos apibrėžimas ir pagrindinės savokos

Ontologija – tai esminių dalykinės srities esybių ir jų semantinių ryšių aprašymas. Ontologija turi daug apibrėžimų. Ontologija (didžioji "O") yra filosofinė disciplina; ontologija (mažoji "o") yra specifinis artefaktas, sukurtas norint apibrėžti numatytą žodyno prasmę [16]. Ontologija turi apimti bendro naudojimo žodyną ir jo prasmės aprašymą - be prasmės išaiškinimo žodynas gali būti nevienareikšmiškas. Trumpiausias apibrėžimas: ontologija yra aiškiai išreikšta bendrai naudojamos konceptualizacijos specifikacija [23].

Ontologija apibrėžia tam tikroje žinių srityje naudojamas išraiškas ir konceptus (prasmę) bei jų tarpusavio ryšius [19]. Ontologijos aprašas (žodynas) gali turėti įvairius pavidalus:

– Taksonomija, turinti minimalią žinių hierarchiją – tėvo/vaiko ryšius;

– Tezauras (išplėstas žodynas), apimantis žodžius ir sinonimus;

– Konceptualus modelis, turintis sudėtingesnius ryšius;

– Loginė teorija, apimanti labai sudėtingas, išraiškingas, suderintas ir prasmingas žinias.

Taisyklinga ontologija turi būti išreikšta aiškiai apibrėžta sintakse, turinčia aiškiai apibrėžtą mašininę interpretaciją, atitinkančią ankstesnį ontologijos apibrėžimą.

Detaliau panagrinėsime eksperimentinio pavyzdžio – konfigūracijos ontologiją**. Konfigūracija** yra abstrakčios sistemos elementų ir jų komponavimo taisyklių aprašas. **Produkto konfigūracija –** tai produkto dalių ir jų komponavimo taisyklių aprašymas. **Konfigūracijos valdymo sistema** yra informacinė sistema, leidžianti valdyti konfigūraciją. Darbe nagrinėjamų sistemų veikimas yra paremtas metaduomenų bazėje saugomais konfigūracijos modeliais. Sukurta konfigūracijos sistema leistų įvesti duomenis apie produktus ir vykdyti paiešką konkrečioje produktų klasėje pagal bet kokią tai klasei būdingą savybę.

Konfigūracijos ontologija - tai konfigūracijos aprašyme dalyvaujančių esybių ir jų semantinių ryšių aprašymas. Formalizuotas ontologijos aprašymas šiame darbe vadinamas modeliu.

Ontologijos esybės gali būti tokios:

- Produktas
- Produktas :: produkto_atributas
- Produktas :: produkto_elementas
- Produktas :: būtinas_elementas :: produkto_elementas
- Produktas :: produkto_elementas :: nesuderinami_elementai
- Produktas :: panašūs_produktai
- Produktas :: nesuderinami_produktai

Konfigūracijos modelis - tai griežčiau aprašyta (formalizuota) konfigūracijos ontologija. Konfigūracijos modelis yra metainformacija apie produktų klasę. Šiame darbe konfigūracijos modeliai yra dviejų lygių:

- **Formalus**: kai modelis aprašytas OWL (angl. *Web Ontology Language*) sintakse ir gali būti transformuojamas į reliacinį modelį;
- **Reliacinis**: kai modelis aprašytas DDL (angl. *Data Definition Language*) sintakse, kad būtų patalpintas į metaduomenų bazę.

Metaduomenų bazė konfigūracijos sistemos atveju – tai duomenų bazė, sauganti konkrečių konfigūracijos modelių reliacinius aprašus.

Konfigūracijos metaontologija nusako konfigūracijos ontologijai aprašyti naudojamas esybes ir jų semantinius ryšius. Pagal metaontologiją kuriamas metamodelis, kuriuo galima remtis kuriant metaduomenų bazės schemą.

Metaontologijos esybės gali būti tokios:

klausimas :: atsakymo_galimi_variantai

klausimas :: keli_atsakymo_variantai_galimi

atsakymovariantas :: sekantis_klausimas

atsakymovariantas :: nesuderinami_kiti_to_paties_arba_kito_klausimo_atsakymo_variantai

ir t.t.

Konfigūracijos metamodelis - tai griežčiau aprašyta (formalizuota) konfigūracijos metaontologija. Konfigūracijos metamodeliu galima (bet neprivaloma) remtis kuriant metaduomenų bazės schemą. **Metaduomenų bazės schema** nusako, kaip konfigūracijos modelius aprašyti reliaciniu pavidalu.

2.4 Ontologijų taikymas produktų konfigūravimui

Produkto konfigūravimo sistemos arba konfigūratoriai yra laikomi vienomis iš sėkmingiausių programų, kuriose taikomos dirbtinio intelekto technologijos [13]. Konfigūratoriai yra informaciniai įrankiai, kurie įgalina užsakymų priėmimo proceso automatizavimą, surinkdami kliento poreikius ir reikalavimus nedalyvaujant žmogiškiems tarpininkams.

Daugumoje atvejų, konfigūratorius įdiegiamas vartotojo sąsajoje tarp tiekėjo ir jo internetinio kliento. Jo pagrindinė užduotis yra padėti klientams patiems susikonfigūruoti savo produktus pagal savo pačių individualius poreikius [3]. Pavyzdžiui, klientams gali būti suteikiama galimybė keisti bazinį produktą, tuo pat metu šiuos pokyčius pavaizduojant grafiškai.

Konfigūratoriai palaiko patį konfigūracijos procesą. Šis procesas turi būti atliekamas tokio asmens, kuris tiksliai suprastų klientų poreikius ir tokiu būdu sugebėtų sukurti pilną produkto aprašą, kuris atitiktų kiekvieno kliento individualius reikalavimus. Konfigūravimo užduotis yra turint aibę klientų reikalavimų ir produktų šeimos aprašą, iš visų alternatyvų surasti veiksmingą, efektyvų ir pilnai specifikuotą atskirą produkto atvejį, kuris aprašytų bendrą visų šeimos produktų struktūrą.

Techninėje literatūroje sutinkama įvairių produkto konfigūratoriaus apibrėžimų. Bendru atveju dirbtinio intelekto bendruomenė konfigūratoriumi laiko programinę įrangą. Pavyzdžiui, produkto konfigūratorius apibrėžiamas kaip „...programa, turinti logines galimybes kurti, palaikyti ir naudoti produktų modelius, kas suteikia pilną produktų pasirinkimų ir produktų variantų kombinacijų aprašą, naudojant minimalius duomenų įvedimus ir minimalų duomenų

saugojimą“ [14]. Pagrindinis techninis konfigūratoriaus komponentas yra žinių bazė, kuri susideda iš dviejų dalių, t.y. duomenų bazės ir konfigūracijos logikos.

Duomenų bazė talpina visą komponentų tipų ir egzempliorių aibę, tuo tarpu konfigūracijos logika apibrėžia tarp skirtingų komponentų egzistuojančius apribojimus, kas leidžia sudaryti tik taisyklingus ir pilnai struktūrizuotus produkto variantus.

Kataloge esančiame produkto aprašyme yra daugybė žinių ir taisyklių apie pačius produktus ir jų tarpusavio ryšius, tačiau visa tai būna paslėpta duomenyse, kuriuos sunku ištraukti ir panaudoti programiniuose įrankiuose [13]. Dėl tokios produkto duomenų struktūros, kur savybės ir ryšiai nėra tiksliai išreikšti kokiu nors standartizuotu terminologiniu būdu, yra labai sudėtinga ir brangu manipuliuoti šiais duomenimis, naujai interpretuoti juos kitokiam panaudojimui arba kurti naują katalogo struktūrą, taip pat dalintis jais tarp skirtingų padalinių (kiekvienas jų gali turėti skritingą duomenų reikšmės ir panaudojimo tikslų supratimą).

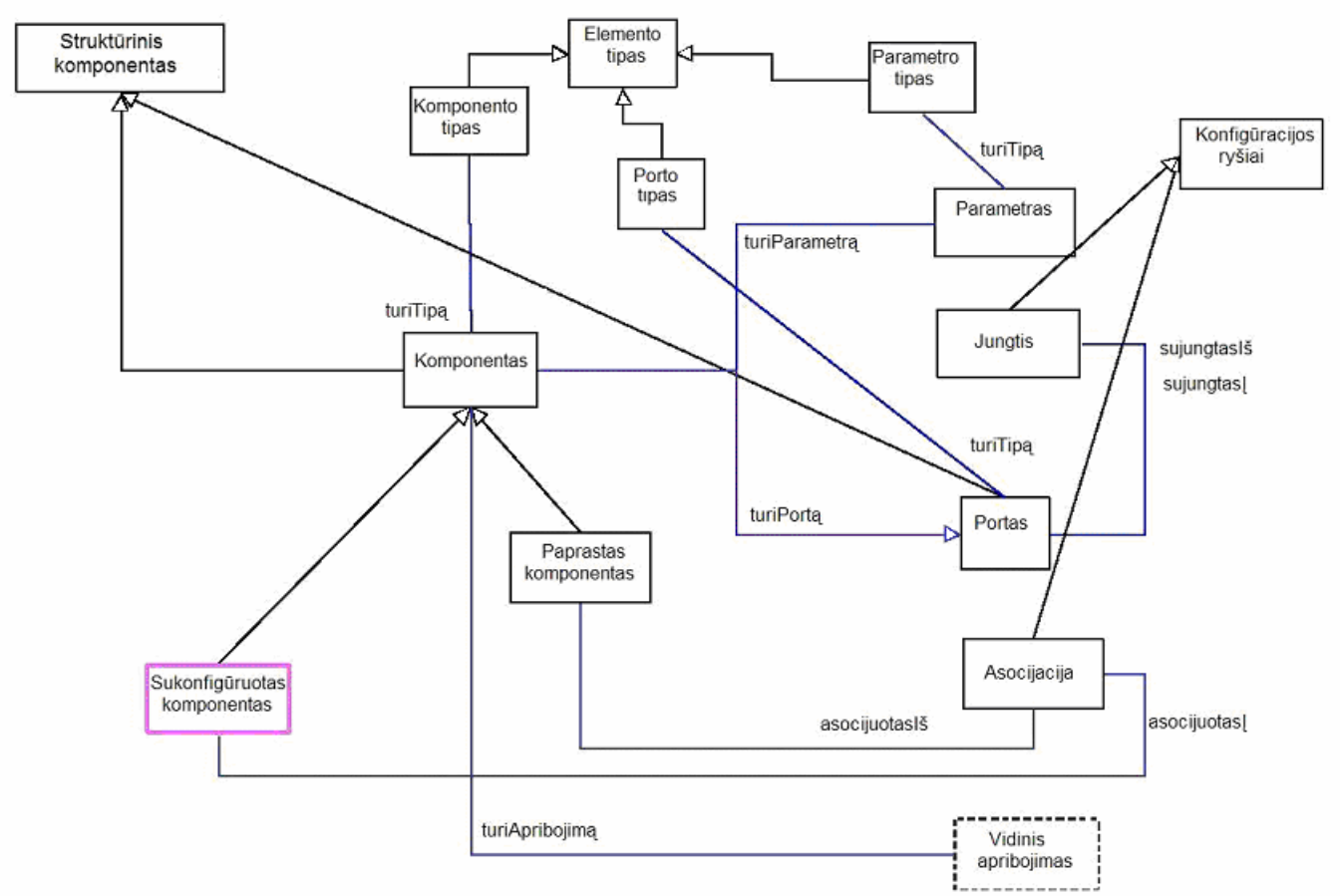
Nepaisant to, ontologijos gali būti perspektyvus ir įgyvendinamas būdas patobulinti ir supaprastinti produktų katalogų kūrimą ir jų palaikymą viso produkto gyvavimo ciklo metu. Ontologijos suteikia pastovią produktų terminologiją viso jų gyvavimo ciklo metu, taip pat suteikia galimybę generuoti įvairius tų pačių produktų vaizdus priklausomai nuo esančio konteksto.

Norint sukurti konfigūracijos ontologiją, pirmiausia reikia išsiaiškinti, iš ko susideda konfigūracija. Pagal Mittal ir Frayman [17] apibrėžimą, konfigūracijos ontologija yra:

- Rinkinys komponentų (produktų), kurie gali būti aprašyti rinkiniais savybių ir jungčių, jungiančių juos su kitais komponentais.
- Apribojimai kiekvienai jungčiai, apibrėžiantys kokie komponentai gali būti prijungiami prie jungies. Taip pat ir kiti struktūriniai apribojimai.
- Vartotojo reikalavimai su pageidaujamos konfigūracijos aprašymu. Taip pat gali būti tam tikri sprendimo optimizavimo kriterijai.

Taigi, pagrindiniai konceptai, kurie gali būti naudojami konfigūracijos ontologijoje yra: komponentai, jungtys, ryšiai, savybės arba atributai, apribojimai, vartotojo reikalavimai.

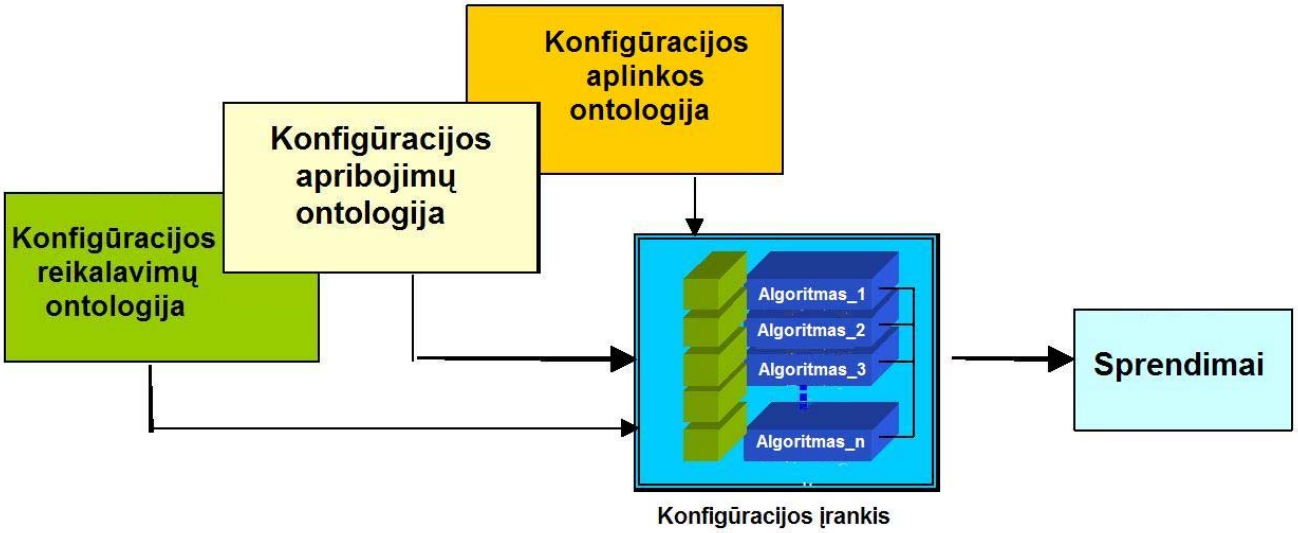
Žemiau esančiame paveiksle (2 pav.) pateikiama konfigūracijos ontologija, sukurta naudojant šiuos konceptus [17]. Sukonfigūruotas komponentas apibrėžtas raudonai, tuo tarpu abribojimai vaizduojami taškiniame stačiakampyje. Kadangi ontologijoje kardinalumo apribojimai gali būti nenurodyti, jie čia nevaizduojami.



2 pav. Konfigūracijos ontologija

Konfigūracijos ontologija yra geriau suprantama kai padalinama į tris ontologijas [14], apibrėžiančias skirtingus konfigūracijos ontologijos aspektus (3 pav.):

- **Konfigūracijos aplinkos ontologija** – tai pagrindinė konfigūracijos problemos ontologija. Ši ontologija apima visą statinę informaciją, pateikia pagrindinius taksonomijos apibrėžimus, tokius kaip komponentai, atributai, ryšiai ir t.t.
- **Konfigūracijos reikalavimų ontologija** – tai ontologija, kuri apima informaciją kaip išspręsti tam tikras konfigūracijos užduotis, t.y. suteikia taksonomijas, apibrėžiančias vartotojo reikalavimų specifikacijas. Tai sistemos, atliekančios konfigūracijos užduotis, įvesties/išvesties aprašymas.
- **Konfigūracijos apribojimų ontologija** – tai ontologija, kuri apima informaciją, apibrėžiančią komponentų apribojimus. Šie apribojimai gali atsirasti tiek iš aplinkos, tiek iš reikalavimų ontologijos.



3 pav. Konfigūracijos ontologijos išskaidymo modelis

2.5 Ontologijos aprašymo kalbų analizė

Atlikus ontologijų kalbų analizę, ontologijų kūrimui buvo pasirinkta žiniatinklio ontologijų kalba OWL (angl. *Web Ontology Language*). OWL yra nauja formali kalba, sukurta ontologijų atvaizdavimui semantiniam tinkle [11]. OWL skirta naudoti tuomet, kai dokumentuose saugoma informacija turi būti apdorojama taikomųjų programų, ne tik pateikiama žmonėms. OWL naudojama norint aiškiai pateikti žodynuose esančias išraiškas, jų prasmę bei tarpusavio ryšius. Šios išraiškos ir jų semantiniai tarpusavio ryšiai vadinami ontologija.

OWL turi keleto ankstesnių atvaizdavimo kalbų bruožų, tokių kaip RDF (angl. *Resource Description Framework*). Tačiau OWL turi daugiau galimybių prasmės ir semantikos išreiškimui nei XML (angl. *eXtensible Markup Language*), RDF ir RDFS (angl. *Resource Description Framework Schema*), todėl OWL pirmauja prieš šias kalbas savo galimybėmis pateikti tinkle mašinos apdorojamą turinį [12]. OWL pirmiausiai buvo sukurta atvaizduoti informaciją apie objektų kategorijas ir kaip objektai yra tarpusavyje susiję – šis informacijos tipas dažnai vadinamas ontologija. Taip pat OWL gali atvaizduoti informaciją apie pačius objektus – ši informacija dažnai laikoma duomenimis [13].

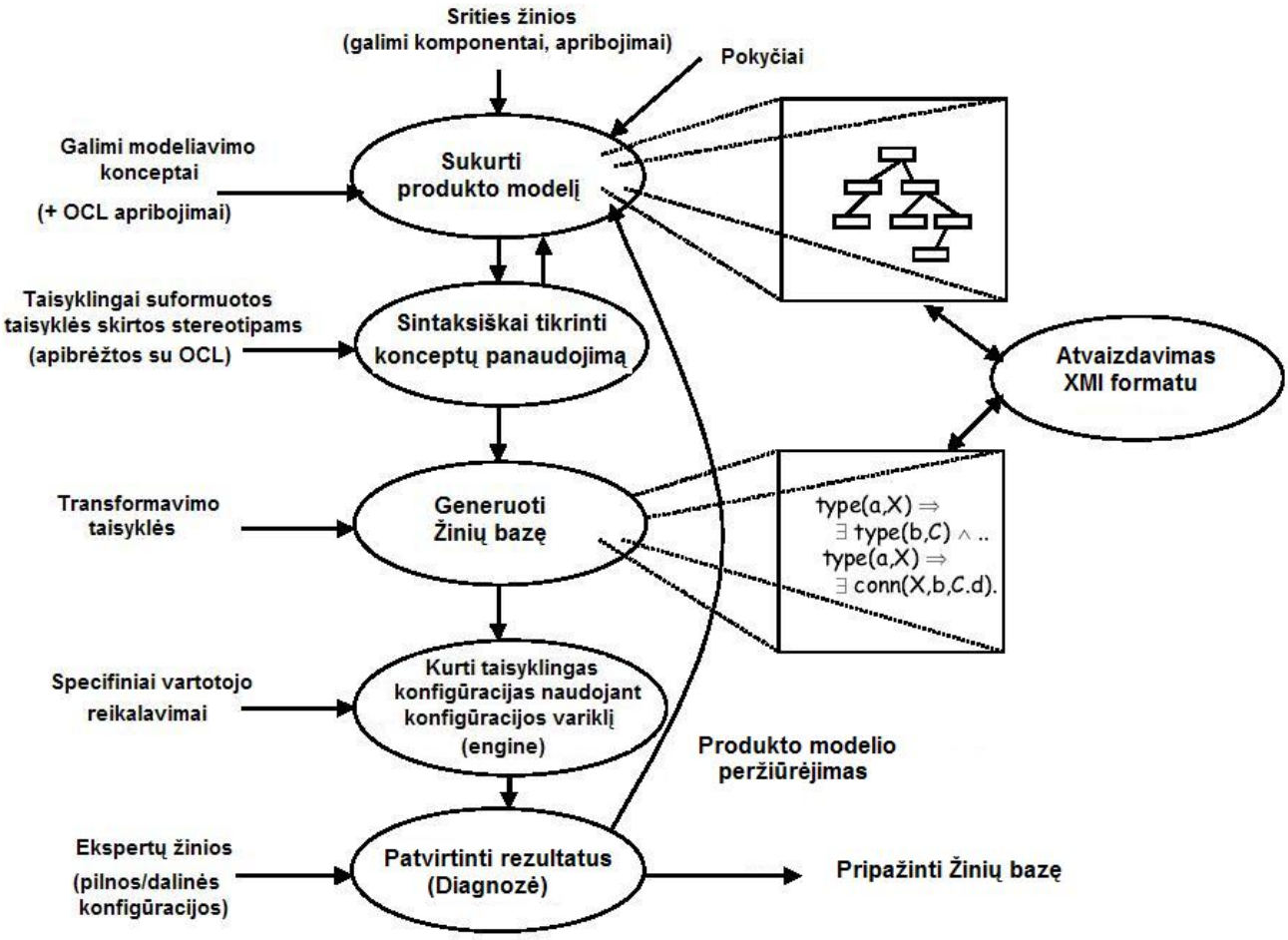
Lentelėje 1 yra pateikiamas ontologijų aprašymo kalbų savybių palyginimas.

Lentelė 1. Ontologijų aprašymo kalbų palyginimas

	XML DTD	XML Schema	RDF(S)	OWL
Susieti sąrašai (bounded lists) (pvz.: žinoma, kad X turi tiksliai 5 vaikus)			X	X
Kardinalumo apribojimai	X	X		X
Klasių išraiškos (pvz.: unionOf, complementOf)				X
Duomenų tipai		X	X	X
Apibrėžtos klasės				X
Išvardinimai (enumerations)	X	X		X
Ekvivalentiškumas (savybių, klasių, egzempliorių)				X
Išplečiamumas			X	X
Formali semantika			X	X
Paveldėjimas			X	X
Išvedimas (inference) (tranzityvumas, inversija)				X
Vietiniai apribojimai (pvz.: visi vaikai yra asmens tipo)				X
Sudaiktinimas (reification)			X	X

2.6 Literatūros šaltiniuose pateiktų sprendimų problemai spręsti lyginamoji analizė

Yra keletas alternatyvų, kaip galėtų būti kuriama žinių bazė. Felfernig [1] siūlomas konfigūracijos žinių bazės kūrimo procesas yra pateikiamas 4 pav. Pirmiausiai, naudojant kokią nors modeliavimo kalbą, pavyzdžiui UML (angl. *Unified Modelling Language*), yra sukuriamas konfigūruojamo produkto modelis. Po sintaksinio konceptų korektiškumo tikrinimo šis modelis nedviprasmiškai transformuojamas į logines išraiškas, kurios yra naudojamos konfigūracijos variklio, kuris generuoja produktų konfigūracijas. Galiausiai žinių bazė yra tikrinama dalykinės srities eksperto, kuris testuoja sukurtą žinių bazę, naudodamas kontrolinius pavyzdžius.



4 pav. Taisyklingos konfigūracijos žinių bazės konstravimas

Vienas iš darbo tikslų yra sukurtą konfigūracijos ontologiją išsaugoti reliacinėje duomenų bazėje. Tam reikia sukurti transformacijos algoritmą, kuris nagrinėdamas OWL failą galėtų generuoti DDL skriptą. Tarp XML ir OWL dokumentų yra daug panašumų, todėl verta panagrinėti keletą literatūroje siūlomų XML dokumentų atvaizdavimo į reliacinę duomenų bazę požiūrių.

- ***Lanko***. Paprasčiausia schema, pasiūlyta Florescu [1], yra patalpinti visus atributus į vieną lentelę, vadinamą „lanko“ (angl. Edge) lentele. „Lanko“ lentelė saugo atributo šaltinio ir tikslo objektų identifikatorius (oids), atributo vardą, vėliavėlę, kuri identifikuoja, ar atributas yra vidinė objekto nuoroda, ar rodo į reikšmę, taip pat eilės numerį, naudojamą atstatant visus objekto atributus teisinga tvarka, bei išvengiant klaidų, jeigu objektas turi keletą atributų su tuo pačiu vardu. Taigi „lanko“ lentelė yra tokios struktūros:

  Lankas (šaltinis, eilėsNr, vardas, vėliavėlė, tikslas).

  Raktiniai „lanko“ lentelės laukai yra *šaltinis* ir *eilėsNr*.

- ***Atributo***. Atributo atvaizdavimo schema sugrupuoja visus atributus su tuo pačiu vardu į vieną lentelę. Konceptiškai šis požiūris horizontaliai suskaido „lanko“ lentelę, o skaidymo atributu naudojamas *vardas*. Taigi, atributų lentelės yra sukuriamos kiekvienam skirtingam XML dokumento atributo vardui. Kiekviena lentelė yra tokios struktūros:
  Avardas (šaltinis, eilėsNr, vėliavėlė, tikslas)
  Atributo lentelės raktiniai laukai yra *šaltinis* ir *eilėsNr*, o visi kiti laukai turi tokią pačią prasmę kaip ir ankstesniame požiūryje.

- ***Universalusis***. Universali lentelė saugo visus XML dokumento atributus. Jeigu n1, n2,…,nk yra XML dokumento atributų vardai, tuomet universali lentelė atrodys taip:
  Universal(šaltinis, eilėsNrn1, vėliavėlėn1, tikslasn1, eilėsNrn2, vėliavėlėn2, tikslasn2, . . ., eilėsNrnk, vėliavėlėnk, tikslasnk)
  Universali lentelė turi daug laukų, kurie yra tušti, be to, ji turi riziką būti perpildyta.

Yra ir daugiau požiūrių, pavyzdžiui, normalizuotas universalusis bei kiti, tačiau nei vienas neišskiriamas kaip geriausias ar siūlomas naudoti.

A. Gali [2] pasiūlė požiūrį, kaip sukurti sistemą, kuri atvaizduotų OWL ontologiją į reliacinės duomenų bazės schemas išsaugant apribojančią informaciją. Sistema susideda iš trijų dalių – ontologijų modeliavimo, dokumentų valdymo ir ontologijų resursų valdymo.

Ontologijų modeliuotojas paima OWL dokumentus kaip įeigą ir sukuria ontologijos modelį, kuris yra panašus į DOM (angl. *Document Object Model*) naudojamą XML dokumentuose. Nagrinėjant OWL dokumentus visi apribojimai taip pat būtų aptikti ir išsaugoti.

Dokumentų valdytojas padeda apdoroti OWL dokumentus. Jis sukuria ryšį tarp įkeltų dokumentų, taip pat sukuria naujus modelius skirtus įkeltiems dokumentams.

Ontologijų resursų valdytojas suteikia metodus, skirtus resursų tipų gavimui ir nustatymui. Savybė rdf:type apibrėžia daugybę įvairių ontologijų kalbų semantinių modelių taisyklių.

Taip pat buvo sukurtas OWL2DB algoritmas [2], kuris atvaizduoja OWL dokumentus į reliacinės duomenų bazės lenteles:

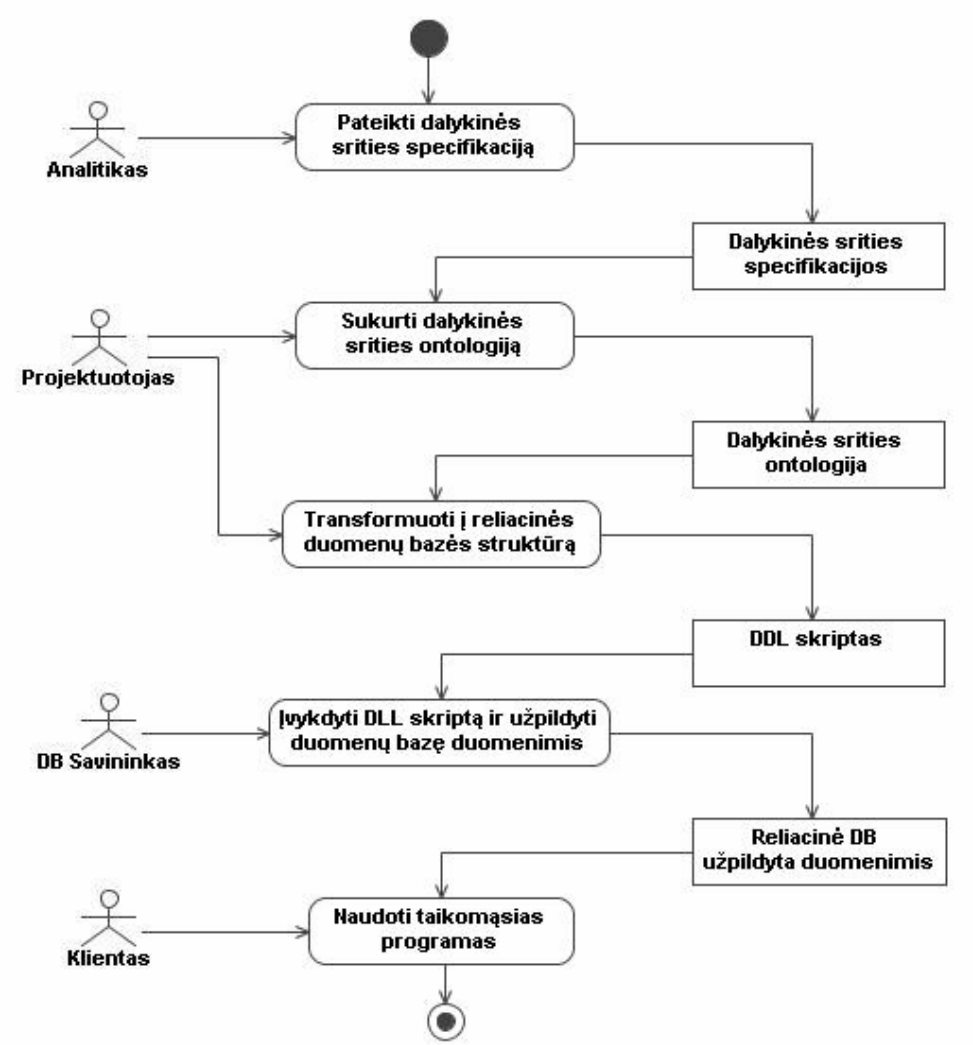
1. OWL failas pirmiausiai nagrinėjamas ieškant šakninių (Root) klasių. Grąžinamas rezultatas yra iteracijų skaičius per visas šaknines klases.
2. Atliekama iteracija per visas šaknines klases tam, kad nustatyti palikuonių gylį šakninėse klasėse.

3. Iškviečiamas *ShowClass* metodas, kuris tikrina poklasės gylį nuo šaknies. Kai grafo gylis pasiekia 3, tuomet sukuriamos reliacinės lentelės visoms poklasėms, esančioms virš jos. Atributų vardai/egzemplioriai tampa stulpelių vardais.
4. Ankstesnis žingsnis kartojamas tol, kol visoms poklasėms, esančioms gylyje 3 nuo šaknies yra surenkami lentelių ir atributų vardai.
5. *Union/complement/disjoint* poklasių gylis tame pačiame lygyje yra toks pats.
6. Klasėms, kurios yra nesikertančios, sukuriami individualūs ryšiai.
7. Surinkus lentelių vardų ir atitinkamų atributų informaciją, reikia surinkti egzempliorių reikšmes tam, kad užpildyti duomenų bazę. Grafas vėl yra nagrinėjamas, išrenkamos reikšmės, kurios saugomos duomenų struktūrose.
8. Galiausiai prisijungiama prie duomenų bazės, sukuriamos visos lentelės ir įrašomos reikšmės.

Kadangi šis metodas yra skirtas atvaizduoti ontologijas, išreikštas OWL, į reliacinę duomenų bazę, todėl jo idėjos bus naudojamos šiame darbe. Produkto konfigūracijos ontologija, aprašyta OWL kalba, bus transformuota į DDL (angl. *Data Definition Language*) skriptą, kuris bus naudojamas sukuriant reliacinę duomenų bazę.

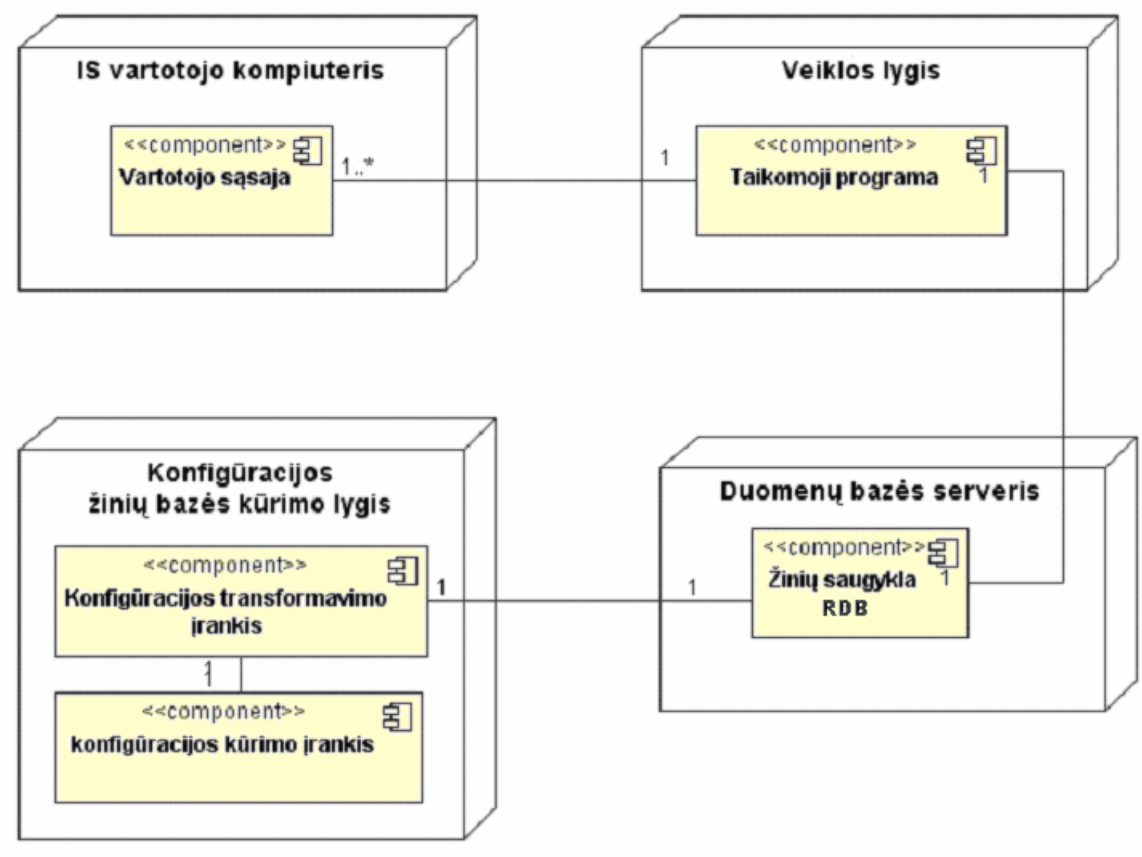
2.7 Dalykinės srities žinių bazės kūrimo procesas ir galimas architektūros modelis

Šiame darbe siūlomas dalykinės srities žinių bazės kūrimo proceso modelis pateikiamas 5 pav. [5], [6]. Dalykinės srities analitikas pateikia IS projektuotojui dalykinės srities specifikacijas, pavyzdžiui, produkto konfigūraciją natūralia kalba. Projektuotojas, naudodamas modeliavimo įrankį, sukuria formalų dalykinės srities modelį (ontologiją). Naudojant transformacijos įrankį, ontologijos aprašas su visais apribojimais atvaizduojamas į DDL skriptą, kuris naudojamas dalykinės srities modeliui išsaugoti reliacinėje duomenų bazėje. Kliento taikomoji programa kreipiasi į duomenų bazę ir, naudodamasi ten esančiais produkto duomenimis bei metaduomenimis, pateikia rezultatus vartotojui (6 pav.).



5 pav. Dalykinės srities žinių bazės kūrimo veiklos proceso modelis

Pagal 5 pav. pavaizduotą dalykinės srities žinių bazės kūrimo veiklos procesą galima sudaryti jo galimos realizacijos architektūros modelį (6 pav.). Ji galima išskaidyti į keturias dalis: informacinės sistemos vartotojas, taikomųjų programų (veiklos) lygis, žinių saugykla ir žinių bazės kūrimo (bei atnaujinimo) lygis. IS inžinierius, naudodamas konfigūracijos ontologijos kūrimo bei transformavimo įrankius, sukuria žinių saugyklą. Klientas per vartotojo sąsają bendrauja su taikomąja programa, kuri, imdama informaciją iš žinių saugyklos, suranda bei pateikia atsakymus klientui.

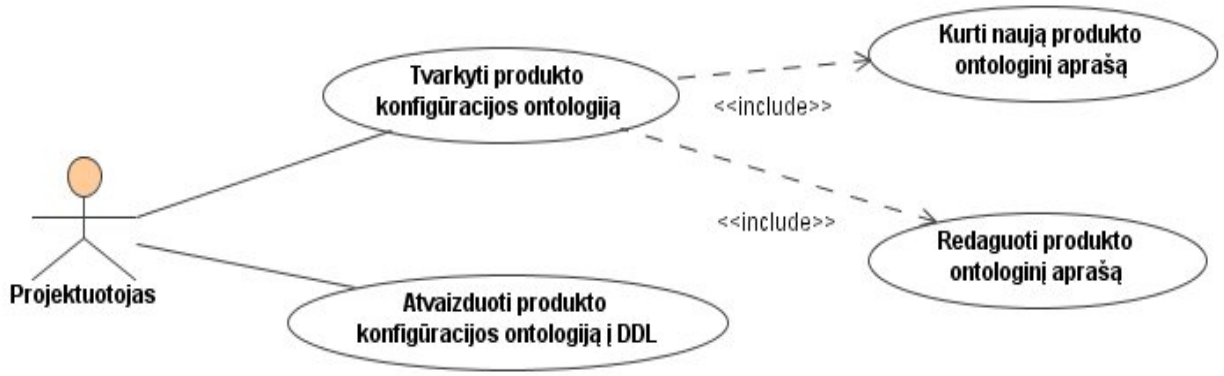


6 pav. Veiklos proceso realizacijos architektūros modelis

2.8 Transformacijos sistemos vartotojų analizė

Kuriamo ontologijų modeliavimo ir atvaizdavimo įrankio vartotojas bus kliento informacinės sistemos duomenų bazės projektuotojas. Jis naudos šį įrankį tam, kad neformalias produktų specifikacijas galėtų ontologijų pagalba aprašyti formaliai, taip pat gautus formalius aprašus atvaizduoti į reliacinę duomenų bazę.

Vartotojo poreikiai parodyti 7 pav.



7 pav. Vartotojo poreikių modelis

2.9 Projekto tikslas ir jo pagrindimas, kokybės kriterijų apibrėžimas

Projekto tikslas yra sukurti arba pritaikyti programinę įrangą, kuri leistų tiriamą dalykinę sritį aprašyti ontologija ir gautą ontologiją transformuoti į DDL (angl. *Data Definition Language*) skriptą, kuris vėliau bus paverstas į reliacinę duomenų bazę.

Kokybės kriterijai:

- Išorinės ir vidinės automatizuotos prieigos prie įmonės informacijos galimybių padidinimas;
- Įmonės veiklos standartizavimas;
- Sąveikos su kitomis sistemomis standartizavimas;
- Verslo efektyvumo padidinimas.

Svarbiausias kriterijus yra verslo efektyvumo padidinimas. Produktų konfigūracijos atveju konfigūracijos valdymo sistema bus pagrindinė darbuotojų sąsaja su produktų žinių baze. Ji bus naudojama visuose įmonės veiklos etapuose, o ypač klientų poreikių įvertinimui ir problemų analizei.

Pačios ontologijos kaip modeliavimo kalbos kriterijai:

- Minimalumas – ontologijoje turi būti atvaizduoti tik reikalingi konceptai;
- Išraiškingumas (šimto procentų principas) – turi būti atvaizduoti visi reikiami dalykinės srities konceptai
- Aiškumas - ontologija turi būti nedviprasmiška, diagramų arba tekstinių išraiškų prasmė intuityviai akivaizdi, o kalbos konceptai ir notacijos lengvai išmokstamos ir atsimenamos.
- Semantinis stabilumas – galimybė išlikti dalykinės srities pasikeitimų fone.
- Semantinis tinkamumas - modeliuojamos tik konceptualiai tinkamos dalykinės srities esybės.
- Patikrinamumas - srities ekspertai turi galėti patikrinti, ar modelis atitinka dalykinę sritį.
- Abstrakcijos mechanizmai – galimybė paslėpti nepageidaujamas detales.
- Formalus pagrindas užtikrina, kad modeliai yra nedviprasmiški ir vykdomi.

2.10 Projektavimo metodų, priemonių parinkimas

Renkantis projektavimo priemones buvo atlikta ontologijų projektavimo priemonių analizė. Nors rinkoje šiuo metu yra nemažai ontologijų aprašymo įrankių, išsami analizė atlikta tarp dviejų lyderių: „*Altova Semantic Works 2006*“ ir „*Protege*“.

„Altova SematicWorks 2006“ yra vizualus RDF/OWL redaktorius. Juo galima vizualiai projektuoti semantinio tinklo dokumentus, žodynus ir ontologijas, po to juos konvertuoti į RDF/XML formatus arba į N-triple - vieną iš standartinių RDF serializacijos formatų.

SemanticWorks 2006 pagreitina darbą naudodamas lenteles egzemplioriams, savybėms, klasėms, kontekstui jautrius įvedimo pagalbininkus ir automatinį formato tikrinimą.

„Altova SemanticWorks 2006“ leidžia grafiškai kurti ir redaguoti RDF dokumentus, RDFS žodynus ir OWL ontologijas su pilna sintakse. Jautrūs kontekstui įvedimo pagalbininkai pateikia sąrašą leistinų pasirinkimo variantų priklausomai nuo to, ar RDF ar OWL kalba naudojama, taigi galima kurti teisingos sintaksės dokumentus lengvai ir greitai.

„Altova SemanticWorks 2006“ suteikiamos galimybės (Lentelė 2):

- Vaizdinis kūrimas ir redagavimas RDF, RDF Schema (RDFS), OWL Lite, OWL DL ir OWL Full dokumentų naudojant intuityvią, vaizdinę vartotojo sąsają ir „*drag-and-drop*“ funkcionalumą.
- Sintaksės tikrinimas užtikrinant atitikimą su RDF/XML specifikacijomis.
- RDF/XML arba N-triples formatų automatinis generavimas ir redagavimas naudojantis grafiniu RDF/OWL projektu.
- Grafinių RDF ir OWL atvaizdų spausdinimas tam, kad sukurti semantinio tinklo elemento dokumentaciją.

Galima keisti grafinį RDF/OWL vaizdą į tekstinį režimą, norint pažiūrėti, kaip dokumentas atrodo RDF/XML ar N-triples formatu, taip pat galima eksportuoti savo failą iš RDF/XML į N-triples formatą arba atvirkščiai bet kuriuo metu. Kadangi RDF/XML arba N-triples kodas yra automatiškai generuojamas pagal projektą, galima mokytis ir eksperimentuoti su semantinio tinklo konceptais nesigilinant kaip reikia parašyti sudėtingą kodą.

Lentelė 2. Altova Semantic Works 2006” charakteristika

Semantinio tinklo kūrimo palaikymas	▫ grafinis RDF dokumentų redaktorius ▫ grafinis RDF schemų žodynų redaktorius ▫ grafinis OWL ontologijų redaktorius ▫ semantinis ontologijų tikrinimas

RDF/RDFS redagavimas (***RDF/RDFS Editing***)	▫ Pasirinkimas reikiamo RDF lygmens (RDF, RDFS); ▫ Sintaksės tikrinimas; ▫ Grupavimo pagal klases, savybes, egzempliorius peržiūra; ▫ Galimybė paslėpti tuščius mazgus ▫ Dokumentų importavimas ▫ Importuotų dokumentų atidarymas iš naujo ▫ Galimybė keisti peržiūros nustatymus: braižymo kryptį, atstumus ir t.t.; ▫ RDF, RDF schemų, OWL failų atidarymas ir saugojimas N-triple (nt) formatu; ▫ RDF/XML formato eksportavimas į N-triples formatą; ▫ N-triples eksportavimas į RDF/XML formatą; ▫ Patogus spausdinimas RDF/XML failų per RDF/XML eksportavimą; ▫ Sutrumpinimai skirti URI (angl. *Uniform Resource Identifiers*) nuorodoms gali būti patalpinti į specialią lentelę; ▫ Galimybė įvesti URI tiek pilna, tiek sutrumpinta forma;
OWL redagavimas (***OWL Editing***)	▫ Pasirinkimas reikiamo OWL lygmens (OWL Lite, OWL DL, OWL Full); ▫ Sintaksės tikrinimas ▫ Grupavimo peržiūra pagal klases, savybes, egzempliorius, ontologijas; ▫ Konteksto peržiūra, kuri pateikia visas savybes susijusias su pažmėtomis klasėmis ar jų egzemplioriais; ▫ Semantinis OWL Lite ir OWL DL tikrinimas (logiškas tikrinimas) ▫ Galimybė paslėpti tuščius mazgus ▫ Dokumentų importavimas ▫ Importuotų dokumentų atidarymas iš naujo ▫ Galimybė keisti peržiūros nustatymus: braižymo kryptį, atstumus ir t.t.; ▫ RDF, RDF schemų, OWL failų atidarymas ir saugojimas N-triple (nt) formatu; ▫ RDF/XML formato eksportavimas į N-triples formatą; ▫ N-triples eksportavimas į RDF/XML formatą; ▫ Patogus spausdinimas RDF/XML failų per RDF/XML eksportavimą; ▫ Sutrumpinimai skirti URI (Uniform Resource Identifiers) nuorodoms gali būti patalpinti į specialią lentelę; ▫ Galimybė įvesti URI tiek pilna, tiek sutrumpinta forma;
Vartotojo sąsaja (***User Interface***)	▫ Nauja ir pagerinta sąsajos išvaizda ▫ Praplėstas šriftų savybių nustatymas, skirtas visiems modeliams ▫ Konfigūruojamas spausdinimas ir spausdinimo peržiūrėjimas ▫ Paieškos ir pakeitimo savybių lankstumas ▫ Įrankių juosta ir nuorodos pritaikytos lanksčiam vartotojų naudojimui ▫ Neribotas atšaukimo/pakartojimo funkcijų naudojimas ▫ Pagerintas kilnojamų/fiksuojamų meniu juostų naudojimas ▫ Daugiadokumentinė sąsaja ▫ Lengvai naudojama, efektyvi kontekstinė vartotojo pagalba;
Palaikomos operacinės sistemos (***Platforms***)	▫ Microsoft Windows application (NT 4.0, 2000, XP, Server 2003)
Tarptautinis palaikymas (***International support***)	▫ Unicode (UTF-7, UTF-8, UTF-16, ISO-10646, UCS-2, UCS-4) ▫ Visi pagrindiniai simbolių rinkinių atkodavimai (ASCII, ISO-8859, CJKV, t.t.) ▫ Bendradarbiaviams tarp skirtingų simbolių rinkinių ir Unicode

Kitas ontologijų kūrimo įrankis „*Protege*“ yra rinkos lyderis dėl savo paplitimo, atviro kodo licencijos ir praplečiamumo.

„*Protege*“ platforma palaiko du pagrindiniu ontologijų kūrimo kelius: naudojant *Protege*-karkasus (angl. *Protege-Frames*) ir *Protege*-OWL redaktorius. „*Protege*“ ontologijos gali būti eksportuojamos į daugybę formatų, įskaitant RDF(S), OWL ir XML Schema. „*Protege*“ sukurta naudojantis Java programavimo kalba, pats produktas yra praplečiamas, turi „*plug and play*“ aplinką.

Lentelė 3. Altova Semantic Works ir Protege palyginimas

	Altova Semantic Works	Protege
Populiarumas	+	+
Ontologijų kūrimas	+	+
OWL palaikymas	+	+
Patogi vartotojo sąsaja	+	+
Platus OS palaikymas	-	+
Nemokamas	-	+
Praplečiamumas	-	+

Abu ontologijų kūrimo įrankiai yra pakankamai galingi ir palaiko ontologijų aprašymo kalbą OWL, tačiau „*Altova Semantic Works*“ yra mokamas komercinis produktas, be to, jo neįmanoma praplėsti, t.y. labiau pritaikyti specifinei tyrimo sričiai. Tuo tarpu „*Protege*“ yra ne tik atviro kodo įrankis, bet ir suteikia plėtinių mechanizmą, kuris bus panaudotas šiame darbe siekiant sukurti redaktorių, kuris neleistų kurti ontologinio aprašo netenkinančios konfigūracijos. Remiantis šiais kriterijais nuspręsta ontologijų kūrimui naudoti „*Protege*“ kūrimo įrankį.

2.11 Analizės išvados

Darbe buvo apžvelgtos problemos, susijusios su dalykinės srities ontologijų naudojimu, detaliau nagrinėjant produktų konfigūravimo žinių bazės modeliavimą. Analizės metu padarytos tokios išvados:

1. Ontologijos yra perspektyvus ir įgyvendinamas būdas patobulinti ir supaprastinti žinių modelių kūrimą ir naudojimą informacinėse sistemose viso jų gyvavimo ciklo metu. Kūrimas supaprastėja, kadangi ontologija galima aprašyti tam tikrą objektą, jo savybes, ryšius su kitais objektais, taip pat įvairius apribojimus žmogui artima kalba. Modeliai patobulėja, nes gaunami formalūs, mašinos interpretuojami aprašai. Pavyzdžiui, produktų konfigūracijos atveju ontologijos suteikia pastovią produktų terminologiją ir galimybę generuoti įvairius tų pačių produktų vaizdus priklausomai nuo esančio konteksto.
2. Išnagrinėjus literatūros šaltinius, galima daryti išvadą, kad šiuo metu nėra standartų, kaip informacinėse sistemose konstruoti žinių bazes remiantis ontologijomis. Tačiau yra pavyzdžių, kurių pagrindu galima sukurti savo dalykinės srities žinių bazės konstravimo ciklo modelį.
3. Analizės metu apibrėžtas veiklos proceso modelis, kurio metu iš dalykinės srities analitiko pateiktos dalykinės srities specifikacijos gaunama žinių bazė, naudojama taikomųjų programų. Nustatyta, jog proceso vykdymo metu reikalingi dalykinės srities ontologinio modeliavimo ir formalaus aprašo transformavimo į duomenų bazę įrankiai.
4. Atlikta ontologijas aprašančių kalbų ir ontologijų modeliavimo įrankių analizė, tolesniam tyrimui pasirinkta OWL kalba ir „Protege“ ontologijų valdymo sistema.
5. Išanalizuoti ontologijų atvaizdavimo į reliacinę duomenų bazę metodų ir algoritmų pasiūlymai, iš jų pasirinktas tinkamiausias algoritmas, kurio principai bus realizuojami šiame darbe.
6. Suformuluotas tolesnio tyrimo uždavinys:
    - sukurti ontologinio aprašo OWL transformavimo į reliacinį pavidalą algoritmą;
    - suprojektuoti bei sukurti transformacijos įrankio programinį prototipą;
    - sukurti eksperimentinę konfigūravimo ontologiją;
    - sukurtą ontologinio aprašo OWL transformavimo į reliacinį pavidalą algoritmą eksperimento metu išbandyti su ontologiniu aprašu.

3. Transformacijos sistemos reikalavimų specifikacija

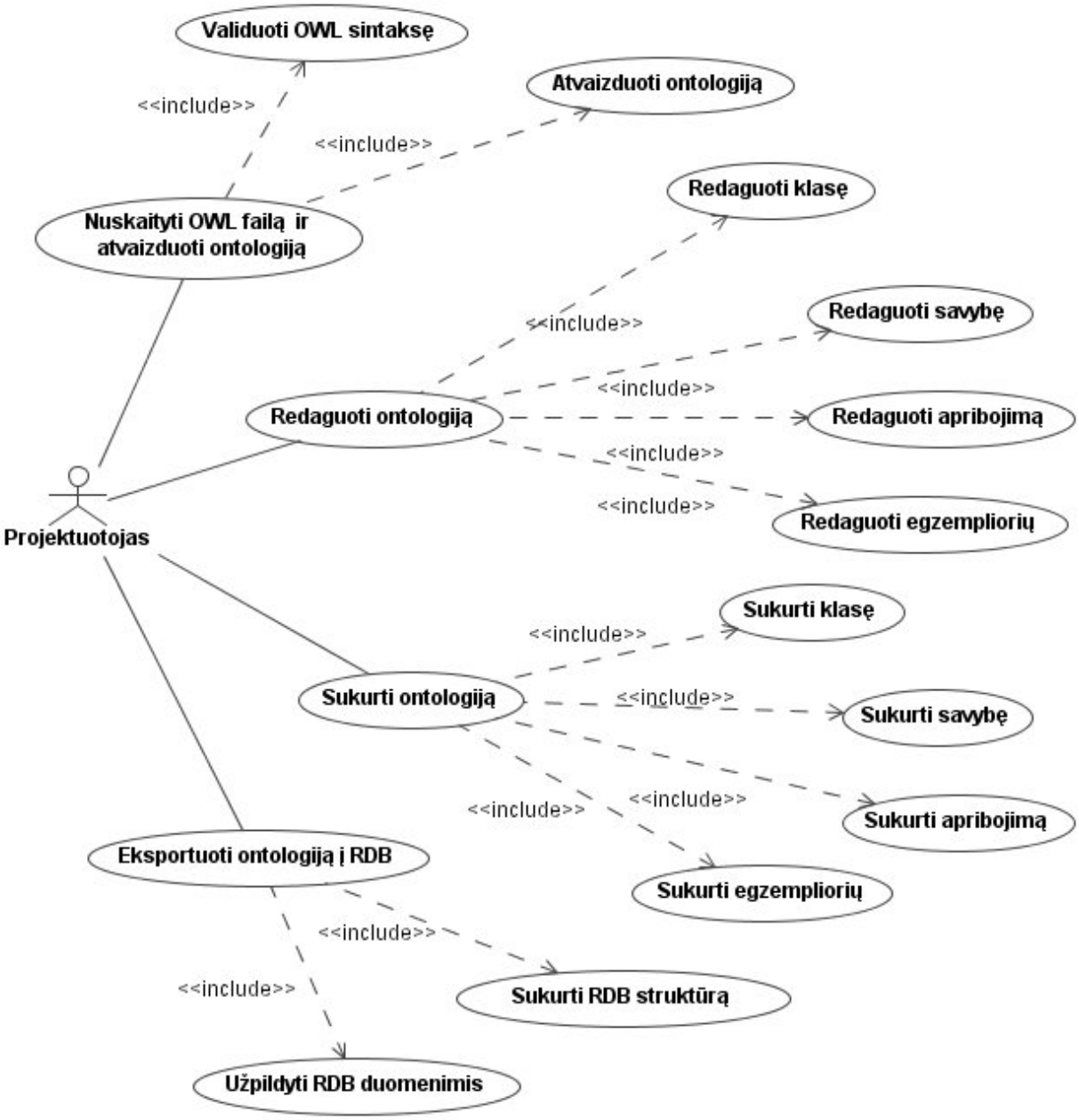
3.1 Transformacijos įrankio nefunkciniai reikalavimai

Transformacijos įrankiui buvo iškelti šie nefunkciniai reikalavimai:

- Sistema turi veikti patikimai, t.y. jos darbas neturi būti nutraukiamas ir duomenys negali būti sugadinami dėl ontologinio aprašo sintaksės klaidų.
- Sistama privalo veikti nepriklausomai nuo naudojamos platformos, t.y. turi būti galimybė ja naudotis įvairiose operacinėse sistemose.
- Sistema turi būti palaikoma, t.y. naujos sistemos galimybės gali būti nesudėtingai įtraukiamos. Turi būti galimybė integruoti transformacijos įrankį su jau egzistuojančiomis ontologijų valdymo sistemomis.
- Sistema privalo teikti pagalbą vartotojui esant nesuprantamai situacijai ar klaidai.
- Sistema turi būti suderinama su įvairiomis duomenų bazių valdymo sistemomis, pvz. MS SQL Server, Oracle, MySql ir kitomis.

3.2 Sistemos panaudojimo atvejai

Ontologijų konstravimo bei transformacijos sistemos funkciniai reikalavimai pavaizduoti panaudojimo atvejų modelyje (8 pav.). Įrankis turėtų nuskaityti ontologijos aprašą OWL kalba, atlikti OWL kalbos sintaksės validavimą bei atvaizduoti modelio struktūrą. Vartotojas turėtų turėti galimybę sukurti objektus bei redaguoti dalykinės srities ontologiją. Pagrindinė sistemos savybė – ontologijos aprašų eksportavimas į reliacinių duomenų bazių schemas.



8 pav. Sistemos panaudojimo atvejų modelis

3.3 Panaudojimo atvejų specifikacijos

PA „Nuskaityti OWL failą ir atvaizduoti ontologiją“ specifikacija

Panaudojimo atvejis	Nuskaityti OWL failą ir atvaizduoti ontologiją
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis OWL ontologijos nuskaitymą.
Numeris	PA1
Aktorius	Projektuotojas
Sistema	Produkto konfigūracijos žinių bazės modeliavimo įrankis
Prieš sąlyga	OWL failas turi būti sukurtas
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas prisijungia prie sistemos 2. Vartotojas naudojasi pagrindiniu programos meniu ir pasirenka nuskaityti OWL ontologiją. 3. Vartotojas pasirenka OWL failą. 4. Vartotojas peržiūri ontologijos turinį	1. Vartotojui parodomas pagrindinis programos langas. 2. Vartotojui parodomas failo pasirinkimo dialogo langas. 3.1. Sistema patikrina OWL sintaksę. 3.2. Sistema nuskaito OWL faile aprašytą ontologiją. 3.3. Ontologija atvaizduojama vartotojui patogia forma.
Po sąlyga	Nuskaityta ontologijos informacija atvaizduota ekrane.
Alternatyvos (nesėkmės atvejai)	1. Vartotojas neranda OWL failo. 2. Failas neatitinka OWL sintaksės. 3. OWL ontologija yra nekorektiška.
Vykdymo variantai	1. Vartotojas prisijungia prie sistemos ir pasirenka nuskaityti OWL ontologiją.
Veiklos taisyklės	Vartotojas privalo pasirinkti owl tipo failą. Jei vartotojas pasirenka kito tipo failą, sistema apie tai informuoja vartotoją ir neleidžia atlikti pasirinkto veiksmo.
Specialūs (nefunkciniai) reikalavimai	Ontologijos informacija turi būti atvaizduota vartotojui patogia forma.
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	-
Ryšiai su kitais PA	Apibendrina panaudojimo atvejus „Validuoti OWL sintaksę“ ir „Atvaizduoti ontologiją“.

PA „Redaguoti ontologiją“ specifikacija

Panaudojimo atvejis	Redaguoti ontologiją
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis produkto konfigūracijos ontologijos redagavimą.
Numeris	PA2
Aktorius	Projektuotojas
Sistema	Produkto konfigūracijos žinių bazės modeliavimo įrankis
Prieš sąlyga	Produkto konfigūracijos ontologija turi būti sukurta, nuskaityta ir atvaizduota sistemoje.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi pagrindiniu programos meniu ir pasirenka redaguoti OWL ontologiją. 2. Vartotojas redaguoja ontologijos klases, savybes, apribojimus ir egzempliorius. 3. Vartotojas išsaugoja redaguotą ontologiją.	1. Vartotojui parodomas produkto konfigūracijos ontologijos redagavimo langas. 2. Sistema atlieka pakeitimus OWL faile, aprašančiame produkto konfigūracijos ontologiją. 3. Sistema išsaugo pakeistą OWL failą su produkto konfigūracijos ontologija.
Po sąlyga	Ontologijos informacija išsaugota OWL faile.
Alternatyvos (nesėkmės atvejai)	1. OWL ontologija yra nekorektiška.
Vykdymo variantai	1.Vartotojas pasirenka redaguoti produkto konfigūracijos ontologiją.
Veiklos taisyklės	Vartotojas privalo įvesti korektiškas duomenų tipų savybių reikšmes, atitinkančias ontologijos aprašą. Jei vartotojas įveda neteisingus duomenis, sistema apie tai informuoja vartotoją ir neleidžia atlikti pasirinkto veiksmo.
Specialūs (nefunkciniai) reikalavimai	Ontologijos redagavimas turi būti atliekamas vartotojui patogia simboline forma.-
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	-
Ryšiai su kitais PA	Apibendrina panaudojimo atvejus „Redaguoti klasę“ , „Redaguoti savybę“, „Redaguoti apribojimą“ bei „Redaguoti egzempliorių“.

PA „Sukurti ontologiją“ specifikacija

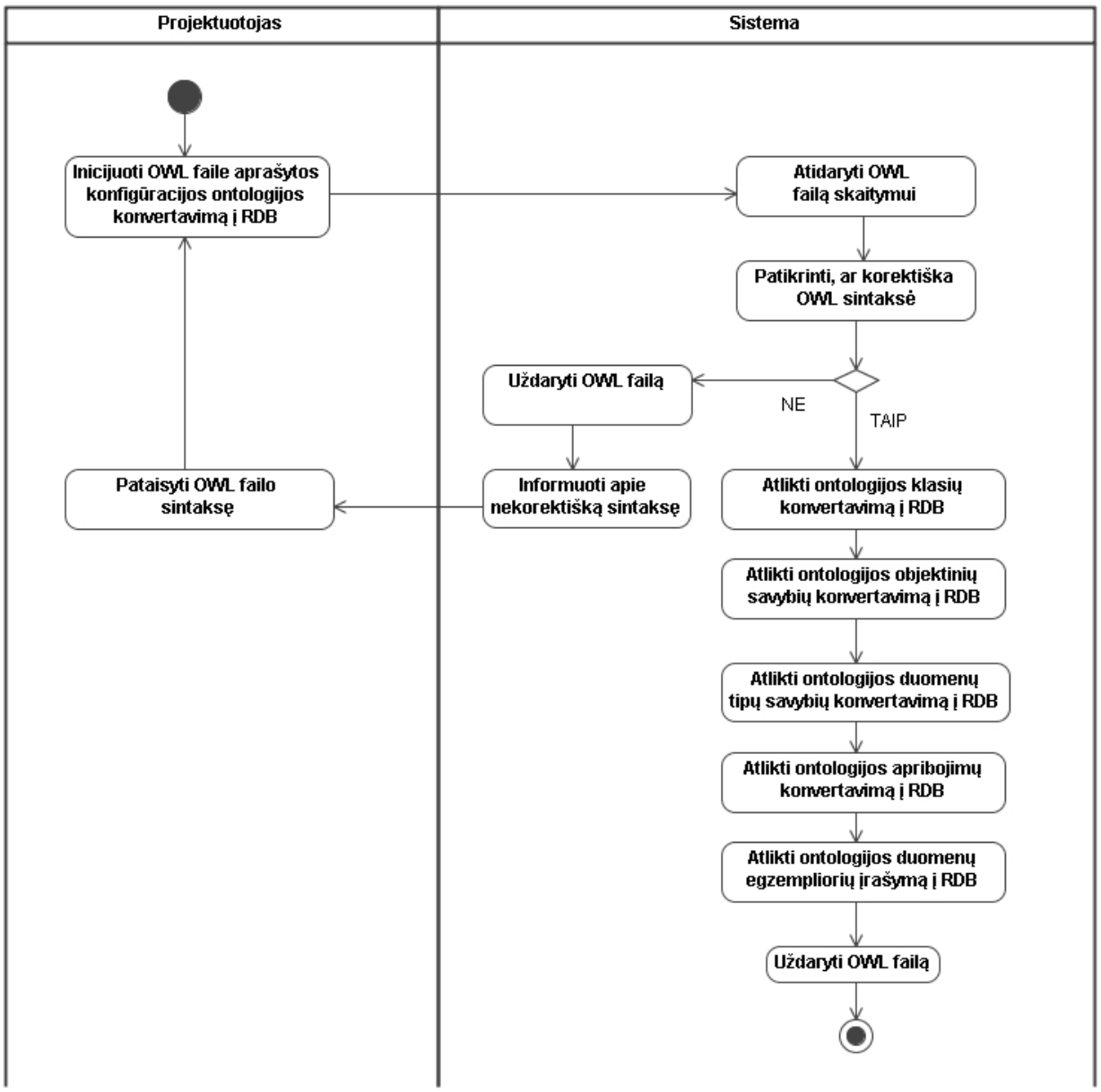
Panaudojimo atvejis	Sukurti ontologiją
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis produkto konfigūracijos ontologijos sukūrimą.
Numeris	PA3
Aktorius	Projektuotojas
Sistema	Produkto konfigūracijos žinių bazės modeliavimo įrankis
Prieš sąlyga	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi pagrindiniu programos meniu ir pasirenka sukurti produkto konfigūracijos ontologiją. 2. Vartotojas sukuria produkto konfigūracijos ontologijos klases, savybes, apribojimus ir egzempliorius. 3. Vartotojas išsaugoja sukurtą produkto konfigūracijos ontologiją.	1.1. Vartotojui parodomas produkto konfigūracijos ontologijos sukūrimo langas. 1.2. Sistema įrašo privalomas ontologijos reikšmes į OWL failą, tokias kaip vardų sritys, šakninė OWL klasė ir t.t. 2. Sistema atlieka pakeitimus OWL faile, aprašančiame produkto konfigūracijos ontologiją. Sukuria klases, savybes, apribojimus ir egzempliorius. 3. Sistema išsaugo sukurtą OWL failą su produkto konfigūracijos ontologija.
Po sąlyga	Produkto konfigūracijos ontologijos informacija išsaugota OWL faile.
Alternatyvos (nesėkmės atvejai)	1. Vartotojas suformuoja produkto konfigūracijos ontologiją, neatitinkančią produkto konfigūracijos specifikacijos.
Vykdymo variantai	1.Vartotojas pasirenka sukurti produkto konfigūracijos ontologiją.
Veiklos taisyklės	Vartotojas privalo įvesti korektiškas duomenų tipų savybių reikšmes, atitinkančias ontologijos aprašą. Jei vartotojas įveda neteisingus duomenis, sistema apie tai informuoja vartotoją ir neleidžia atlikti pasirinkto veiksmo.
Specialūs (nefunkciniai) reikalavimai	Ontologijos konstravimas turi būti atliekamas vartotojui patogia simboline forma.
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	-
Ryšiai su kitais PA	Apibendrina panaudojimo atvejus „Sukurti klasę“ , „Sukurti savybę“, „Sukurti apribojimą“ bei „Sukurti egzempliorių“.

PA „Eksportuoti ontologiją į RDB“ specifikacija

Panaudojimo atvejis	Eksportuoti ontologiją į RDB.
Panaudojimo atvejo aprašymas	Tai panaudojimo atvejis, apibūdinantis produkto konfigūracijos ontologijos, aprašytos OWL faile, atvaizdavimą į reliacinę duomenų bazę.
Numeris	PA4
Aktorius	Projektuotojas
Sistema	Produkto konfigūracijos žinių bazės modeliavimo įrankis
Prieš sąlyga	-
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas naudojasi pagrindiniu programos meniu ir pasirenka eksportuoti OWL ontologiją į reliacinę duomenų bazę. 2. Vartotojas pasirenka eksportuojamą OWL failą, kuriame aprašyta produkto konfigūracijos ontologija. 3. Vartotojas pasirenka duomenų bazių valdymo sistemos serverį ir duomenų bazę. 4. Vartotojas inicijuoja eksportavimo procesą.	1. Vartotojui parodomas dialogo langas, kuriame galima pasirinkti eksportavimui skirtą OWL failą. 2. Sistema patikrina, ar produkto konfigūracijos ontologijos failas atitinka OWL kalbos sintaksę ir struktūrą. 3. Sistema prisijungia prie duomenų bazių valdymo sistemos serverio ir duomenų bazės. 4. Sistema atlieka ontologijos eksportavimo į RDB procesą. Eksportuojamos ontologijos klasės, savybės, apribojimai ir klasių egzemplioriai.
Po sąlyga	OWL failas eksportuotas į reliacinę duomenų bazę.
Alternatyvos (nesėkmės atvejai)	1. Vartotojo pasirinktas failas neatitinka OWL kalbos sintaksės arba struktūros. Vartotojas informuojamas apie klaidą. 2. Duomenų bazių valdymo sistemos serveris neveikia. 3. Sistemai nesuteiktos teisės prisijungti prie duomenų bazių valdymo sistemos serverio.
Vykdymo variantai	1.Vartotojas pasirenka eksportuoti owl faile aprašytą produkto konfigūracijos ontologiją į reliacinę duomenų bazę.
Veiklos taisyklės	1. Vartotojas privalo pasirinkti owl tipo failą. 2. Vartotojas privalo nurodyti teisingą duomenų bazių valdymo sistemos serverio adresą. 3. Sistemai turi būti suteiktos pilnos teisės dirbti su duomenų baze.
Specialūs (nefunkciniai) reikalavimai	Eksportavimas turi būti galimas į įvairias duomenų bazių valdymo sistemas
Kitos sistemos, su kuriomis sąveikauja sistema vykdydama PA	Vartotojo pasirinkta duomenų bazių valdymo sistema.
Ryšiai su kitais PA	Apibendrina panaudojimo atvejus „Sukurti RDB struktūrą“ ir „Užpildyti RDB duomenimis“

4. Ontologijos aprašo OWL transformavimo į reliacinę duomenų bazę algoritmai

Produkto konfigūracijos žinių bazės projektuotojas, konstruodamas produkto konfigūracijos ontologiją, sukuria failą, kuriame yra aprašyta produkto konfigūracija OWL ontologijų aprašymo kalba. Sekančiame etape projektuotojas, naudodamasis konvertavimo įrankiu, transformuoja OWL sintakse aprašytą ontologiją į reliacinę duomenų bazę. Transformavimo algoritmas pateikiamas 9 pav.



9 pav. Ontologijos nuskaitymo į RDB algoritmas

Projektuotojas inicijuoja OWL faile aprašytos produkto konfigūracijos ontologijos konvertavimą į reliacinę duomenų bazę. Sistema atidaro failą skaitymui ir patikrina, ar korektiška OWL failo sintaksė. Jeigu sintaksė neatitinka OWL notacijos, failas uždaromas, projektuotojas informuojamas apie klaidas. Jeigu sintaksė atitinka OWL notaciją, sistema vykdo produkto konfigūracijos, aprašytos OWL sintakse, konvertavimo į reliacinę duomenų bazę žingsnius. Pirmiausiai atliekamas ontologijos klasių konvertavimas, toliau vykdomas objektinių ir duomenų tipų savybių konvertavimas, įrašomi ontologijos klasių ir savybių apribojimai bei galiausiai vykdomas duomenų bazės lentelių užpildymas ontologijos egzemplioriais. Baigus transformaciją OWL failas uždaromas.

4.1 Ontologijos klasių transformavimo į RDB algoritmas

Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, pirmiausiai vykdomas ontologijos klasių transformavimas į reliacinės duomenų bazės lenteles.

Ontologijos klasė OWL sintakse aprašoma taip:

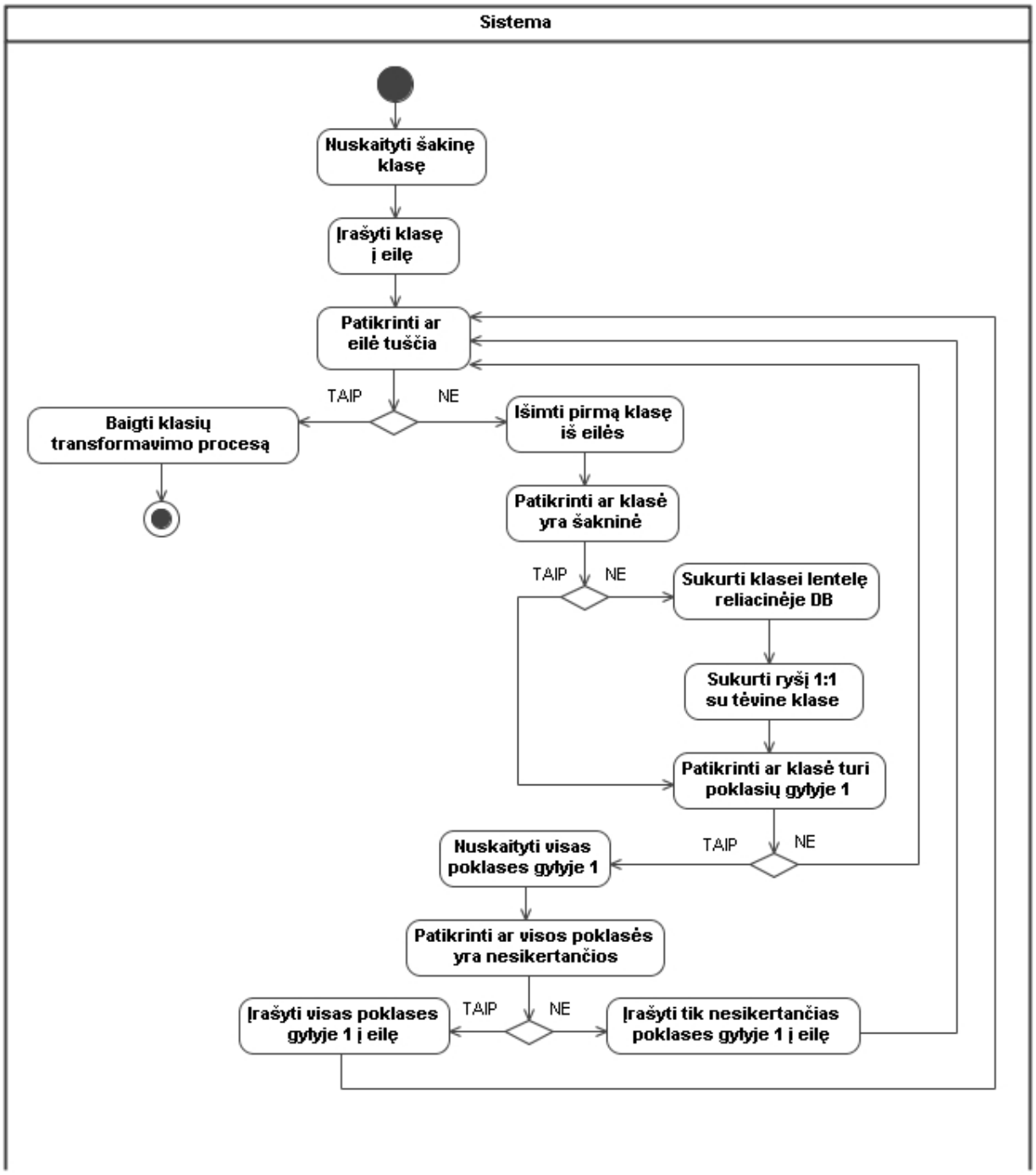
```
<owl:Class rdf:ID="Vynuogynas"/>
<owl:Class rdf:ID="Regionas"/>
<owl:Class rdf:ID="ValgomasDaiktas"/>
```

Pagrindinis klasių taksonomijos konstruktorius yra `rdfs:subClassOf`. Jis susieja labiau specializuotą klasę su bendresne klase. Jeigu X klasė yra Y klasės poklasė, tuomet kiekvienas X klasės egzempliorius yra taip pat Y klasės egzempliorius. Ryšys `rdfs:subClassOf` yra tranzityvus. Jeigu X klasė yra Y klasės poklasė ir Y yra Z klasės poklasė, tuomet X yra Z klasės poklasė. OWL sintaksės pavyzdys, aprašantis klasių hierarchinius ryšius:

```
<owl:Class rdf:ID="GeriamasSystis">
  <rdfs:subClassOf rdf:resource="#ValgomasDaiktas" />
  ...
</owl:Class>
```

Visos klasės ontologijoje yra klasės „Thing“ poklasės.

Algoritmas, transformuojantis ontologijos klases į reliacinės duomenų bazės lenteles, pateikiamas 10 pav.



10 pav. Ontologijos klasių transformavimo į RDB algoritmas

Algoritmas, transformuojantis ontologijos klases į reliacinę duomenų bazę, naudoja paieškos į plotį algoritmą. Paieškos į plotį algoritmas pereina visas ontologijos klases „skersai“ klasių medžio, t.y. pirmiausiai nagrinėjamos aukščiausio hierarchinio lygio klasės, toliau jų poklasės ir t.t. Kiekvienai ontologijos klasei duomenų bazėje sukuriama lentelė, klasių hierarchiniai ryšiai užtikrinami 1:1 ryšiais tarp klasės ir jos poklasės. Kadangi naudojamas paieškos į plotį algoritmas, garantuojama, kad kuriant poklasės lentelę hierarchiškai už ją aukštesnės klasės lentelė jau bus sukurta.

4.2 Ontologijos objektinių savybių transformavimo į RDB algoritmas

Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, po ontologijos klasių transformavimo į reliacinės duomenų bazės lenteles vykdomas objektinių klasių savybių transformavimas į ryšius tarp klasių lentelių.

Objektinė savybė yra ryšys tarp dviejų klasių egzempliorių. Sukuriant savybę, jos apribojimą galima apibrėžti keliais būdais: galima nurodyti domeną (rdfs:domain) ir sritį (rdfs:range); galima nurodyti, kad savybė yra tam tikros kitos savybės specializacija. Pavyzdys, rodantis, kaip aprašoma objektinė savybė OWL sintakse:

```
<owl:ObjectProperty rdf:ID="pagamintasIsVynuogiu">
  <rdfs:domain rdf:resource="#Vynas"/>
  <rdfs:range rdf:resource="#Vynuoge"/>
</owl:ObjectProperty>
```

Pavyzdys, rodantis, kaip aprašoma savybių hierarchija OWL sintakse:

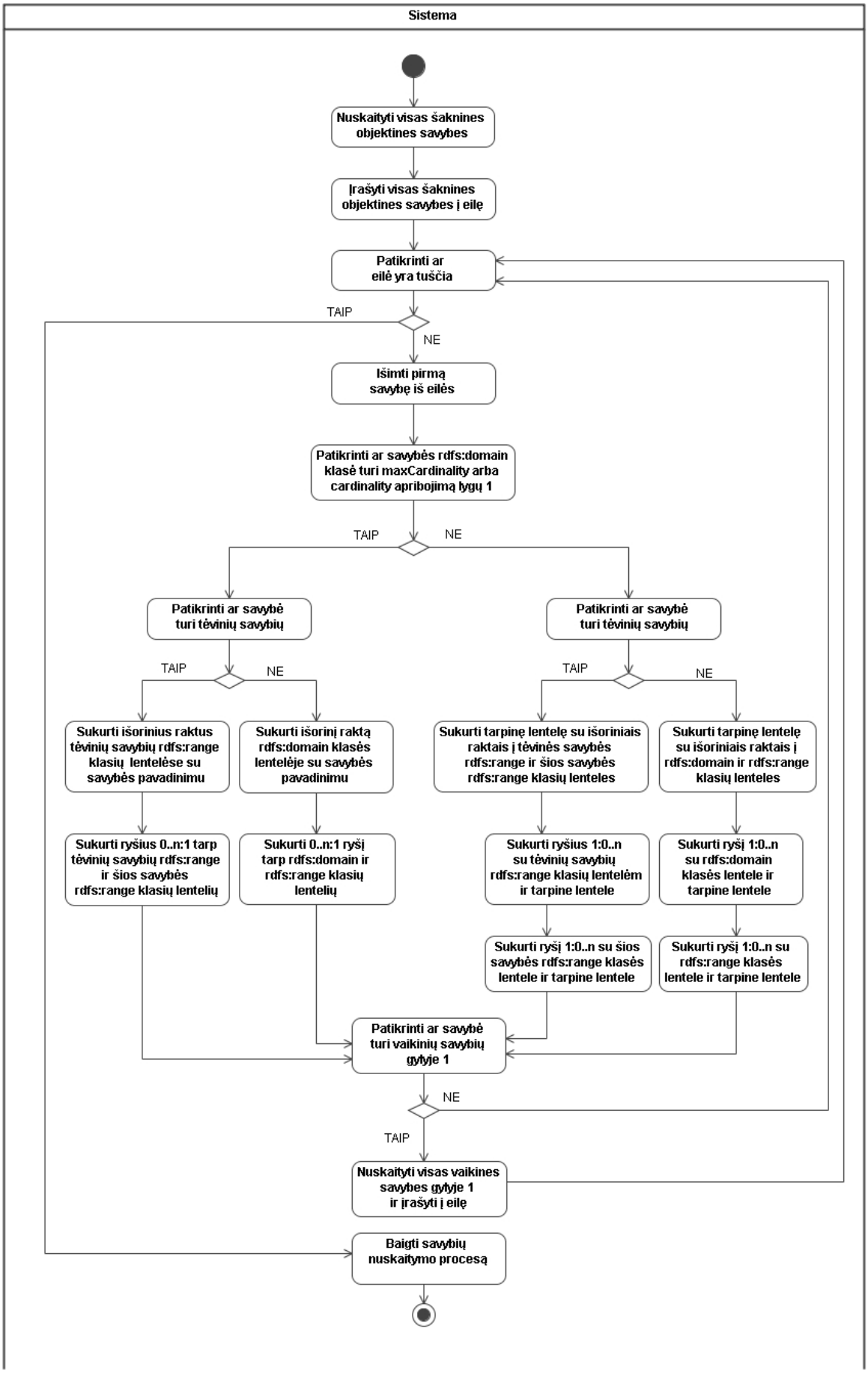
```
<owl:Class rdf:ID="VynoAprasas" />

<owl:Class rdf:ID="VynoSpalva">
  <rdfs:subClassOf rdf:resource="#VynoAprasas" />
  ...
</owl:Class>

<owl:ObjectProperty rdf:ID="turiVynoAprasa">
  <rdfs:domain rdf:resource="#Vynas" />
  <rdfs:range  rdf:resource="#VynoAprasas" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="turiSpalva">
  <rdfs:subPropertyOf rdf:resource="#turiVynoAprasa" />
  <rdfs:range rdf:resource="#VynoSpalva" />
  ...
</owl:ObjectProperty>
```

Algoritmas, transformuojantis ontologijos objektines savybes į reliacinės duomenų bazės ryšius tarp lentelių pateikiamas 11 pav.



11 pav. Ontologijos objektinių savybių transformavimo į RDB algoritmas

Algoritmas, transformuojantis ontologijos savybes į reliacinės duomenų bazės ryšius tarp lentelių, taip pat naudoja paieškos į plotį algoritmą. Pirmiausiai nagrinėjamos savybės, kurios neturi hierarchiškai aukštesnių savybių, toliau jas paveldinčios savybės ir t.t. Priklausomai nuo klasės lokalaus kardinalumo apribojimo savybei, sukuriamas vienas-su-daug arba daug-su-daug ryšys tarp klasių lentelių. Daug-su-daug ryšio atveju sukuriama tarpinė lentelė, taip daug-su-daug ryšį pakeičiant dviem vienas-su-daug ryšiais.

4.3 Ontologijos duomenų tipų savybių transformavimo į RDB algoritmas

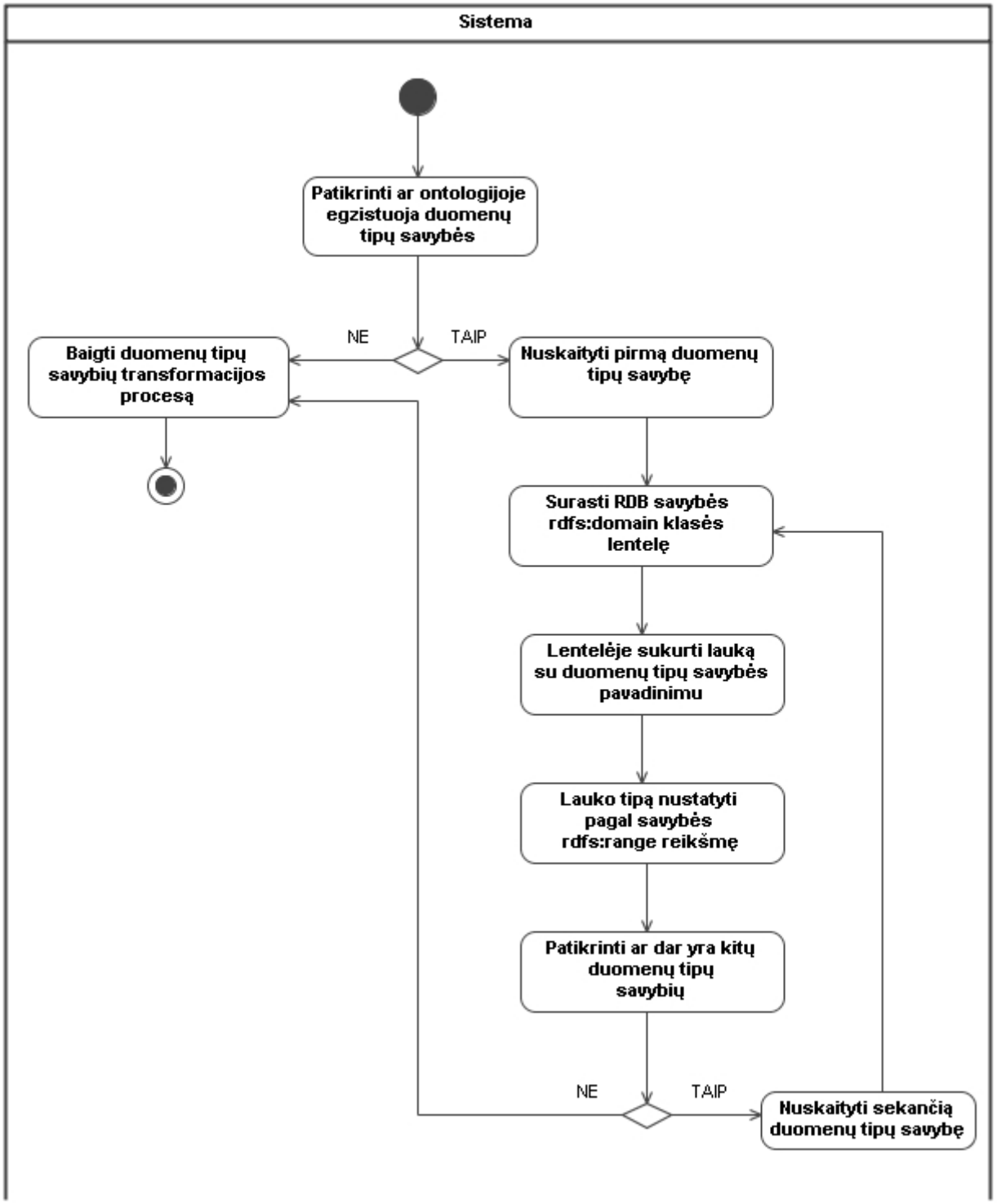
Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, po ontologijos objektinių savybių transformavimo į reliacinės duomenų bazės ryšius tarp lentelių, vykdomas duomenų tipų klasių savybių transformavimas į klasių lentelių duomenų stulpelius.

Duomenų tipų savybės ontologijoje susieja klasių egzempliorius su duomenų tipais. Duomenų tipų savybės reikšmių sritis yra xml schemos duomenų tipai. Pavyzdys, rodantis, kaip OWL sintakse aprašomos duomenų tipų savybės:

```
<owl:Class rdf:ID="DerliausMetai" />

<owl:DatatypeProperty rdf:ID="metai">
  <rdfs:domain rdf:resource="#DerliausMetai" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

Algoritmas, transformuojantis ontologijos duomenų tipų savybes į reliacinės duomenų bazės lentelių stulpelius pateikiamas 12 pav.



12 pav. Ontologijos duomenų tipų savybių transformavimo į RDB algoritmas

Algoritmas, transformuojantis ontologijos duomenų tipų savybes į reliacinės duomenų bazės lentelių stulpelius, analizuoja visas duomenų tipų savybes. Pagal rdfs:domain reikšmę surandama duomenų bazės lentelė, lentelėje sukuriamas stulpelis su savybės pavadinimu. Lentelės stulpelio laukams nustatomas duomenų tipas pagal savybės rdfs:range reikšmę. Patikrinus visas duomenų tipų savybes, algoritmo darbas baigiamas.

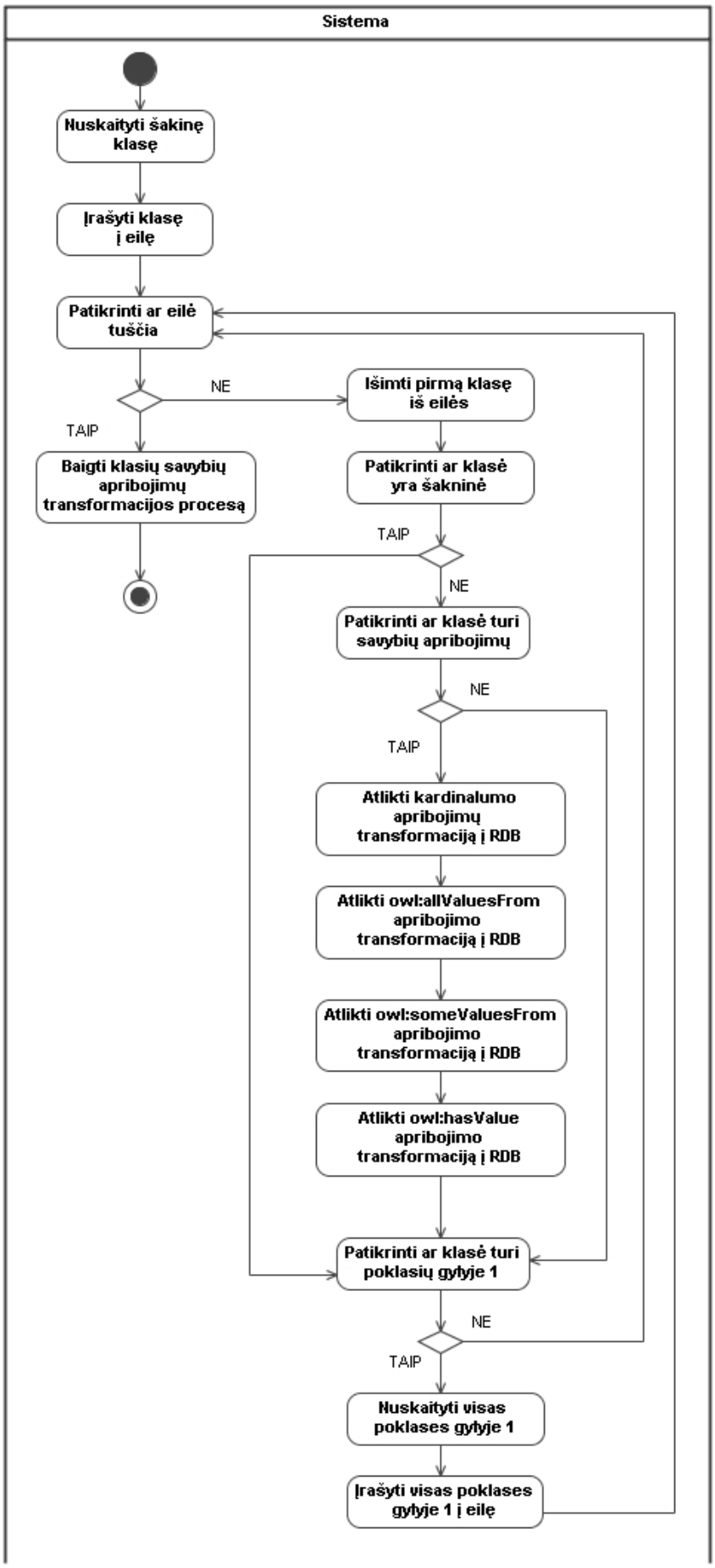
4.4 Ontologijos apribojimų transformavimo į RDB algoritmas

Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, po ontologijos duomenų tipų savybių transformavimo į reliacinės duomenų bazės lentelių stulpelius vykdomas ontologijos apribojimų įrašymas į reliacinės duomenų bazės metaduomenų lenteles.

Ontologijoje apribojimas yra taikomas klasės ir savybės santykyje, t.y. nustatomas savybės apribojimas, tačiau jis galioja tik tam tikrai klasei lokaliai, bet neliečia kitų klasių, kurios turi šią savybę. Pavyzdys, rodantis, kaip OWL sintakse aprašoma savybės apribojimas klasei:

```
<owl:Class rdf:ID="Vynas">
  <rdfs:subClassOf rdf:resource="&maistas;GeriamasSkystis"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#pagamintasIsVynuogiu"/>
      <owl:minCardinality
         rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:minCardinality>
      </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Algoritmas, transformuojantis ontologijos apribojimus į reliacinės duomenų bazės metaduomenų lenteles, pateikiamas 13 pav.



13 pav. Ontologijos apribojimų transformavimo į RDB algoritmas

Algoritmas, transformuojantis ontologijos apribojimus į reliacinės duomenų bazės metaduomenų lenteles, vykdo paiešką į plotį. Pirmiausiai patikrinama šakninių klasių savybių apribojimai, toliau jų poklasių ir t.t. Jeigu klasė turi savybių apribojimų, tuomet vykdomi apribojimų įrašymo į metaduomenų lenteles algoritmai. Išskiriami keturių tipų apribojimai – kardinalumo, „owl:allValuesFrom“, „owl:someValuesFrom“ bei „owl:hasValue“.

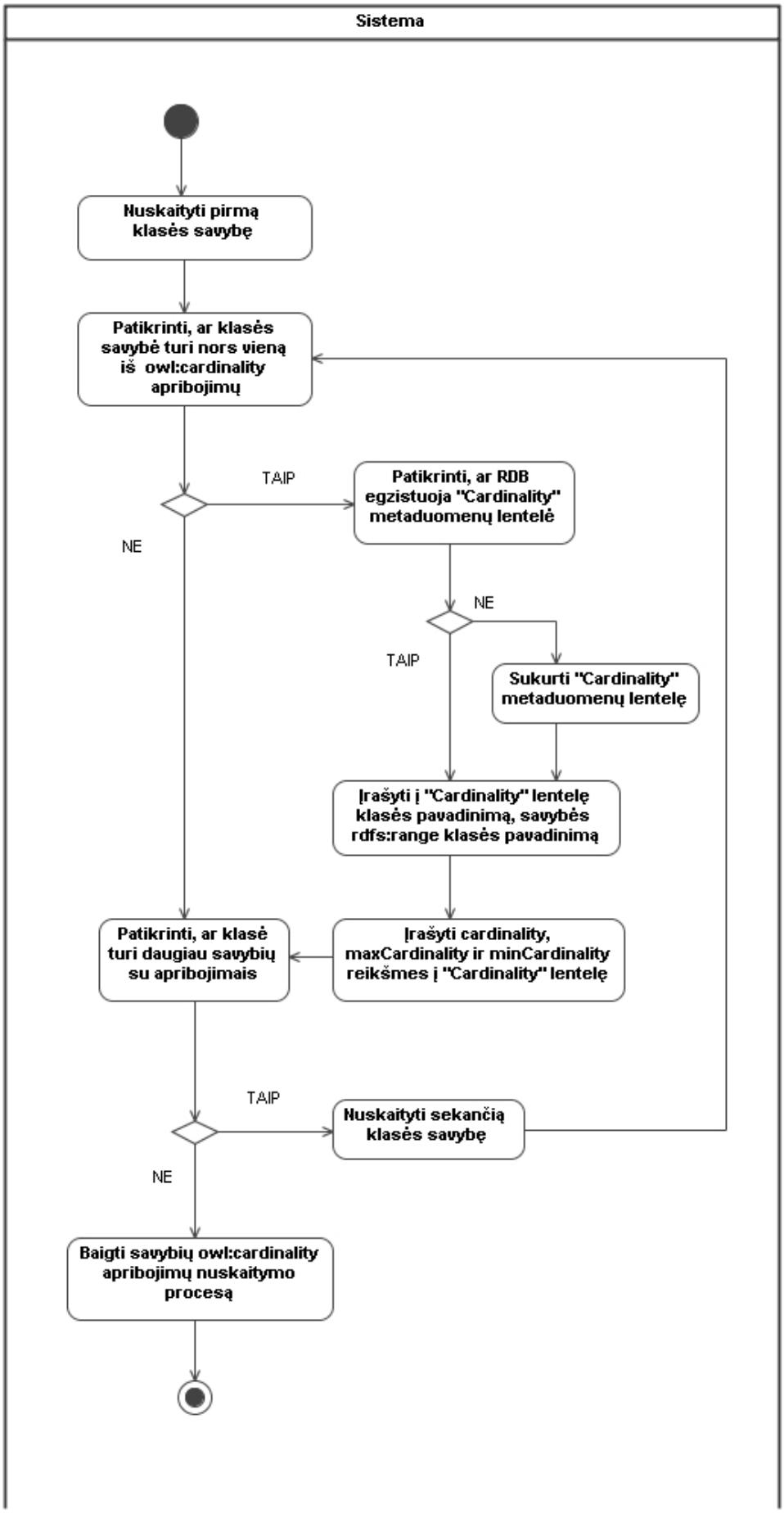
4.4.1 Ontologijos kardinalumo apribojimų transformavimo į RDB algoritmas

Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, ontologijos klasių savybių apribojimų transformavimo etape pirmiausiai atliekamas kardinalumo apribojimų išsaugojimas reliacinėje duomenų bazėje.

Ontologijoje gali apibrėžiami trijų tipų kardinalumo apribojimai: owl:cardinality, owl:minCardinality ir owl:maxCardinality. Kardinalumo tipas owl:cardinality tiksliai apibrėžia, kiek vienos klasės egzempliorius turi turėti ryšių su kitų klasių egzemplioriais. Kardinalumo tipai owl:minCardinality ir owl:maxCardinality atitinkamai apibrėžia apatinę ir viršutinę kardinalumo ribą. Pavyzdys kaip aprašomas kardinalumo apribojimas OWL sintaksėje:

```
<owl:Class rdf:ID="Vynas">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#turiDerliausMetus"/>
      <owl:cardinality
           rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Algoritmas, įrašantis ontologijos kardinalumo apribojimus į reliacinės duomenų bazės metaduomenų lentelę, pateikiamas 14 pav.



14 pav. Ontologijos kardinalumo apribojimų transformavimo į RDB algoritmas

Ontologijos kardinalumo apribojimų transformavimo į reliacinę duomenų bazę metu, visos ontologijos savybės nagrinėjamos nuosekliai. Aptikus kardinalumo apribojimą, į „Cardinality“ metaduomenų lentelę įrašoma apribojamas ryšys, t.y. ryšio klasių lentelių pavadinimai. Taip pat įrašomos minCardinality, maxCardinality ir cardinality reikšmės. Jeigu „Cardinality“ lentelė dar nėra sukurta, ji sukuriama.

4.4.2 Ontologijos owl:allValuesFrom apribojimų transformavimo į RDB algoritmas

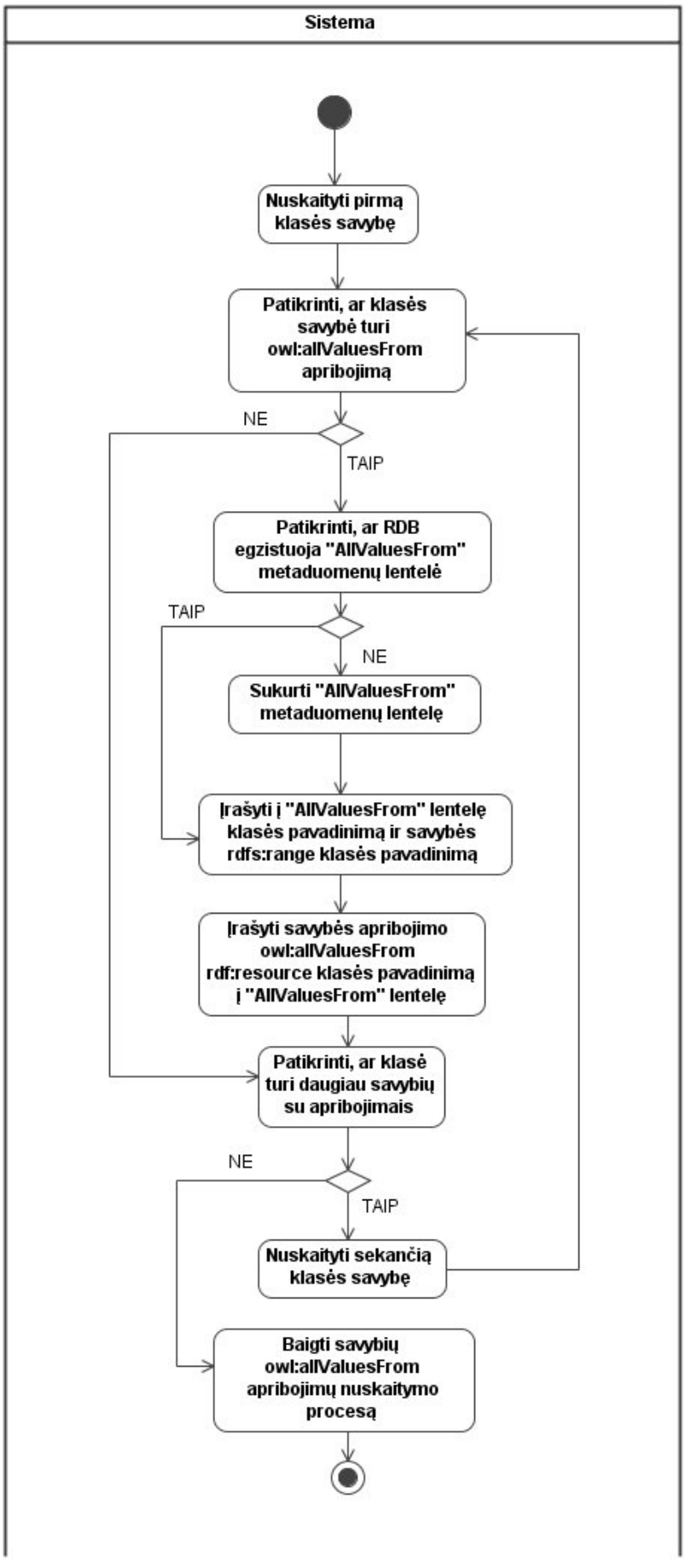
Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, ontologijos klasių savybių apribojimų transformavimo etape po kardinalumo apribojimų transformacijos atliekamas owl:allValuesFrom apribojimų išsaugojimas reliacinėje duomenų bazėje.

Ontologijoje owl:allValuesFrom apribojimas reikalauja, kad visi klasės egzemplioriai, kurie turi tam tikros savybės egzempliorius, turėtų tik apribojime nurodytos klasės savybės egzempliorius. Pavyzdys, kaip aprašomas owl:allValuesFrom apribojimas OWL sintaksėje:

```
<owl:Class rdf:ID="Vynas">
  <rdfs:subClassOf rdf:resource="&maistas;GeriamasSystis" />
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#turiGamintoja" />
      <owl:allValuesFrom rdf:resource="#Vyndarys" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Ontologijoje aprašyto vyno gamintojas turi būti vyndarys. Apribojimas taikomas vyno klasei lokaliai, pavyzdžiui, sūrio gamintojas nebūtinai turi būti vyndarys.

Algoritmas, įrašantis ontologijos owl:allValuesFrom apribojimus į reliacinės duomenų bazės metaduomenų lentelę, pateikiamas 15 pav.



15 pav. Ontologijos owl:allValuesFrom apribojimų transformavimo į RDB algoritmas

Ontologijos owl:allValuesFrom apribojimų transformavimo į reliacinę duomenų bazę metu visos ontologijos savybės nagrinėjamos nuosekliai. Aptikus owl:allValuesFrom apribojimą, į „AllValuesFrom“ metaduomenų lentelę įrašoma apribojamas ryšys, t.y. ryšio klasių lentelių pavadinimai. Taip pat įrašoma apribojančios poklasės lentelės pavadinimas. Jeigu „AllValuesFrom“ lentelė dar nėra sukurta, ji sukuriama.

4.4.3 Ontologijos owl:someValuesFrom apribojimų transformavimo į RDB algoritmas

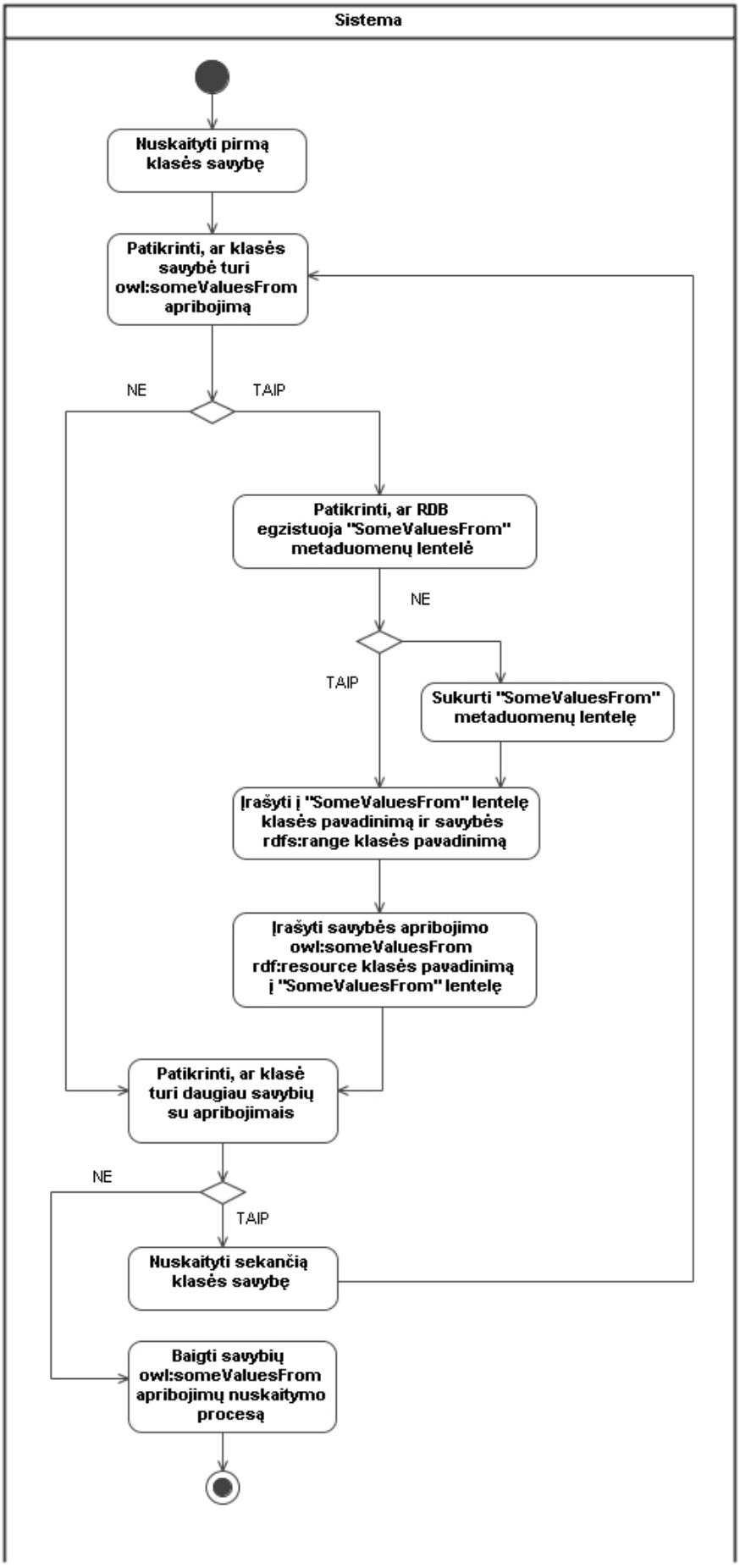
Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, ontologijos klasių savybių apribojimų transformavimo etape po owl:allValuesFrom apribojimų transformacijos atliekamas owl:someValuesFrom apribojimų išsaugojimas reliacinėje duomenų bazėje.

Ontologijoje owl:someValuesFrom apribojimas reikalauja, kad visi klasės egzemplioriai turėtų bent vieną tam tikros savybės egzempliorių iš apribojime nurodytos klasės. Pavyzdys, kaip aprašomas owl:someValuesFrom apribojimas OWL sintaksėje:

```
<owl:Class rdf:ID="Vynas">
  <rdfs:subClassOf rdf:resource="&maistas;GeriamasSkystis" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#turiGamintoją" />
      <owl:someValuesFrom rdf:resource="#Vyndarys" />
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
</owl:Class>
```

Ontologijoje aprašyto vyno bent vienas gamintojas turi būti vyndarys. Apribojimas taikomas vyno klasei lokaliai, pvz. sūrio klasės gamintojas nebūtinai turi turėti bent vieną gamintoją vyndarį.

Algoritmas, įrašantis ontologijos owl:someValuesFrom apribojimus į reliacinės duomenų bazės metaduomenų lentelę, pateikiamas 16 pav.



16 pav. Ontologijos owl:SomeValuesFrom apribojimų transformavimo į RDB algoritmas

Ontologijos owl:someValuesFrom apribojimų transformavimo į reliacinę duomenų bazę metu, visos ontologijos savybės nagrinėjamos nuosekliai. Aptikus owl:someValuesFrom apribojimą, į „SomeValuesFrom“ metaduomenų lentelę įrašoma apribojamas ryšys, t.y. ryšio klasių lentelių pavadinimai. Taip pat įrašoma apribojančios poklasės lentelės pavadinimas. Jeigu „SomeValuesFrom“ lentelė dar nėra sukurta, ji sukuriama.

4.4.4 Ontologijos owl:HasValue apribojimų transformavimo į RDB algoritmas

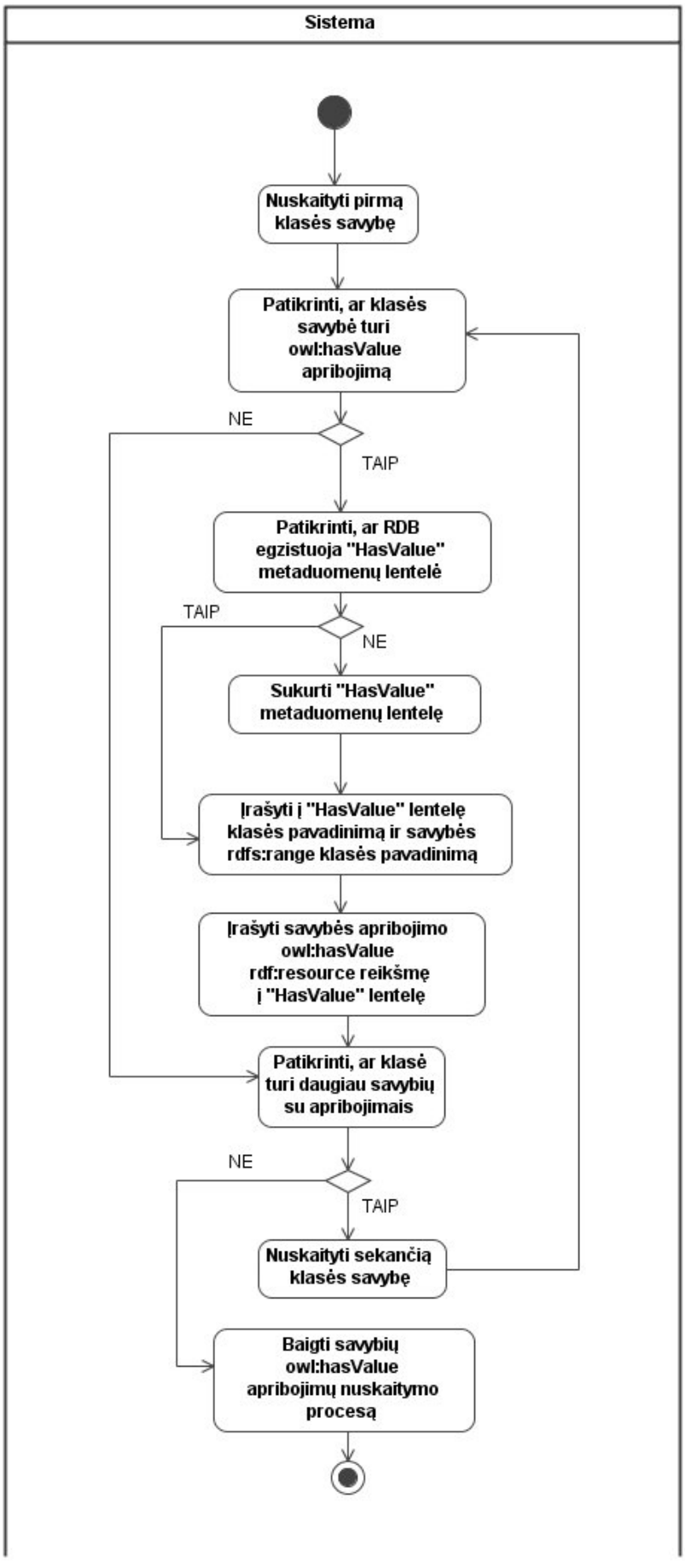
Atliekant produkto konfigūracijos ontologijos transformavimą į reliacinę duomenų bazę, ontologijos klasių savybių apribojimų transformavimo etape po owl:someValuesFrom apribojimų transformacijos atliekamas owl:hasValue apribojimų išsaugojimas reliacinėje duomenų bazėje.

Ontologijoje owl:hasValue apribojimas reikalauja, kad visi klasės egzemplioriai turėtų bent vieną tam tikros savybės egzempliorių, kurio reikšmė būtų lygi apribojime nurodytai reikšmei. Pavyzdys, kaip aprašomas owl:hasValue apribojimas OWL sintaksėje:

```
<owl:Class rdf:ID="Burgundiskas">
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#cukringumas" />
      <owl:hasValue rdf:resource="#Sausas" />
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Ontologijoje aprašyto vyno rūšis „burgundiškas“ turi turėti savybę „cukringumas“ lygią reikšmei „sausas“. Kaip ir kiti apribojimai, šis apribojimas taikomas klasei lokaliai.

Algoritmas, įrašantis ontologijos owl:hasValue apribojimus į reliacinės duomenų bazės metaduomenų lentelę, pateikiamas 17 pav.



17 pav. Ontologijos owl:hasValue apribojimų transformavimo į RDB algoritmas

Ontologijos owl:hasValue apribojimų transformavimo į reliacinę duomenų bazę metu visos ontologijos savybės nagrinėjamos nuosekliai. Aptikus owl:hasValue apribojimą, į „HasValue“ metaduomenų lentelę įrašoma apribojamas ryšys, t.y. ryšio klasių lentelių pavadinimai. Taip pat įrašoma apribojanti reikšmė. Jeigu „HasValue“ lentelė dar nėra sukurta, ji sukuriama.

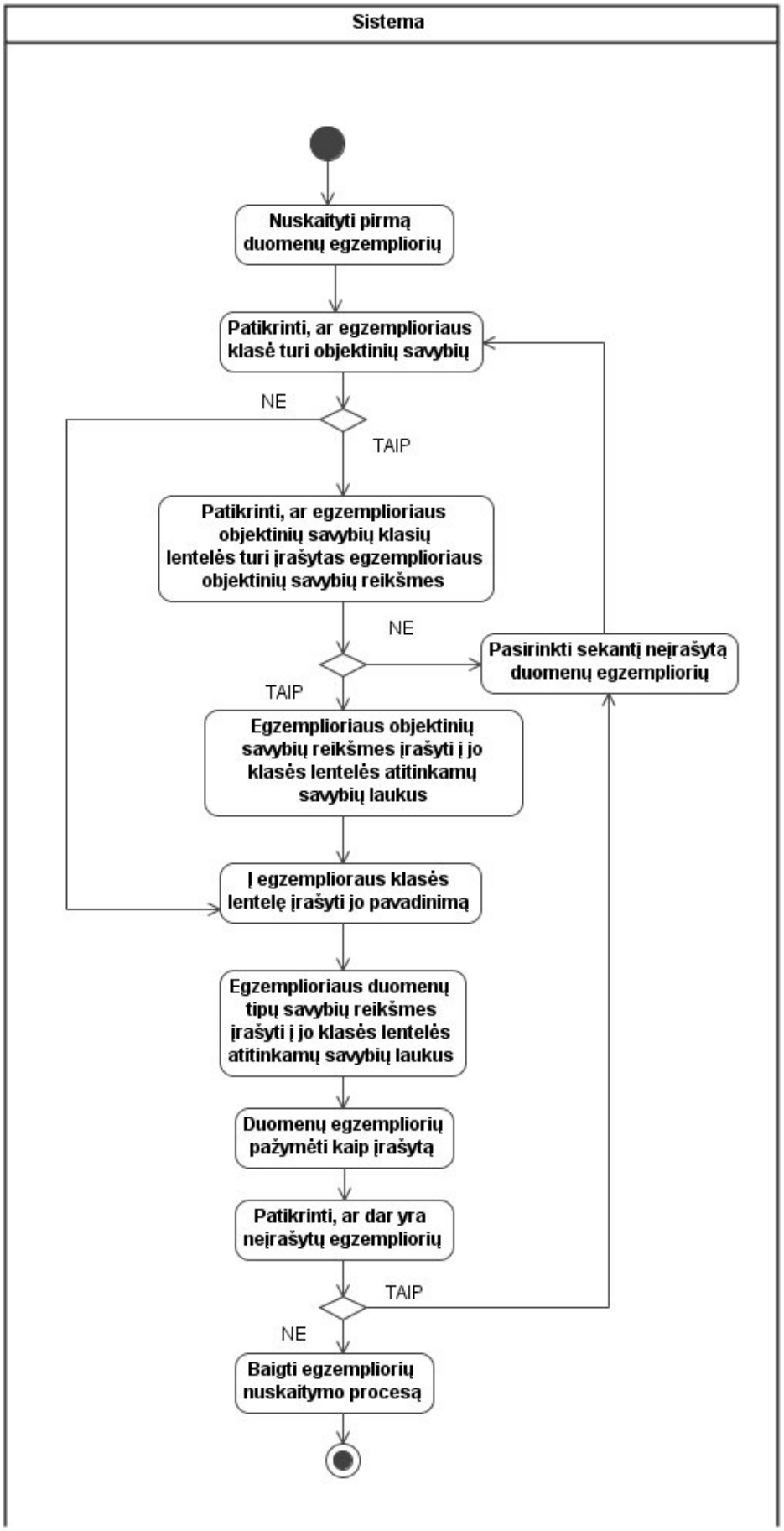
4.5 Ontologijos duomenų egzempliorių įrašymo į RDB algoritmas

Paskutiniu produkto konfigūracijos ontologijos transformavimo į reliacinę duomenų bazę etapu ontologijoje esantys egzemplioriai įrašomi į suformuotą reliacinę duomenų bazę.

Ontologijoje egzemplioriais vadinami klases sudarantys nariai. Egzemplioriai aprašomi nurodant, kokiai klasei jie priklauso. Taip pat gali būti nurodomos egzempliorių savybių reikšmės. Pavyzdys, kaip gali būti aprašomas egzempliorius OWL sintaksėje:

```
<Automobilis rdf:ID="Mazda626">
  <pagamMetai rdf:datatype="&xsd;positiveInteger">1996</pagamMetai>
  <gamintojas   rdf:resource="#MAZDA"/>
</Automobilis>
```

Algoritmas, įrašantis ontologijos egzempliorių įrašymą į reliacinę duomenų bazę, pateikiamas 18 pav.



18 pav. Ontologijos duomenų egzempliorių įrašymo į RDB algoritmas

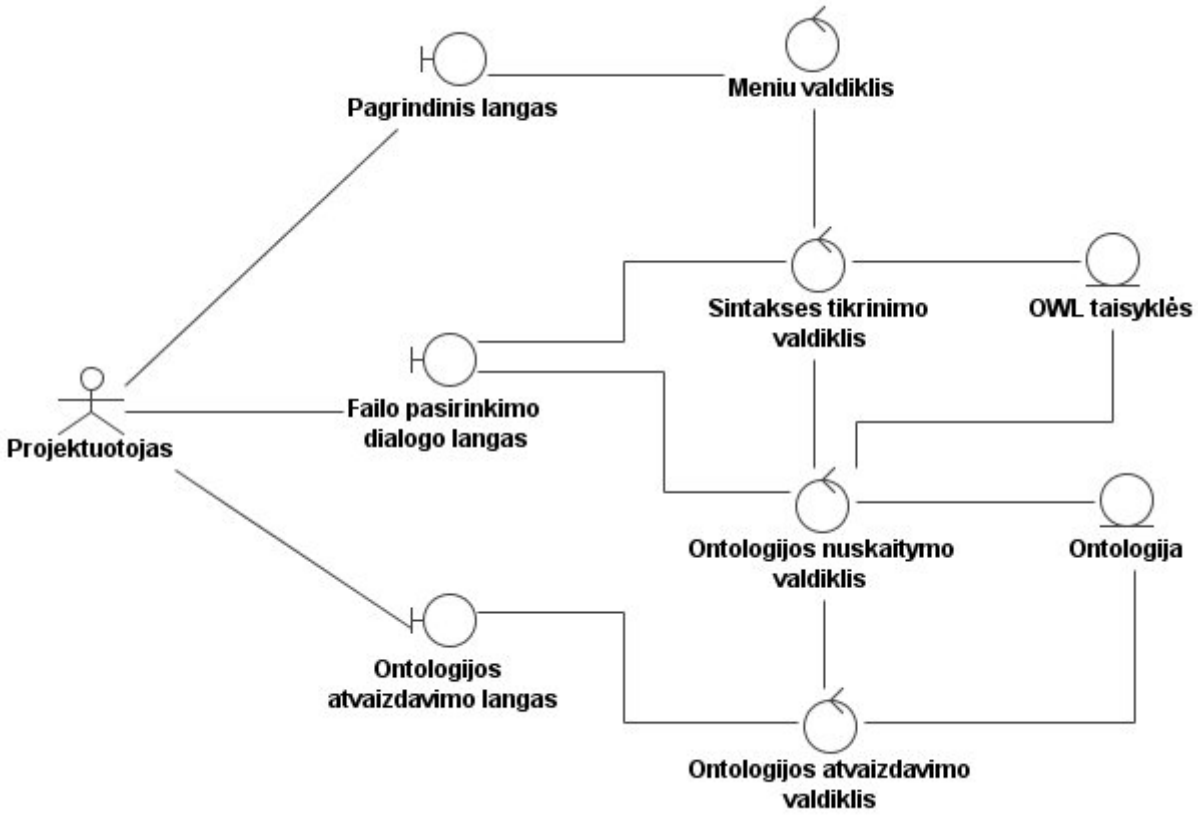
Ontologijos egzempliorių įrašymo į reliacinę duomenų bazę metu visi ontologijos egzemplioriai nagrinėjami nuosekliai. Kadangi duomenų įrašymas į reliacinė duomenų bazę negali pažeisti vaiko-tėvo lentelių ryšių kardinalumo, tikrinama, ar nagrinėjamas egzempliorius turi objektinių savybių. Jeigu turi, tuomet tikrinama, ar į tėvines lenteles (objektinių savybių srities klases) yra įrašytos reikšmės. Jeigu ne, tuomet egzemplioriaus įrašymas praleidžiamas. Jeigu taip, tuomet egzemplioriaus reikšmės įrašomos į lentelę, o pats egzempliorius pažymimas kaip įrašytas. Imamas sekantis neįrašytas egzempliorius ir ciklas kartojamas tol, kol nebelieka neįrašytų egzempliorių.

5. Ontologijos aprašo transformavimo įrankio programinės realizacijos projektas

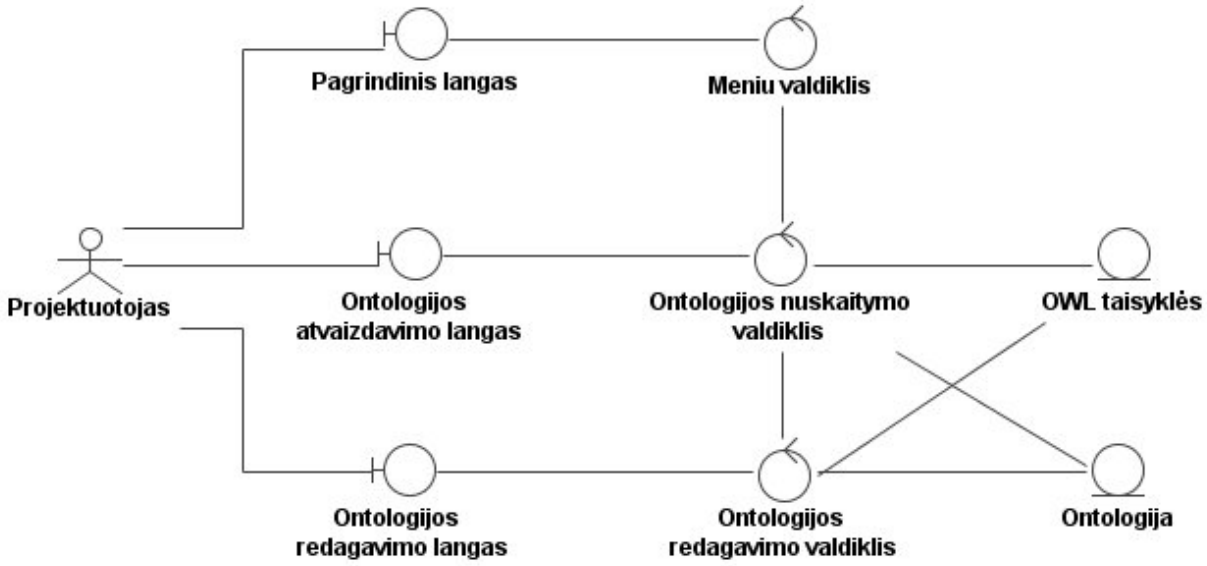
5.1 Panaudojimo atvejų analizės ir realizacijos klasės

Analizės klasių diagramos sudaromos tam, kad nustatyti sistemos vartotojų veiksmus ir susieti juos su vidinėmis sistemos saugyklomis bei valdikliais. Turint šias klasių diagramas, galima kurti projekto klasių modelį.

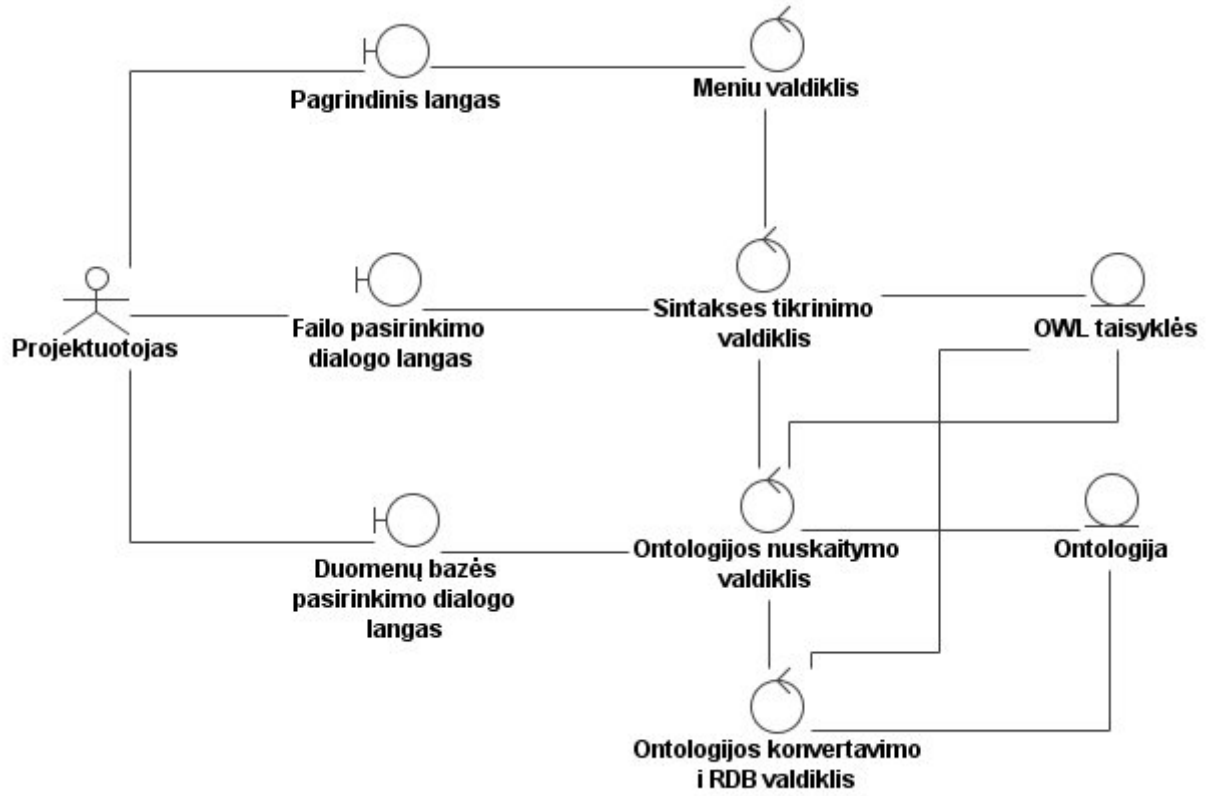
Analizės klasių modeliai pateikiami 19 pav.20 pav.ir 21 pav. Panaudojimo atvejų realizacijos modelis pavaizduotas 22 pav.



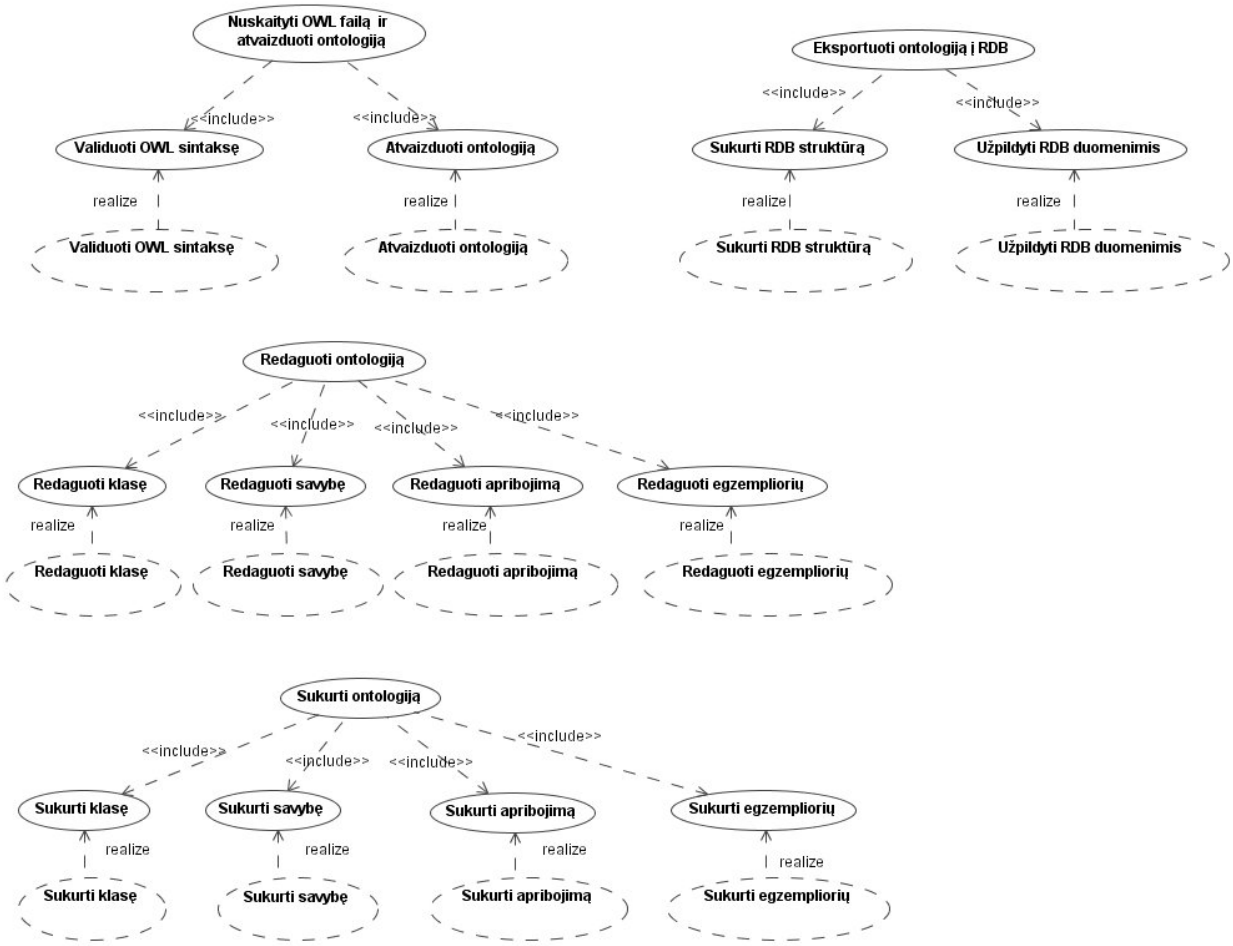
19 pav. Ontologijos nuskaitymo ir atvaizdavimo analizės klasių modelis



20 pav. Ontologijos redagavimo analizės klasių modelis



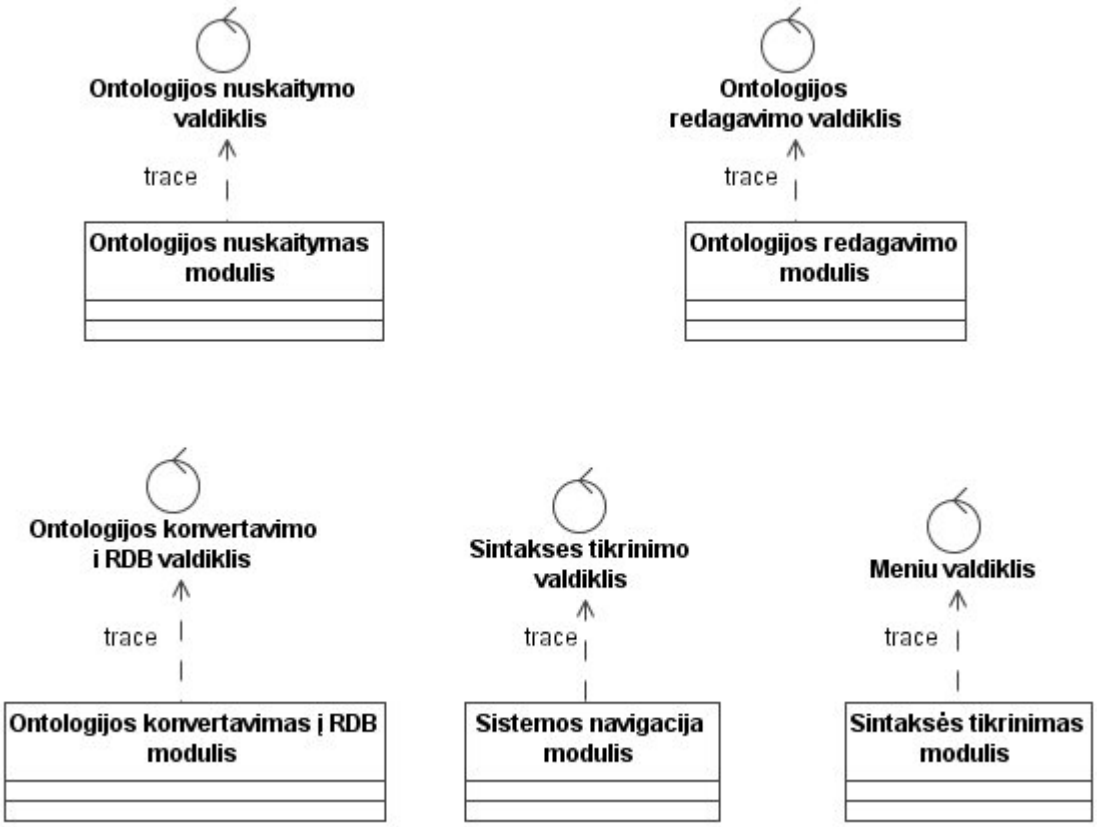
21 pav. Ontologijos konvertavimo į RDB klasių modelis



22 pav. Panaudojimo atvejų realizacijos

5.2 Valdymo klasių modelis

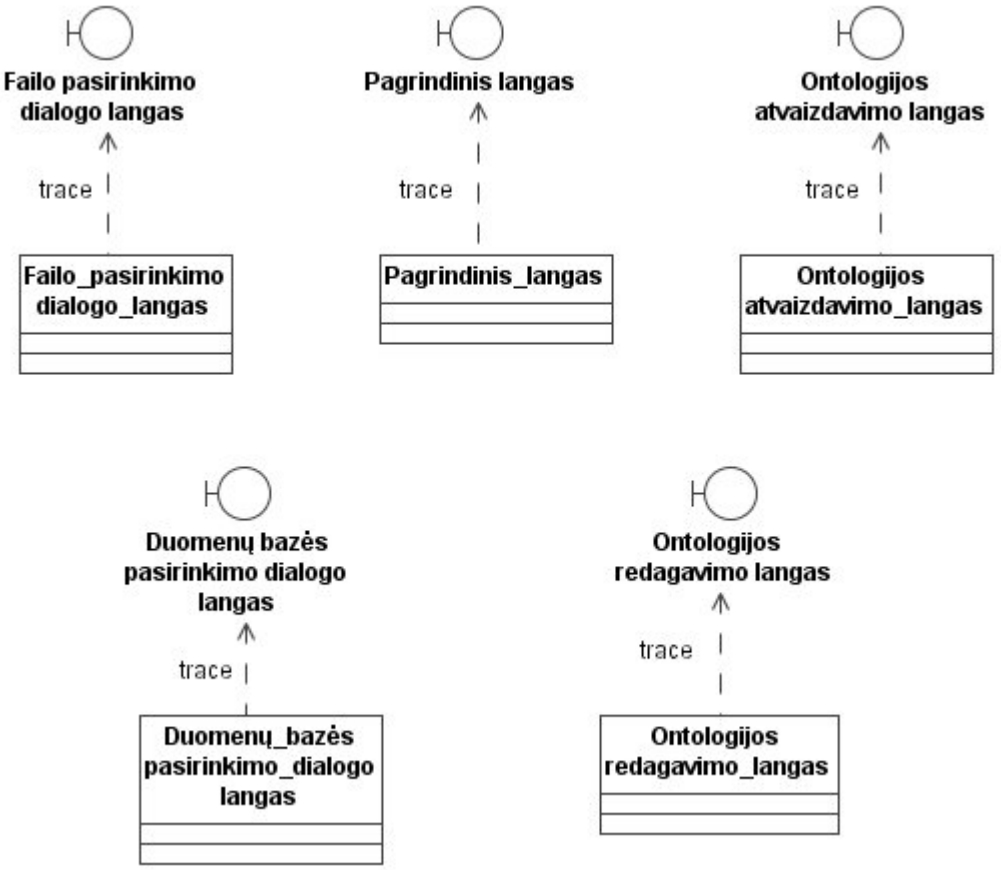
Sistemos projekto modelyje sukuriamos valdymo klasės. Valdymo klasės gaunamos naudojantis analizės klasėmis – valdikliais. Trasų diagrama atspindi ryšį tarp analizės ir projektavimo etapo valdymo klasių (23 pav.).



23 pav. Valdymo klasės (moduliai)

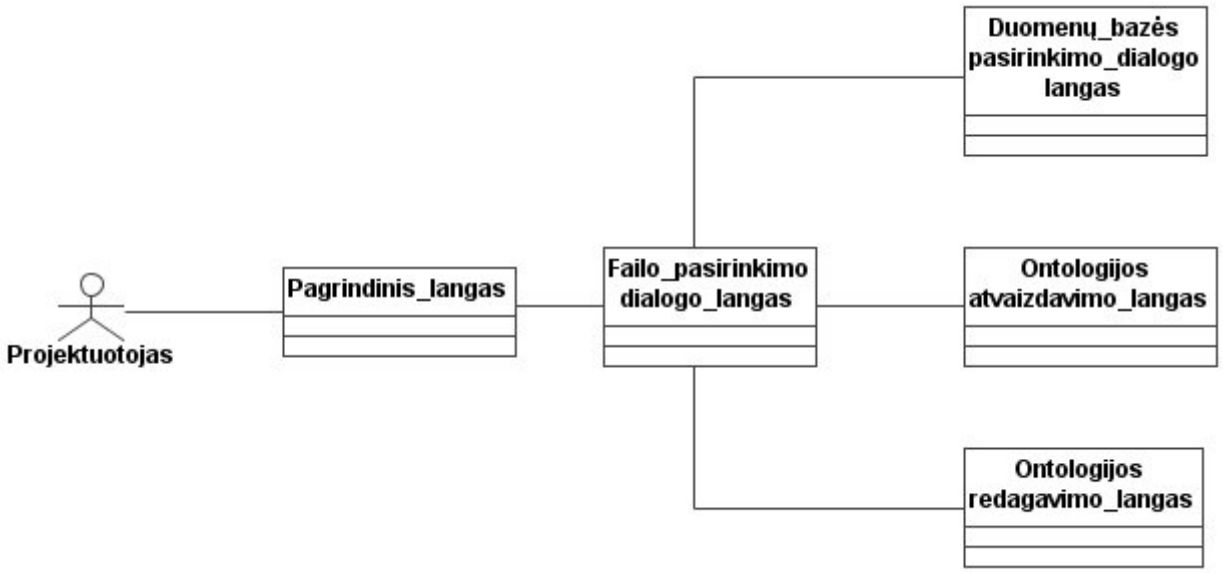
5.3 Vartotojo sąsajos modelis

Sistemos projekto modelyje sukuriamos vartotojo sąsajos klasės. Vartotojo sąsajos klasės gaunamos naudojantis analizės vartotojo sąsajos klasėmis. Trasų diagrama atspindi ryšį tarp analizės ir projektavimo etapo vartotojo sąsajos klasių 24 pav.



24 pav. Vartotojo sąsajos klasės

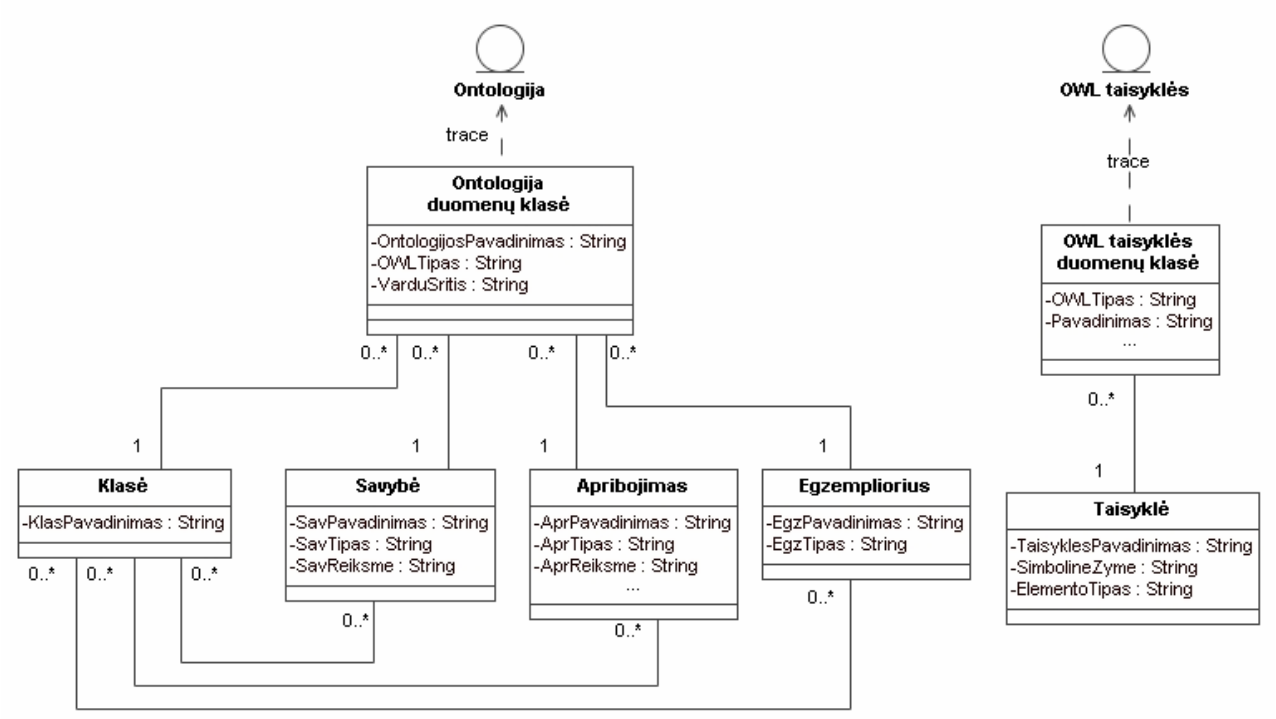
Vartotojo sąsajos modelis (navigacijos planas) kuriamas atsižvelgiant į panaudojimo atvejų diagramas ir parodo iš kokio sistemos būsenos taško galima patekti į kitą būseną bei kuo tai realizuojama (25 pav.).



25 pav. Vartotojo navigacijos planas

5.4 Duomenų klasių modelis

Duomenų klasių modelis buvo sudarytas naudojantis analizės klasių diagrama. Trasų diagrama (26 pav.) atspindi ryšį tarp analizės ir projekto duomenų klasių.



26 pav. Duomenų klasių modelis

Ontologijos duomenų klasė saugo bendrą informaciją apie pačią ontologiją, tokią kaip jos pavadinimas, kokio tipo OWL kalba naudojama (OWL Lite, OWL DL ar OWL Full), taip pat vardų sritį (angl. *namespace*). Ontologija sudaryta iš klasių, savybių, apribojimų ir egzempliorių, kuriems sukurtos atitinkamos duomenų klasės. Klasė gali turėti keletą savybių, taip pat tą pačią savybę gali turėti ir keletas klasių. Pagal OWL standartą, taip pat yra ir su apribojimais bei egzemplioriais. Savybės ir apribojimai, be pavadinimo, dar turi tipą (pvz. duomenų tipų savybė arba kuri nors objektinė savybė, kardinalumo arba „AllValuesFrom“ apribojimas) bei reikšmes.

OWL sintaksės taisyklių informacijai saugoti sukurta duomenų klasė turi savo pavadinimą bei OWL tipą (pvz. OWL Lite ar kiti). Pagal konkretų OWL tipą sužinoma, kokias sintaksės taisykles naudoti. Konkrečios taisyklės duomenų klasėje saugoma OWL žymės atpažinimo aprašymas, bei kokius elementus galima naudoti šios žymės viduje.

6. Transformacijos įrankio prototipo realizacija

6.1 Transformacijos įrankio realizacijos platforma

Transformacijos įrankiui realizuoti buvo pasirinkta Java2SE platforma. Pagrindinės pasirinkimo priežastys yra šios: Java kalbos paprastumas, platformos portabilumas, saugumas, patikimumas. Taip pat naudojant Java programavimo kalbą buvo galima praplėsti egzistuojantį atviro kodo Protege modeliavimo įrankį, skirtą darbui su ontologijomis. Java technologijos palyginimas su viena iš populiariausių C#.NET programavimo aplinkų [x1] pateikiamas 4 lentelėje.

Lentelė 4. Java ir C#.NET charakteristikų palyginimas

Savybė	Java	C#.NET
Daugiaplatformiškumas	+	-
Greitaveika	+-	+
Greitas taikomųjų programų kūrimas	-	+
Programavimo kalbos sintaksės paprastumas	+	+
Didelė palaikymo bendruomenė, daug atviro kodo programų	+	-
Didelis bibliotekų (API) pasirinkimas	+	-
Stabilumas, saugumas	+	+
Patogūs IDE redaktoriai	+	+

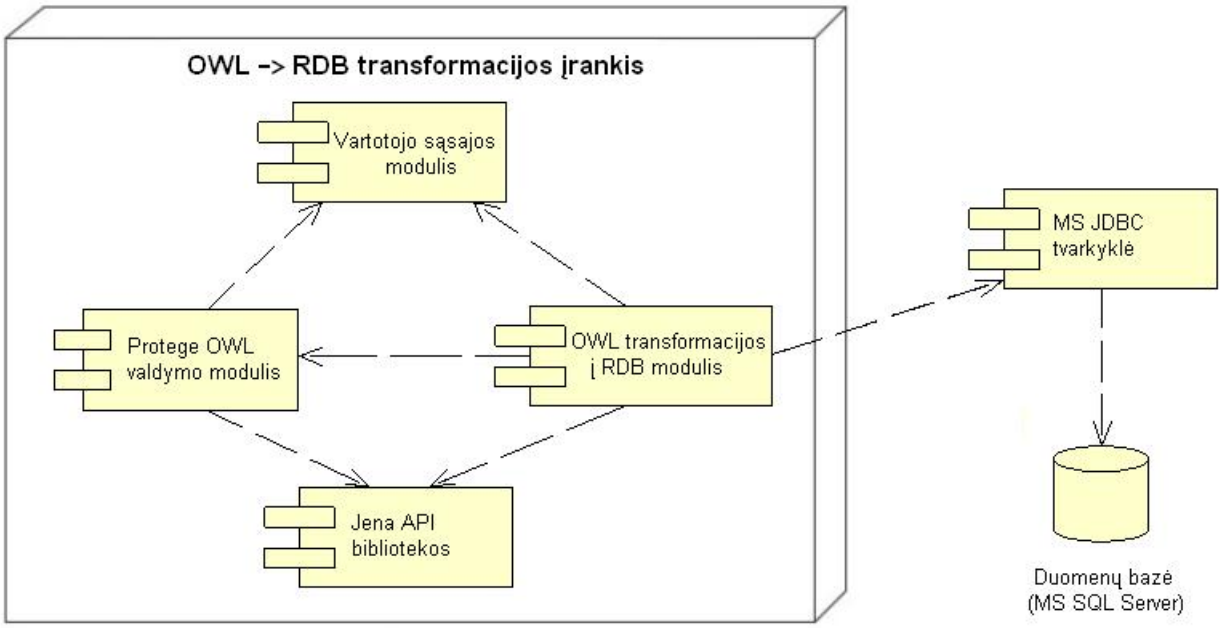
6.2 Transformacijos įrankio architektūra

Realizuotas transformacijos įrankis susideda iš vartotojo sąsajos modulio, „Protege“ OWL ontologijų valdymo modulio, Jena API (universali programavimo terpė) ir OWL transformacijų į RDB modulio. Vartotojo sąsajos modulis skirtas palaikyti ryšį tarp projektuotojo ir sistemos, leidžia atlikti tokius veiksmus, kaip pasirinkti ontologijos aprašo failą, atvaizduoti ontologiją grafiškai, pasirinkti duomenų bazių serverį ir iššaukti kitas reikalingas komandas. „Protege“ OWL valdymo modulis atlieka ontologijos aprašo sintaksės validavimą, formuoja ontologijos objektus, pateikia įvairius ontologijos valdymo veiksmus. OWL transformacijos į RDB modulis nagrinėja ontologijos objektus, vykdo OWL transformaciją į reliacinės duomenų bazės schemą, formuoja bei įvykdo SQL užklausas, užtikrina sąsają tarp sistemos ir duomenų bazių serverio.

Tam, kad paprasčiau atlikti veiksmus su ontologijos objektais Protege OWL ir transformacijos moduliai naudoja Jena API bibliotekas. Sistema naudoja Microsoft JDBC

tvarkyklę, kuri leidžia operuoti su Microsoft SQL serveriu standartiniu būdu, naudojant SQL užklausas. Transformacijos įrankis yra atskirtas nuo MS JDBC modulio, kuris veikia nepriklausomai. Naudojant vartotojo sąsają galima pasirinkti norimą JDBC tvarkyklę, priklausomai nuo naudojamo duomenų bazių serverio, kuriame ruošiamasi išsaugoti sukurtą duomenų bazę. Eksperimiantiniam tyrimui pasirinkta viena iš plačiausiai naudojamų komercinių duomenų bazių valdymo sistemų MS SQL Server 2005.

Komponentų diagrama (27 pav.) rodo fizinį sistemos vaizdą: komponentus bei jų tarpusavio priklausomybes.

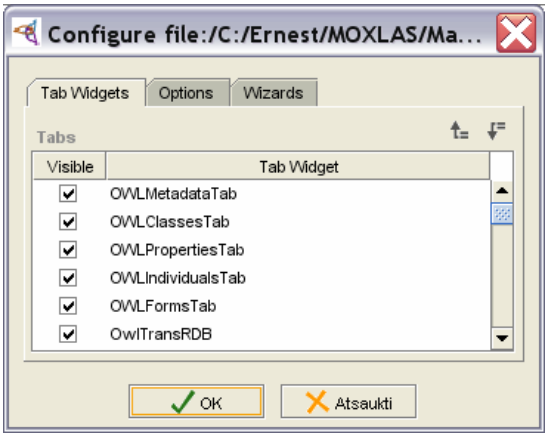


27 pav. Transformacijos įrankio komponentų modelis

6.3 Transformacijos įrankio realizacijos aprašymas

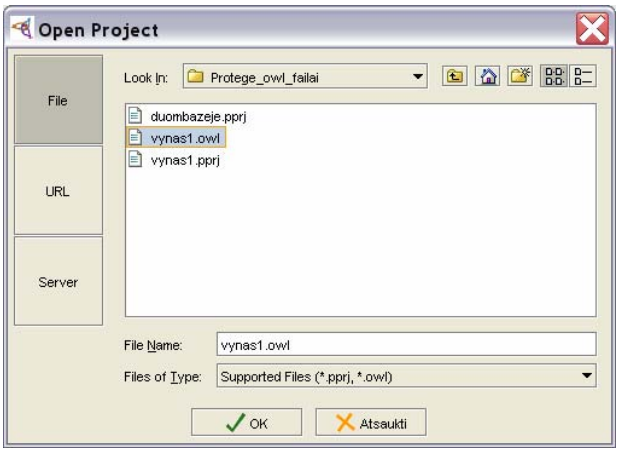
Pasirinkta realizacijos platforma leidžia kiekvienam vartotojui nesudėtingai integruoti OWL transformacijos įrankį į ontologijų pasaulyje plačiausiai naudojamą „*Protege*“ sistemą. Programos bibliotekų rinkinys „owl_trans_rdb.jar“ turi būti įkeliamas į „Protege/plugins“ direktoriją. Paleidus „*Protege*“ sistemą, pagrindiniame meniu pasirenkamas punktas „Projektas“ ir spaudžiamas mygtukas „Konfigūruoti...“. Atsidariusiame lange pateikiamas visų sistemos matomų ir nematomų modulių sąrašas (28 pav.). Skirtingi moduliai parenkami priklausomai nuo to, su kokio tipo ontologijomis ar jų sritimis dirbama. Sukurtas OWL ontologijų transformacijos

į reliacines duomenų bazes modulis pavadintas „OWLTransRDB“. Jį pasirinkus, sistemos aktyvių posistemių juostoje susikuria naujas modulis, kuris ir bus naudojamas atlikti transformacijoms.



28 pav. „Protege“ sistemos modulių konfigūracija

Bendras darbo su sistema procesas prasideda ontologijos OWL aprašo įkėlimu į sistemą. Tam atlikti pagrindiniame meniu pasirenkame punktą „Failai->Atidaryti“. Atsivėrusiame failų paieškos lange susirandame ontologijos aprašo failą .owl ir spaudžiame mygtuką „OK“ (29 pav.). Ontologijos modelio informacija užkraunama į operatyviąją atmintį. Šiuos veiksmus atlieka „Protege“ sistemos standartiniai moduliai. Kartu patikrinama ir OWL sintaksės bei failo struktūros korektiškumas, taigi dirbama tik su sintaksiškai tvarkingu ontologijos aprašu.

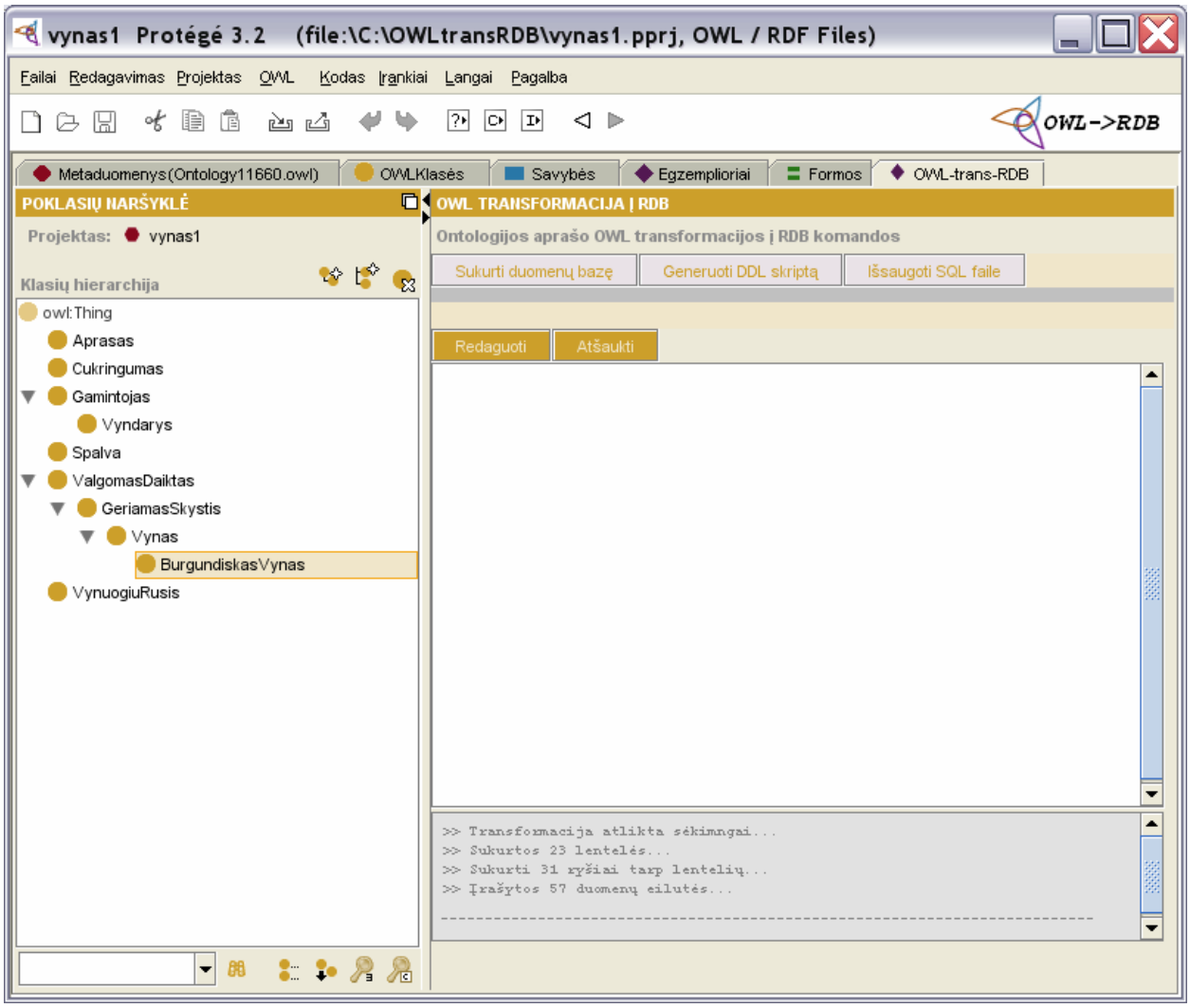


29 pav. OWL aprašo failo pasirinkimas

Sistemoje įkeltos ontologijos turinį galima peržiūrėti ir redaguoti naudojant „Protege.OWL“ modulius. Tai „OWLKlasės“ modulis, skirtas darbui su klasėmis, „Savybės“ ir „Egzemplioriai“ moduliai, skirti savybių ir klasių egzempliorių peržiūrai, redagavimui arba

kūrimui. Taip pat yra ir kitų modulių, skirtų darbui su ontologijomis, tačiau, kadangi jie visi neįeina į šio darbo apimtį, smulkiau jų nedetalizuosime.

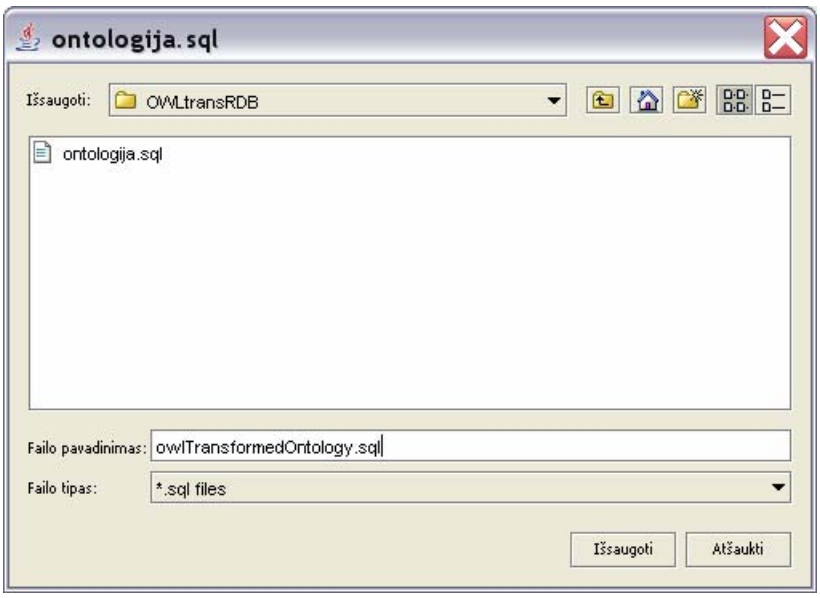
Pagrindinis transformacijos į RDB įrankio langas pateikiamas 30 pav. paveiksle. Jis susideda iš modelio poklasių naršyklės, transformacijos komandų bei SQL peržiūros lango.



30 pav. OWL transformacijos į RDB sistemos pagrindinis langas

Poklasių naršyklės srityje pateikiama visos ontologijos modelio klasės. Klasės yra išdėstytos hierarchiškai, pagal tėvo-vaiko ryšius, kiekvieną hierarchinį lygmenį galima išskleisti arba sutraukti, paliekant tik tėvinę klasę.

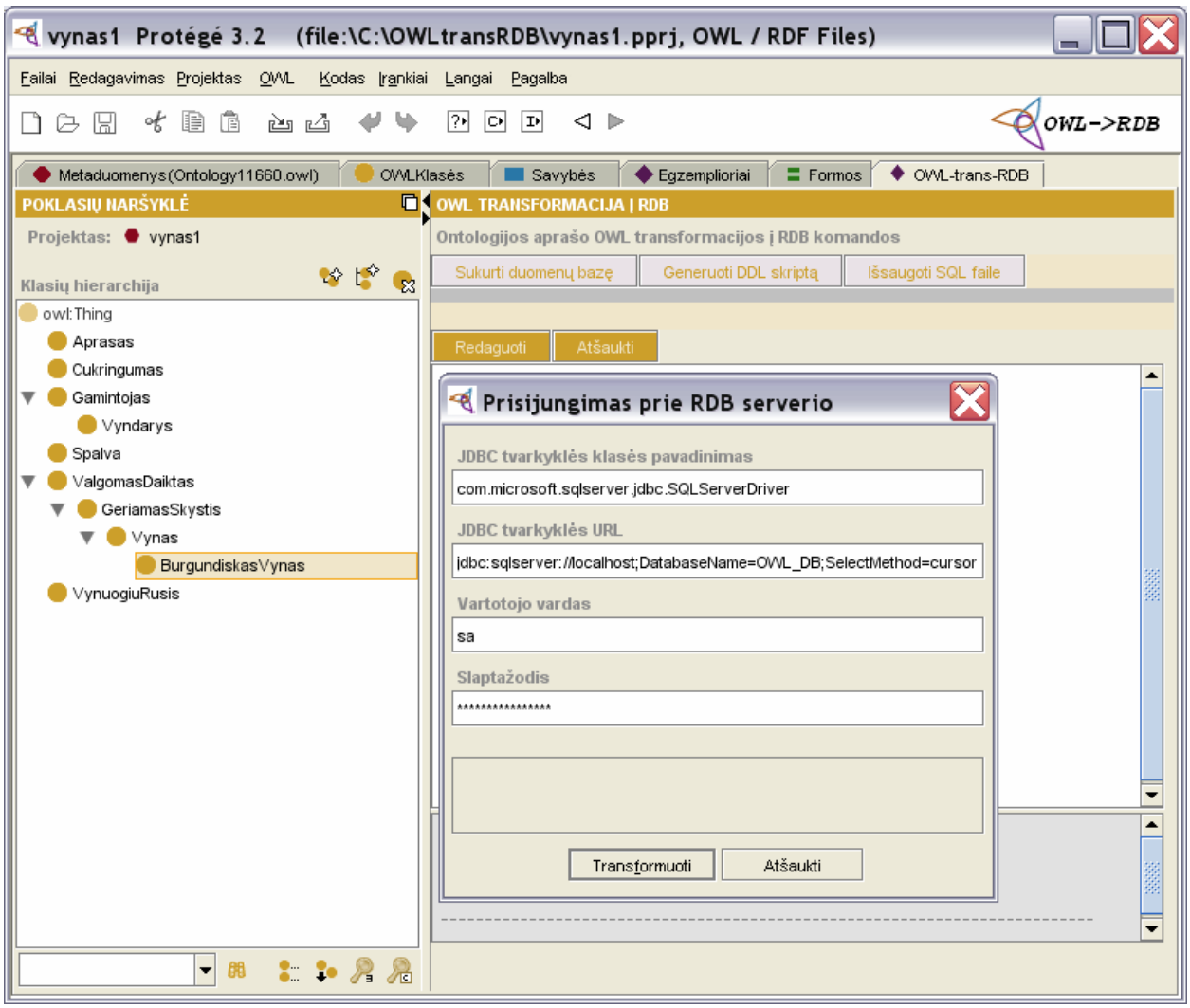
Kitame sistemos lange „OWL transformacija į RDB“ pateikiamos transformacijos komandos. Pirma komanda „Sukurti duomenu bazę“ atidaro langą, skirtą atlikti transformacijai tiesiogiai į reliacinių duomenų bazių serverį, sukuriant duomenų bazę. Komanda „Generuoti DLL skriptą“ atlieka OWL aprašo transformaciją į SQL kodą, ir jį pateikia redagavimo lange (31 pav.). Paskutinioji komanda „Išsaugoti SQL faile“ atidaro langą, kuriame nurodžius failą, atlikta transformacija išsaugoma kaip .sql tipo failas.



31 pav. Transformacijos SQL failo išsaugojimas

Po komandų sąrašo seka transformacijos peržiūros ir redagavimo langas. Poklasių naršyklėje klasių hierarchijos medyje pasirinkus tam tikrą klasė, jos transformacijos SQL kodas pateikiamas peržiūros lange. Transformuojama pati klasė, jos turimos savybės, apribojimai bei klasės egzemplioriai. Transformacijos kodo lango viršuje yra mygtukas „Redaguoti“, kurį paspaudus galima atlikti SQL kodo redagavimą ir vėliau išsaugoti .sql faile pataisytą variantą. Mygtukas „Atšaukti“ vėl parodo transformacijos algoritmų sukurtą SQL kodą.

Norint tiesiogiai prisijungti prie duomenų bazių serverio ir atliktą transformaciją išsaugoti kaip reliacinę duomenų bazę, reikia pasirinkti komandą „Sukurti duomenų bazę“, kuri atveria prisijungimo prie duomenų bazių serverio langą (32 pav.). Ontologijos SQL transformacijoms išsaugoti galima naudoti bet kokią duomenų bazių valdymo sistemą, kuriai yra sukurta JDBC tvarkyklė. Pasirinkta tvarkyklė turėtų būti įkelta į „Protege“ instaliacijos katalogą, tam, kad sistemos paleidimo metu ji būtų prijungta kaip klasių biblioteka ir taptų pasiekiama.

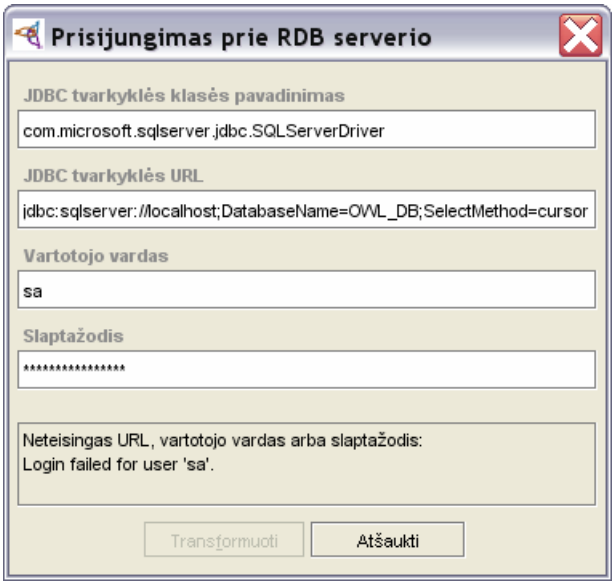


32 pav. Prisijungimas prie RDB serverio

Jungiantis prie duomenų bazių serverio, reikia nurodyti JDBC tvarkyklę, kelią iki serverio ir vartotojo vardą bei slaptažodį. Eksperimento metu buvo naudojama „MS SQL Server 2005“ duomenų bazių valdymo sistema.

Transformaciją galima atlikti tik sėkmingai prisijungus prie duomenų bazių serverio. Įvedant kiekvieną lauką, sistema tikrina jų reikšmes ir vartotojui išveda prasmingus paaiškinimus, kartu pridėdama sisteminius klaidų pranešimus. Tik sėkmingai prisijungus prie duomenų bazės, mygtukas „Transformuoti“ tampa aktyvus, kurį paspaudus atliekami transformacijos veiksmai, t.y. lentelių, ryšių sukūrimas bei duomenų įrašymas.

Pranešimai apie sėkmingai atliktą transformaciją, sukurtus objektus bei įvykusias klaidas pateikiami pagrindiniame transformacijos įrankio lange, sisteminiu pranešimų lauke.



33 pav. Neteisingas prisijungimas prie duomenų bazių serverio

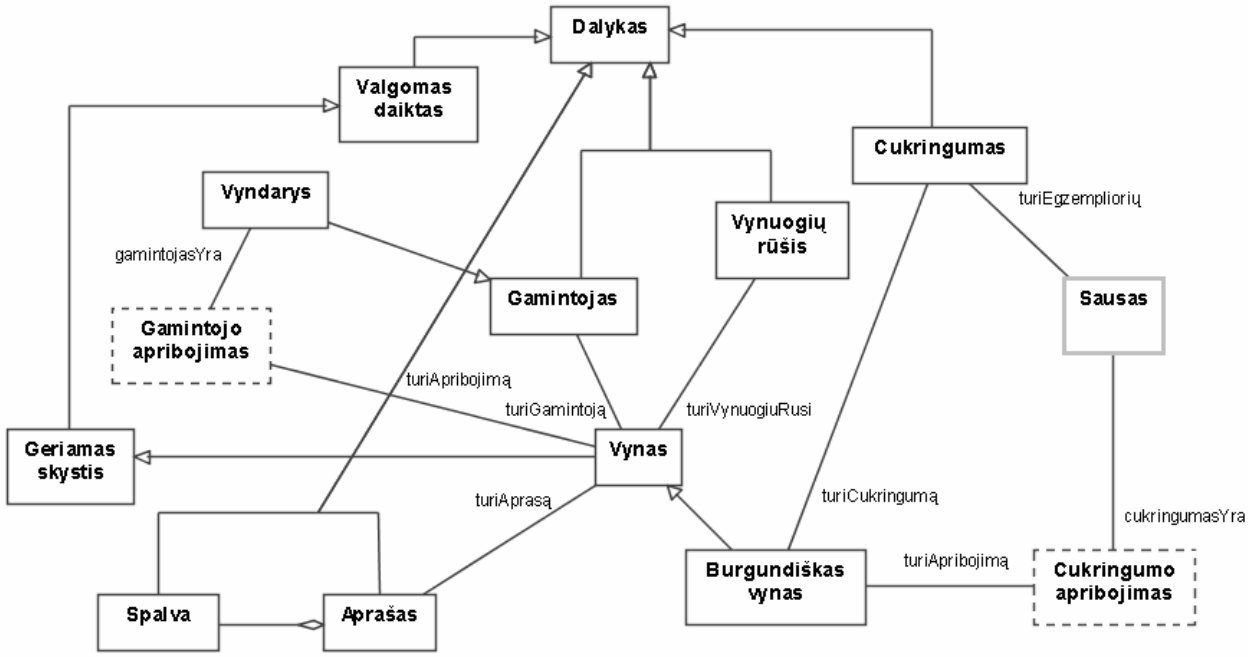
7. Eksperimentinis tyrimas ir darbo rezultatų įvertinimas

7.1 Pavyzdinės dalykinės srities ontologijos kūrimas

Eksperimentiniam tyrimui atlikti buvo sukurta nedidelė ontologija, aprašanti produktą „vynas“. Ontologijų bendruomenėje vyno ontologija dažnai naudojama kaip pavyzdys, kadangi yra nesudėtinga ir kiekvienam suprantama.

Pats vynas, kadangi yra geriamas skystis, nėra konfigūruojamas objektas, tačiau vynas, kaip produktas arba produkto aprašymas, yra pilnai konfigūruojamas, kadangi jis gali turėti daug skirtingų savybių ir charakteristikų, tokių kaip vynuogių rūšis, iš kurių jis yra pagamintas, pagaminimo metai, taip pat apribojimus, kurie apsprendžia jo rūšį, pavydziui cukringumas, sausumas ir daugelį kitų. Galima kurti naujus vynų aprašymus arba atlikti semantinę paiešką jau egzistuojančiuose, taigi šiuo požiūriu vynas yra pilnai konfigūruojamas objektas.

Sudarytas minimalus vyno ontologijos modelis, naudojant konfigūracijos ontologijos vaizdavimo notaciją (2 pav.), pateikiamas 34 pav..



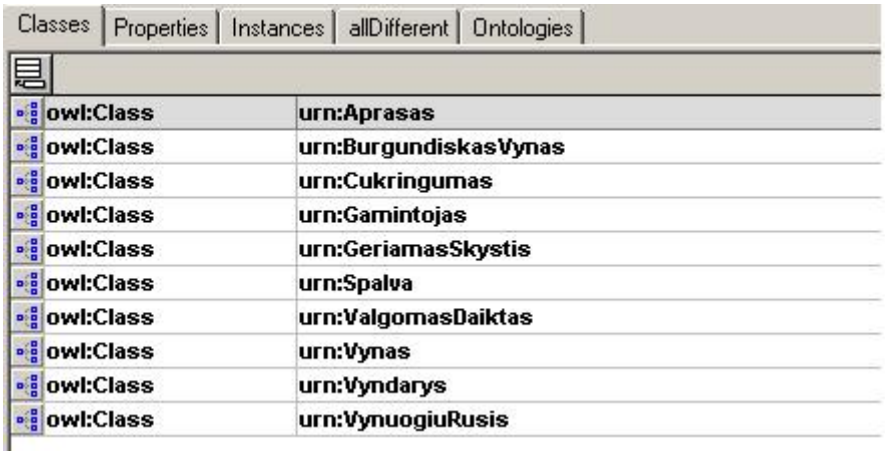
34 pav. Produkto „Vynas“ ontologija

7.2 Ontologijos aprašo OWL konstravimas

Eksperimentui naudojamos vyno ontologijos aprašo konstravimui ir atvaizdavimui buvo pasirinktas *„Altova SemanticWorks 2006“* įrankis. Pasirinkimą nulėmė tai, kad ši programa

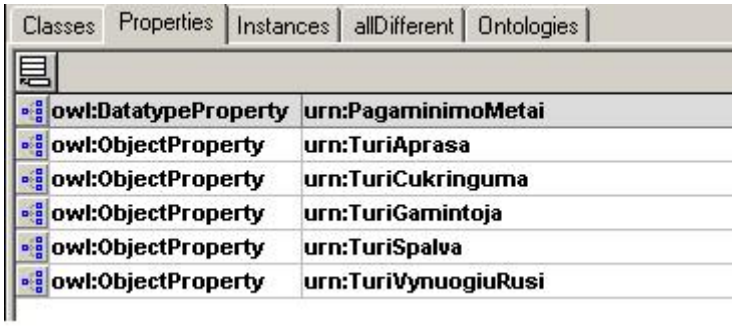
leidžia pakankamai paprastai konstruoti nedideles ontologijas bei suteikia galimybę vizualiai aiškiai ir tiksliai atvaizduoti sukurtas modelio struktūras.

Kuriant ontologiją, pirmiausiai yra aprašomos klasės. Ontologijai „Vynas“ sukurtos klasės pavaizduotos 35 pav.



35 pav. Ontologijos klasės

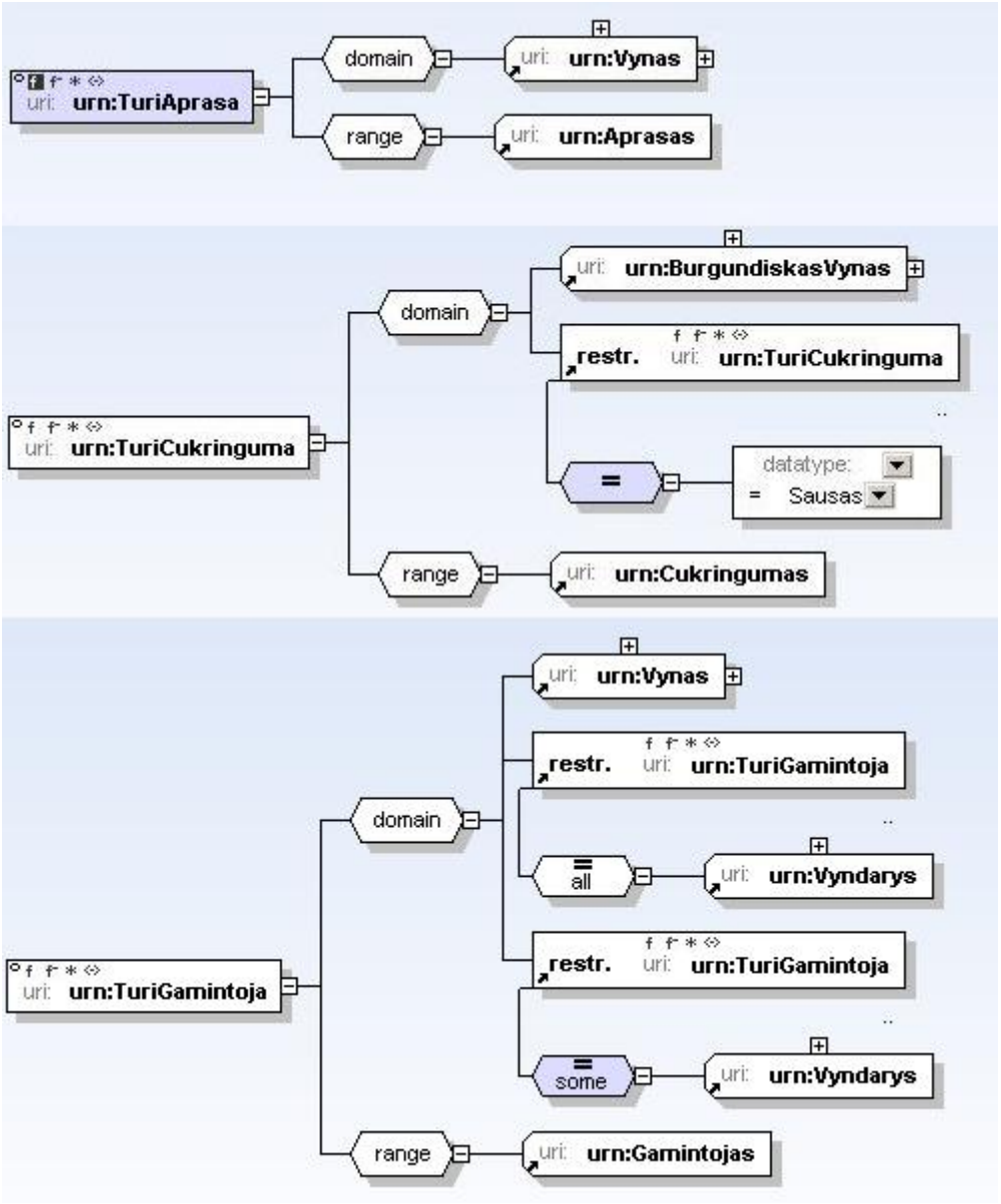
Sukūrus klases, konstruojamos objektinės ir duomenų tipų savybės. Ontologijai „Vynas“ sukurtos savybės pavaizduotos 36 pav. Tai savybės, rodančios kad vynas turi pagaminimo metus, gamintoją ir keletas kitų.



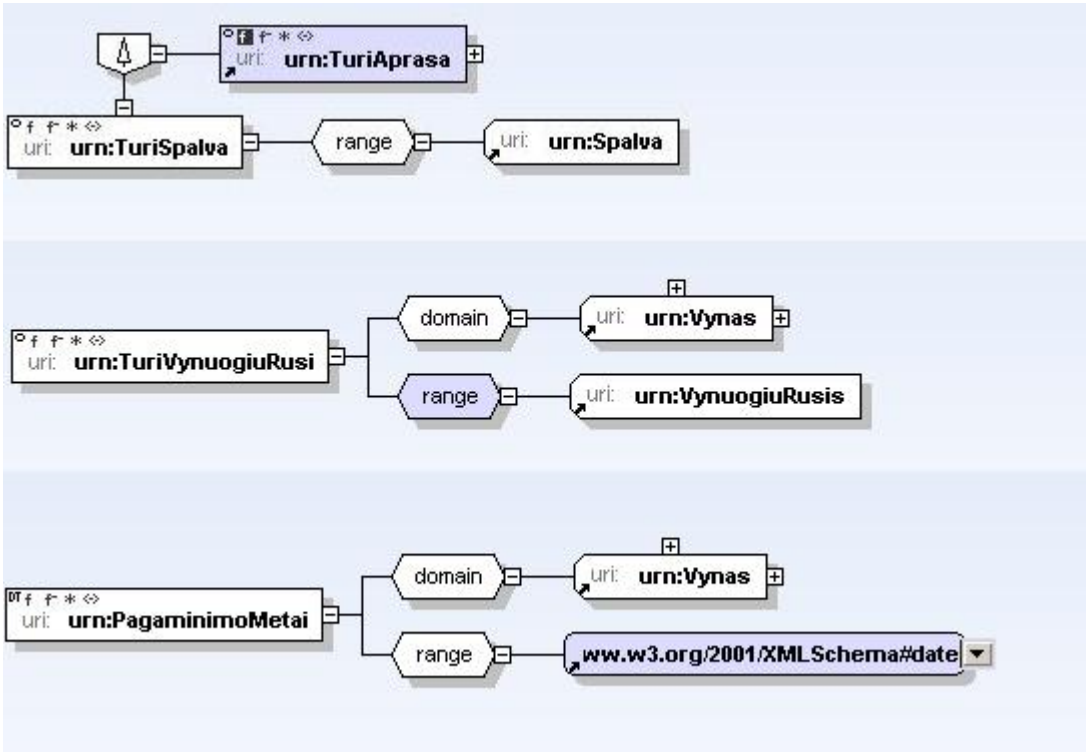
36 pav. Ontologijos savybės

Sekančiame etape kiekvienai savybei nurodomos „domain“ ir „range“ klasės. Taip pat nurodomi klasių savybių apribojimai, tokie kaip „owl:someValuesFrom“, „owl:allValuesFrom“, „owl:hasValue“, kardinalumo ir kiti apribojimai (37 pav., 38 pav.). Pavyzdžiui, vynas yra

burgundiškas tik tuomet, kai tenkinamas apribojimas, kad savybė „cukringumas“ yra lygi „sausam“.

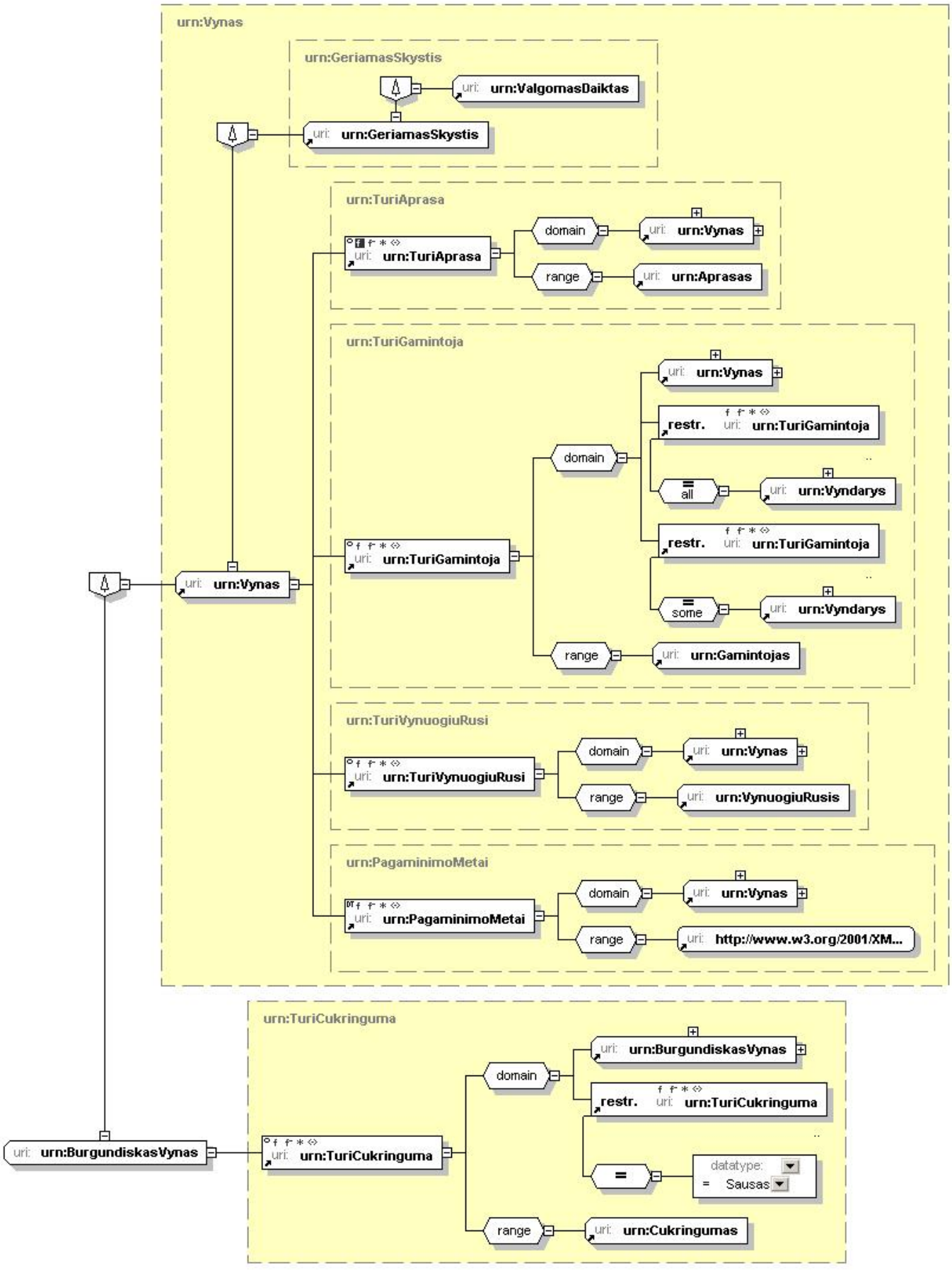


37 pav. Ontologijos savybės I



38 pav. Ontologijos savybės II

Sukūrus klases, nurodžius nesikertančias klases, klasių hierarchinius ryšius, aprašius savybes bei apribojimus, gaunama ontologijos schema, pavaizduota 39 pav. Vertikaliais ryšiais atvaizduojami klasių hierarchiniai ryšiai, horizontaliais – savybės bei apribojimai. Specifinių jungiančiųjų objektų pavadinimai atitinka OWL kalbos notaciją, taigi schema yra lengvai suprantama.



39 pav. Produkto "Vynas" ontologijos OWL aprašas

Sukurta ontologija išsaugoma OWL faile. Ontologijos „Vynas“ OWL failo fragmentai pateikiami 40 pav., 42 pav. ir **Error! Reference source not found.**

```
   . . .
<owl:Class rdf:ID="Vynas">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="GeriamasSkystis"/>
</rdfs:subClassOf>
<owl:Class rdf:about="#GeriamasSkystis">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="ValgomasDaiktas"/>
  </rdfs:subClassOf>
 </owl:Class>
<owl:Class rdf:about="#Gamintojas">
  <owl:disjointWith rdf:resource="#Cukringumas"/>
  <owl:disjointWith rdf:resource="#VynuogiuRusis"/>
  <owl:disjointWith rdf:resource="#Aprasas"/>
</owl:Class>
<owl:Class rdf:ID="Spalva"/>
   . . .
```

40 pav. Klasių aprašo OWL sintakse fragmentas

```
    . . .
<owl:ObjectProperty rdf:ID="TuriSpalva">
  <rdfs:domain rdf:resource="#Vynas"/>
  <rdfs:subPropertyOf>
   <owl:ObjectProperty rdf:about="#TuriAprasa"/>
  </rdfs:subPropertyOf>
  <rdfs:range rdf:resource="#Spalva"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="TuriVynuogiuRusi">
  <rdfs:domain rdf:resource="#Vynas"/>
  <rdfs:range rdf:resource="#VynuogiuRusis"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#TuriGamintoja">
  <rdfs:range rdf:resource="#Gamintojas"/>
  <rdfs:domain rdf:resource="#Vynas"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#TuriAprasa">
  <rdfs:domain rdf:resource="#Vynas"/>
  <rdfs:range rdf:resource="#Aprasas"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#TuriCukringuma">
  <rdfs:domain rdf:resource="#BurgundiskasVynas"/>
  <rdfs:range rdf:resource="#Cukringumas"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="PagaminimoMetai">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
  <rdfs:domain rdf:resource="#Vynas"/>
</owl:DatatypeProperty>
     . . .
```

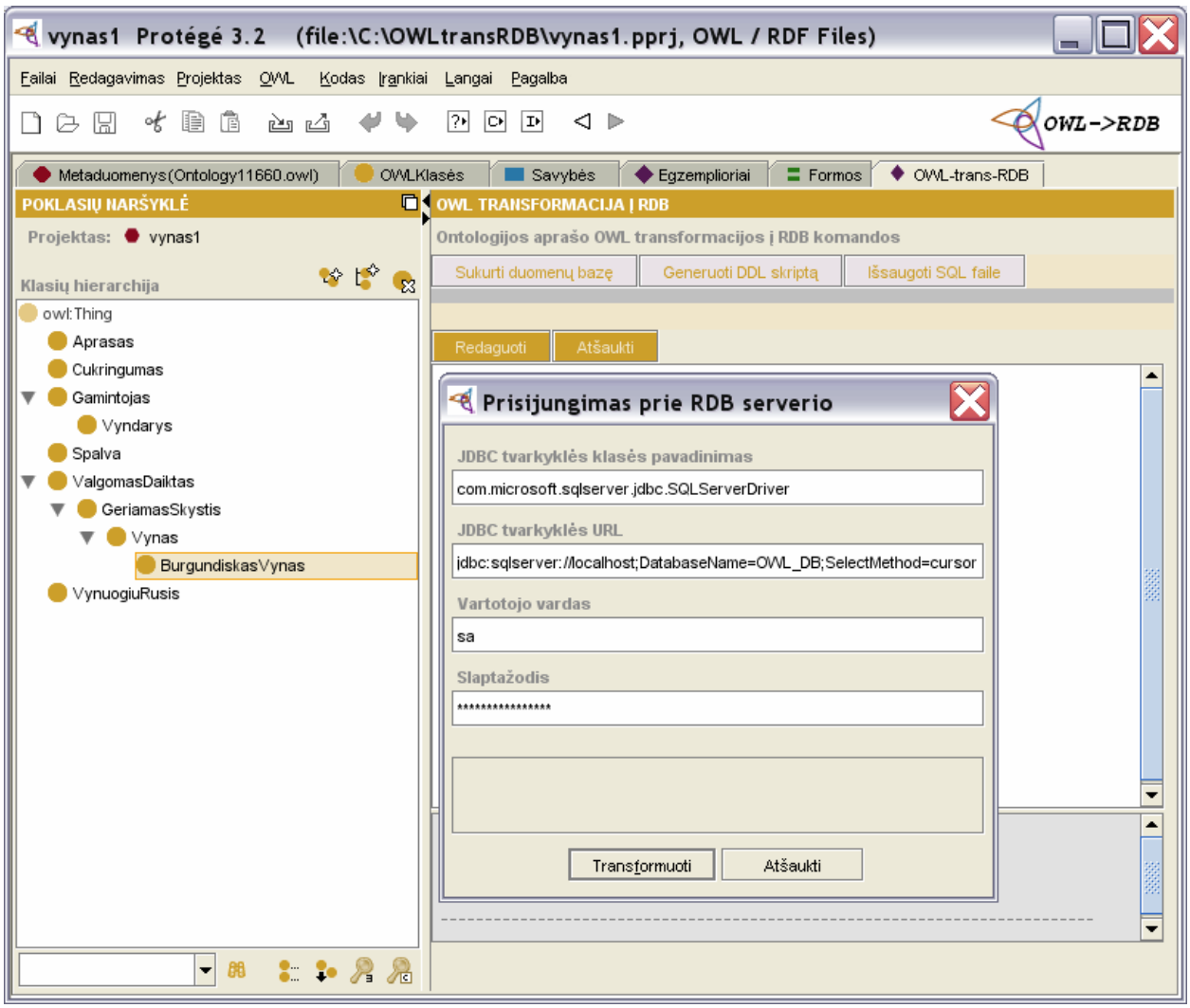
41 pav. Savybių aprašo OWL sintakse fragmentas

42 pav. Apribojimų aprašo OWL sintakse fragmentas

7.3 Ontologijos OWL aprašo transformavimas į reliacinę duomenų bazę

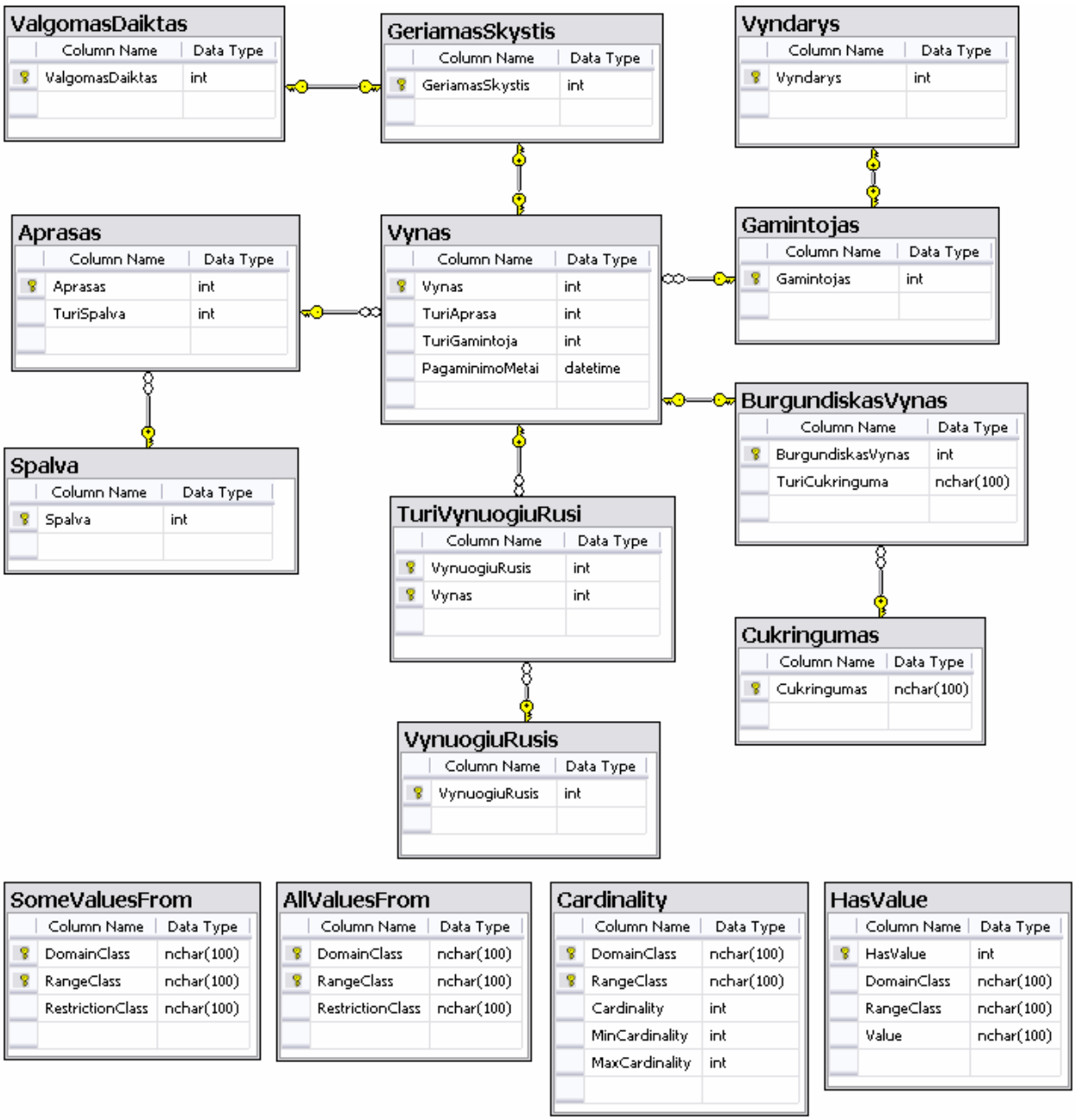
Sukūrus produkto ontologiją ir išsaugojus ją OWL tipo faile, ontologijos aprašas turi būti transformuotas į reliacinę duomenų bazę. Transformacijai naudojome sukurtą „OWLTransRDB“ įrankį. Nuskaičius OWL failą ir atvaizdavus klasių hierarchinę struktūrą matyti, kad ji atvaizduojama taisyklingai, taip kaip ir „SemanticWorks“ sistemoje (43 pav.). Pasirenkame komandą „Sukurti duomenų bazę“ ir atsidariusiame lange sukuriame prisijungimą prie duomenų bazių valdymo sistemos.

Eksperimentiniam tyrimui buvo panaudota „MS SQL Server 2005“ duomenų bazių valdymo sistema. Tai viena iš galingiausių ir plačiausiai naudojamų komercinių sistemų, turinti gerus įrankius, leidžiančius vizualiai peržiūrėti sukurtą duomenų bazės struktūrą.



43 pav. Ontologijos OWL aprašo transformacijos į RDB sistema

Atlikus produkto „Vynas“ ontologijos atvaizdavimą į reliacinę duomenų bazę, gauta schema pavaizduota 44 pav.



44 pav. Produkto "Vynas" ontologijos atvaizdas reliacinėje duomenų bazėje

Apribojimų lentelės užpildomos tokiais metaduomenimis:

Lentelė „AllValuesFrom“

DomainClass	RangeClass	RestrictionClass
Vynas	*Gamintojas*	*Vyndarys*

Lentelė „SomeValuesFrom“

DomainClass	RangeClass	RestrictionClass
Vynas	*Gamintojas*	*Vyndarys*

Lentelė „HasValue“

HasValue	DomainClass	RangeClass	Value
1	*BurgundiskasVynas*	*Cukringumas*	*Sausas*

Lentelė „Cardinality“

DomainClass	RangeClass	Cardinality	minCardinality	maxCardinality
Vynas	*Aprasas*	*1*	*Null*	*Null*

Atlikus transformaciją, gauta vienuolika duomenų lentelių ir keturios metaduomenų lentelės. Savybės, tokios kaip „TuriAprasa“, transformuotos į išorinius raktus, rodančius ryšius su tėvinėmis lentelėmis. Specialūs OWL apribojimai, pavyzdžiui „AllValuesFrom“, yra išsaugomi metaduomenų lentelėse. Metaduomenų lentelės užpildomos duomenimis taip, kaip aprašyta sukurtame OWL2DB algoritme. Pavyzdžiui, kardinalumo apribojimas vynui turėti tik vieną aprašą, išsaugomas lentelėje „Cardinality“, įrašant DomainClass lauke reikšmę „Vynas“, RangeClass – „Aprasas“, o Cardinality lauke – 1.

Ši nedidelė duomenų bazė gali būti naudojama taikomųjų programų, atliekančių tam tikrų vynų semantinę paiešką bendroje vynų aibėje.

7.4 Darbo rezultatų kritinė analizė

Darbo metu sukurtas algoritmas ir įrankis, kuris leidžia atlikti ontologijos, aprašytos OWL kalba, transformaciją į reliacinę duomenų bazę. Dabar apžvelgsime šio darbo rezultatų privalumus ir trūkumus.

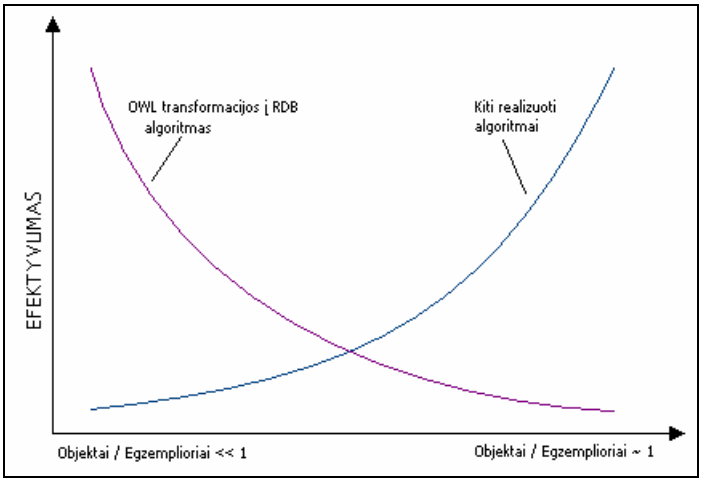
Visų pirma, jau egzistuoja keletas algoritmų arba požiūrių, kaip reikėtų atlikti ontologijų transformaciją į reliacines duomenų bazes. Tačiau jie arba yra tik tam tikri pasiūlymai, kurie nėra pilnai apibrėžti, pavyzdžiui Anuradha Gali [2] siūlomas algoritmas, arba siekiama visai kitokių tikslų, todėl atliekamos transformacijos yra visiškai skirtingos, nei šiame darbe.

Dažniausiai siūlomi OWL aprašo išsaugojimo į duomenų bazę principai yra panašūs i Holger Knublauch [12] sukurtą algoritmą, kurį naudoja „*Protege*“ sistema. Teigiama, jog ta sistema leidžia vartotojams duomenų bazėje išsaugoti ontologijas su šimtais tūkstančių klasių. Tuo tarpu šiame darbe aptartas algoritmas to padaryti tikrai nepajėgus, nes tuomet reikėtų duomenų bazės, kurioje būtų šimtai tūkstančių lentelių.

Tačiau šie algoritmai tuo ir skiriasi, kad skirtingi yra jų panaudojimo tikslai. Sistemos, kurios leidžia išsaugoti ontologijas, turinčias šimtus tūkstančių klasių, naudoja duomenų bazes todėl, kad saugant tokį kiekį OWL tekstiniame faile, labai išauga struktūrų nagrinėjimo laikas ir ontologijų valdymo įrankiai nepajėgūs atlikti jokių veiksmų. Paprastai tokių sistemų naudojamos duomenų bazės sudarytos iš vienos ar poros lentelių su daug laukų, ir duomenų bazės struktūra yra iš anksto apsibrėžta, t.y. universali visoms ontologijoms. Taip yra todėl, kad siekiama tik išsaugoti ontologiją, kurioje yra daug skirtingų objektų (klasių) ir mažai tų objektų egzempliorių. Tokia duomenų bazės struktūra visiškai netinka įprastoms informacinėms sistemoms, kadangi ji neatitinka reliacinėms duomenų bazėms keliamų normalinių formų. Būtų labai sudėtinga sukurti

informacinę sistemą, kur informacija būtų saugoma tokioje duomenų bazėje, dar sunkiau būtų užtikrinti efektyvų jos veikimą.

Šiame darbe siūloma pažvelgti kitaip – pritaikyti ontologijas informacinių sistemų kūrimo procese. Algoritmas pritaikytas tam atvejui, kai yra daug objektų egzempliorių, bet santykinai mažai jų tipų. Sistema, konstruodama duomenų bazės schemą, laikosi reliacinių taisyklių ir tik šiek tiek nukrypsta nuo jų, išsaugodama klasių apribojimus metaduomenų lentelėse. Apibendrinant, jeigu algoritmo efektyvumą vertinsime tokiais parametrais, kaip informacijos paieškos greitis, duomenų struktūrų sudėtingumas bei taikomosios sistemos sukūrimo paprastumas, abiejų tipų algoritmus galima palyginti abstrakčiu grafiku – kaip augant objektų ir egzempliorių santykiui, kinta algoritmų efektyvumas (45 pav.).



45 pav. Algoritmo efektyvumas kintant ontologijos objektų ir egzempliorių santykui

Dar vienas kritinis algoritmo įvertinimas atliekamas pagal OWL savybes, kurias galima transformuoti į reliacinę duomenų bazę. Šiame darbe aprašytas algoritmas gali atlikti klasių, objektinių ir duomenų tipų savybių, egzempliorių ir dalies apribojimų transformaciją į RDB. Tai apima OWL Lite ir dalį OWL DL sintaksės galimybių. Egzistuoja ir OWL Full notacija, kurioje yra dar daugiau apribojimų tipų ir kitų specifinių būdų aprašyti tam tikros dalykinės srities ontologiją.

Norint sukurti algoritmą, kuris atliktų visų galimų ontologijos savybių transformaciją į duomenų bazę, reikėtų laikytis tų pačių principų, t.y. visas specifinius apribojimus saugoti jiems skirtose metaduomenų lentelėse. Tačiau egzistuoja tam tikra ontologijos sudėtingumo riba, nuo kurios darosi neprasminga saugoti informacijos reliacinėje duomenų bazėje, nes jos struktūra tampa pernelyg sudėtinga. Įprastose informacinėse sistemose ši riba dažniausiai nepasiekiama, ji būdinga specifinėms žinių sistemoms, taigi, šis algoritmas leis panaudoti ontologijas informacinių sistemų kūrimo procese.

Išvados

Ontologijų panaudojimas įvairių sričių informacinių sistemų kūrime tampa vis populiaresnis, ypatingai tokiose sudėtingose ir painiose, kokios yra konfigūravimo sistemos. Šiame darbe buvo analizuotas procesas, kaip tam tikros dalykinės srities ontologijos aprašas OWL kalba galėtų būti transformuotas ir išsaugotas reliacinėje duomenų bazėje. Apibendrinant tyrimo rezultatus, suformuluotos tokios išvados:

1. Atlikus analizę galima teigti, kad ontologijos yra perspektyvus ir įgyvendinamas būdas patobulinti ir supaprastinti produktų konfigūracijos kūrimą ir jų palaikymą viso produkto gyvavimo ciklo metu.
2. Literatūros šaltinių analizė parodė, kad trūksta ontologijų transformavimo į reliacines duomenų bazes algoritmų, kurių panaudojimas leistų padidinti informacinių sistemų kūrimo bei informacijos paieškos ir pateikimo uždavinių efektyvumą.
3. Sukurtas transformacijos algoritmo pritaikomumo tyrimas ontologijos konstrukcijų aibei atskleidė, kad efektyviai galima transformuoti pagrindines OWL konstrukcijas, kurios apima OWL Lite ir dalį OWL DL (klasės, savybės, apribojimai, egzemplioriai), tačiau jo plėtimas sudėtingesnėms konstrukcijoms gali būti neefektyvus.
4. Algoritmo eksperimentinis tyrimas transformuojant produkto ontologijos aprašą parodė, kad dabartinė algoritmo versija transformacijos metu informacijos nepraranda, tačiau galimybė atvaizduoti sudėtingesnes OWL savybes reikalautų išsamesnio tyrimo.
5. Darbo rezultatų pritaikomumo analizė nustatė, kad pasiekti darbo tikslai – algoritmas, atvaizduojantis ontologijos klases į RDB lenteles, savybes į ryšius ir atributus, o apribojimus į metaduomenis – leis ontologijas panaudoti informacinių sistemų kūrimo procese.
6. Sukurtą metodą galima efektyviai taikyti informacinėse sistemose, kur klasių egzempliorių ir klasių skaičiaus santykis yra daug didesnis už vienetą. Jei šis santykis artimas vienetui ir klasių skaičius didelis, tokių žinių apdorojimas reliacinėse duomenų bazėse gali būti neefektyvus.

Literatūra

[1] A. Felfernig, G. Friedrich, and D. Jannach. UML as domain specific language for the construction of knowledge-based configuration systems. International Journal of Software Engineering and Knowledge Engineering. Vol. 10, Nr. 4. 2000.

[2] A. Gali, C. X. Chen, K.T. Claypool, R. Uceda-Sosa. From Ontology to Relational Databases. In: Shan Wang et all (Eds.): Conceptual Modeling for Advanced Application Domains, LNCS Vol. 3289, pp. 278-289, 2005.

[3] B. Parsia, E. Sirin, A. Kalyanpur. Debugging OWL ontologies. In: The 14th international world wide web conference (www2005), Chiba, Japan, May 2005. Available at http://www.mindswap.org/papers/ debuggingOWL.pdf 2005.

[4] C. Perez de Laborda, S. Conrad. Relational.OWL – a data and schema representation format based on OWL. In: Hartmann, Sven and Stumptner, Markus (Eds.): Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005)}, ACS, Australia, Vol. 43, pp.89-96, 2004.

[5] Ernestas Vysniauskas, Lina Nemuraite. Produkto konfigūravimo žinių bazės ontologinis modeliavimas. Tarpuniversitetinė doktorantų magistrantų konferencija "Informacinės technologijos 2006"IT 2006, Kaunas, pp. 133-138.

[6] Ernestas Vysniauskas, Lina Nemuraite. Transforming Ontology Representation from OWL to Relational Database. In: The 5th international conference “Business Informatics Research 2006“BIR2006, Kaunas. Information technology and control, ISSN 1392-124X, pp. 333-343.

[7] G. Antoniou, F. van Harmelen. Web Ontology Language: OWL. In: S. Staab and R. Studer (Eds.): Handbook on Ontologies in Information Systems, Springer-Verlag, 2003.

[8] G. Guizzardi, G. Wagner, N. Guarino, M. Sinderen. An Ontologically Well-Founded Profile for UML Conceptual Models. In: A.Person and J.Stirna (Eds.): CAISE 2004, LNCS 3084, pp. 112-126, 2004.

[9] G. Guizzardi. The role of Foundational Ontologies for Conceptual Modelling and Domain Ontology Representation. In: O.Vasilecas et all (Eds.): Proceedings of 2006 Seventh International Baltic Conference on Databases and Information systems (Baltic DB&IS 2006), July 3-6, Vilnius, Lithuania, pp. 17-25, 2006.

[10] I. Astrova. Towards the Semantic Web – An Approach to Reverse Engineering of Relational Databases to Ontologies. In: Advances in Databases and Information Systems: proceedings of the 9th East-European Conference, ADBIS 2005, Tallin, September 12-15, 2005. - ISBN 9985-59-545-9. - Tallin, 2005, p. 111-122.

[11] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. Journal of Web Semantics, 1(1), pp.7–26, 2003.

[12] J. Z. Pan, I. Horrocks. OWL-Eu: Adding Customised Datatypes into OWL. Journal of Web Semantics, 2005, Vol. 4(1), pp. 1-20.

[13] L. Hotz, T. Krebs. Configuration – State of the art and New Challenges. In: L. Hots, T. Krebs (Eds.) Proc. Of 17 th Workshop Plannen, Scheduling and Configurieren, Entwerfen, (PUK2003) – KI 2003 Workshop, pp.145-157, 2003

[14] L. Hotz, T. Krebs. Supporting the product derivation process with a knowledge-based approach. In: Hots, L., Krebs, T. (Eds.) Proc. Of 17th Workshop Plannen, Scheduling and Configurieren, Entwerfen, (PUK2003) – KI 2003 Workshop, p. 24-29, 2003.

[15] L. Hvam. A Multi-perspective Approach for the Design of Product Configuration Systems – an Evaluation of Industry Applications. In: International Conference of Economic, Technical and organizational aspects of Product Configuration Systems, Technical University of Denmark, June 28-29th, pp.1-12, 2004.

[16] N. Guarino, C. Welty. Conceptual Modeling and Ontological Analysis. Available online: http://www.cs.vassar.edu/faculty/welty/aaai-2000/. (ž. 2007-01-03).

[17] N. Pena, E. Garcia, J. M. Lazaro. Configuration Ontology & Multi-product Configuration Tool (I). *O*ntology-*B*ased *EL*ectronic *I*ntegration of Comple*X* Products and Value Chains IST Project IST-2001-33144 OBELIX, 2003.
[18] N. Shadbolt, T. Berners-Lee, W. Hall. The Semantic Web revisited. In: IEEE Intelligent Systems. 2006, pp. 96-101.
[19] Ontology Definition Metamodel. Preliminary Revised Submission to OMG RFP ad/2003-03-40, 18 August, 2004.
[20] R. Goodwin, J. Lee , G. A. Stanoi, M. I. Leveraging Relational Database Systems for Large-Scale Ontology Management. In: CIDR Conference, 2005. Available online: http://www.alphaworks.ibm.com/topics/semantics.
[21] S. Brockmans, P. Haase, and P. Hitzler. A Metamodel and UML Profile for Rule-extended OWL DL Ontologies. In: Y. Sure, J. Domingue (Eds): The Semantic Web: Research and Applications: 3rd European Semantic Web Conference, ESWC 2006 Budva, Montenegro, June 11-14, 2006 Proceedings, LNCS, Vol. 4011, 2006, pp. 303-316.
[22] S. Decker. Semantic Web and Databases: Relationships and some Open Problems. Proceedings of the NSF-EU Workshop on Database and Information Systems: Research for Semantic Web and Enterprises, April 3 - 5, Amicalola Falls and State Park, Georgia, pp.6-14
[23] T. R. Gruber. What is an Ontology? Available online: http://ksl-web.stanford.edu/kst/what-is-an-ontology.html. (ž. 2007-01-03).
[24] T. Krebs, L. Hotz, C. Ranze, G. Vehring. Towards evolving configuration models. In: Hots, L., Krebs, T. (Eds.) Proc. Of 17 th Workshop Plannen, Scheduling and Configurieren, Entwerfen, (PUK2003) – KI 2003 Workshop, pp.123-134, 2003.

Terminų ir santrumpų žodynas

DBVS	Duomenų bazių valdymo sistema. Programinės įrangos rinkinys skirtas organizuoti informacijai duomenų bazėje
Duomenys	Faktai ar teiginiai, kurie yra ar gali būti surinkti, išsaugoti, apdoroti, tačiau nėra organizuoti ar pateikti kontekste
Informacija	Duomenų prasmė, reikšmė, numatyta interpretacija. Duomenų organizavimas, susiejimas asociacijomis, apribojimas, sudarantis galimybę juos panaudoti
Informacinė technologija	Technologija naudojama kurti, apdoroti, saugoti, išrinkti, keistis, naudoti ir pan. bet kokių formų informaciją; ypač – naudojant kompiuterius bei programinę įrangą
Informacinė sistema	Automatizuota ar rankinė sistema, apimanti priemones ir komponentus informacijos surinkimui, apdorojimui, saugojimui, persiuntimui, pateikimui, platinimui, panaudojimui
Koncepcinis modelis	Rinkinys prielaidų, kurios supaprastina realią problemą ar realią taikomąją sritį iki patenkinamo požiūrio apie modeliavimo tikslus ir susijusias valdymo problemas; (Minimalus) rinkinys abstrakčių sąvokų, kurios aprašo tam tikrą taikomąją sritį arba užduočių ar sistemų klasę.
Modelis	Supaprastintas sudėtingos esybės ar proceso aprašas (pvz. formulių rinkiniu), atvaizdas ar pan.
Ontologija	Tam tikros srities bendrai naudojamos sąvokų/konceptų, esybių tipų, jų tarpusavio priklausomybių, sąryšių, aksiomų, dėsningumų ir kt. visumos formalus aprašas.
OWL (angl. *Web Ontology Language*)	Ontologijos aprašymui skirta kalba
RDB	Reliacinė duomenų bazė
RDBVS	Reliacinių duomenų bazių valdymo sistema
RDF (angl. *Resource Description Framework*)	Išteklių aprašymo kalba
SQL (angl. *Structured Query Language*)	Struktūrinių užklausų kalba
XML (angl. *Extensible Markup Language*)	Išplečiamoji žymėjimo kalba, naudojama aprašyti duomenims
Žiniomis grindžiama sistema	Kompiuterinė programa, turinti deklaratyvią žinių bazę, kurioje laikomos užkoduotos žmogaus žinios problemų sprendimu.

Priedas. Publikuoti straipsniai

TRANSFORMING ONTOLOGY REPRESENTATION FROM OWL TO RELATIONAL DATABASE

Ernestas Vysniauskas, Lina Nemuraite

Kaunas University of Technology, Department of Information Systems
Studentų st. 50-308, LT-51368 Kaunas, Lithuania
{ vernest@email.lt, lina.nemuraite@ ktu.lt}

Abstract. The current work has arisen with respect to the growing importance of ontology modelling in Information Systems development. Due to emerging technologies of Semantic Web, it is desirable to use for this purpose the Web Ontology Language OWL. From the other side, the relational database technology has ensured the best facilities for storing, updating and manipulating the information of problem domain. The algorithms for transformation of domain ontology, described in OWL, to relational database are proposed. The methodology is illustrated with an example.

1. Introduction[1]

Ontology is a description of entities of a problem domain and their semantic relations. The short definition of the ontology is as the "explicit specification of conceptualization". The ontology clearly defines fundamental concepts and relationships used in particular problem domain. Ontology representations may vary:

- from simple taxonomy with minimal hierarchy of knowledge – only father/child relationships;
- to thesaurus (extended vocabulary), including words and synonyms;
- to conceptual model, having more complex relationships;
- and, finally, to logic theory, including very complex, expressive, and meaningful knowledge.

Well-formed ontology must have correct syntax and unambiguous machine-understandable interpretation adequate to its previous definition. Ontology descriptions are typically used in Semantic Web/Web2, but nowadays they find more and more adaptability in domains associated with everyday Information Systems. For ontology development, the Semantic Web languages and technologies are dedicated: Resource Description Framework RDF and schema RDFS; Web Ontology Language OWL that consists of three sublanguages – OWL Lite, OWL Description Logic (DL) and OWL Full. But, as [8] argue, the representation of ontology based on Semantic Web languages is insufficient to address semantic interoperability problems that arise in various concrete applications.

"Ontologies are attempts to more carefully define parts of the data world and to allow interactions between data held in different formats" [16]. In conceptual modelling, the Foundational Ontology is needed as domain-independent theoretical basis to guide and validate models of particular domains, as using of right modelling concepts and rules is making a great influence on the quality of Information Systems [7]. For such purpose, the transformations between conceptual models (expressed, for example, in UML) and ontological models, expressed in ontological languages (for example, OWL) are needed.

In this paper, another problem is considered that has arisen from practical needs: namely, possibilities for storing ontological information and processing this information by user applications. For this purpose, Relational Database (RDB) is a good candidate that have proven capabilities to cope with large amounts of data [18]. Methodologies for transforming Entity-relationship and Object-oriented (in nowadays, often expressed in UML) conceptual models to relational database structures, transformations between relational and XML schemas, UML models and XML schemas are well-established and implemented in CASE tools. Ontology Definition Metamodel, initiated by OMG [17], is seeking to define transformations between OWL, UML, ER and other modelling languages, where Simple Common Logic is chosen for definition of constraints. On the base of existing methodologies, there are some possible ways to relate ontological information described by ontological language, with relational schemas:

- Generation of standard ontology descriptions (*e.g.*, OWL) and relational schemas from UML models maybe created using UML profile for ontology modelling [19];
- Generation of UML models from standard ontology descriptions (e.g., OWL) and relational schemas from UML models;
- Extracting or representing ontological descriptions from existing relational databases [9], [4] (unfortunately, this way does not ensure quality of ontological model).

Other scenarios for relating OWL and Relational Databases are possible, although, the direct transformation from OWL to relational schema is not defined anywhere. It is not only feasible way, but sure the fast and theoretically valid one to move ontological information to relational database and to make it accessible by different applications and systems. As

[1] The work is supported by Lithuanian State Science and Studies Foundation according to Eureka programme project „IT-Europe" (Reg. No 3473)

OWL is built on XML, it seems worth to try transformations from XML to RDB. But OWL has more advanced features, than XML documents, namely, constraints and inference support, and these features must be preserved when mapping OWL to RDB. The only source, giving an outline of algorithm for going from OWL to relational data structures, preserving constraints (OWL2DB), is [2]. Our algorithms, presented in this paper, are created on the base of [2] ideas.

The rest of the paper is organized as follows. In section 2, the overview of configuration domain is given, as this domain has become the subject of large amount of ontological research as well as a target of plenty practical applications. Section 3 presents the example of configuration ontology in OWL. Section 4 is devoted to explanation of transformation algorithms from OWL to RDB schema. In Section 5, relational database schema is presented for analyzed ontology example. Finally, Section 6 gives some conclusions and highlights the future work.

2. Ontology of a product configuration

One of domains often considered in association with ontology development is a product configuration system [14]. There are a lot of problems that may be solved from the configuration point of view. For example, customers often desire individual products and solutions, designed for their own needs. One of the ways to solve this problem is to let customer to design the conspicuous product by himself using virtual product configuration system. Product configuration systems or configurators are software tools that enable automation of order submissions by capturing customer requirements without human intermediaries, constructing new products and providing semantic search in supplier's database.

To do all these tasks, product configuration systems need knowledge about products, their components, various design rules and constraints. This information must be stored in a configuration system's knowledge base.

Configuration is an abstract description of system elements and their composition rules. One of the most important features of configuration management system is a semantic design of the product using the configuration knowledge base. The functioning of configuration system is based on formal configuration models. Entities of product configuration ontology are:

- `Product`
- `Product::product_atribute`
- `Product::product_component`
- `Product::necessary_component::product_component`
- `Product::product_component::inconsistent_component`
- `Product::similar_product`
- `Product::incompatible_product`, and others.

The configuration model is a strictly described (formalized) configuration ontology, or meta information about the certain class of products. In this paper, configuration models are considered on two levels – formal (logical) and relational. Formal model is described in OWL, and later transformed into relational schema, described using Data Definition Language (DDL) that may be executed for creating a relational meta database for storing relational descriptions of particular configuration model.

Product configuration systems or configurators are considered as being among the most successful applications of artificial intelligence. They facilitate entering of information about products and provide semantic search in the particular product class by any characteristic proper to that class. In general, configurator implements an interface between a supplier and his customer over the internet. Its main task is to support customers in the self-configuration of their products according to particular individual requirements [12]. For example, customers may be provided with the possibility to alter a basic product and also to graphically visualize the effects of these changes.

Configurators support the configuration process. This process may by handled by an agent right understanding the customer needs in order to create a complete description of the product variant that suits his or her individual requirements. Given a set of customer requirements and the description of a family of products, the task of configuration is to find a valid and completely specified product instance among all of the alternatives that the generic structure describes.

In the technical literature, there are many definitions of product configurators. The artificial intelligence community generally addresses them as the software tools. For example, in [15] product configurator is defined as a "…software with logic capabilities to create, maintain and use electronic product models that allow to complete definition of all possible product options and variation combinations, with a minimum of data entries and maintenance". The main technical component of the configurator is the knowledge base, which consists of two subcomponents, namely, the database, and the configuration logic.

Whereas the database contains the total set of component types and their instances, the configuration logic specifies the constraints existing between the different components to allow only valid and completely structured product variants.

Within a product portfolio, there is a lot of knowledge and rules about the products and their relationships to one another, yet this is hidden in the

data without which it would be hardly possible to extract and use product configurations in supporting tools. Hence typical applications upon product catalogue data (supply chain, marketing, research & development, B2C e-commerce) are implemented in a costly hard-coded approach in which catalogue data are imported and manipulated as they suit the particular context of use.

Due to the structure of product catalogue data, in which characteristics and relationships are not explicitly expressed by some sort of standardized terminology, there is a high cost involved in developing and maintaining this data, reinterpreting it for different contexts of use, or for creating new catalogue structures, working with them in collaborative environments, and sharing them between different participants, each of which may have a different understanding of their purpose and meaning [20].

As a result, an ontology usage is a viable approach to improving the development of product catalogues and their maintenance over the entire product lifecycle, as they offer the consistent terminology and the possibility to generate different views for different contexts for the same products [13], [21].

3. Modelling the configuration ontology

Ontological model simplifies the development and support of the knowledge base of business domain [6], [10], [3], [11] as it defines well-established terminology, and enables the generation of different views of the same object according to existing context. The preferred language for ontology modelling is OWL, as it has the most possibilities for expressing semantics in comparison with RDF and RDFS (Table 1). The exclusive feature of ontological languages is the capability to express constraints and individuals (objects) of problem domain.

Table 1. The capabilities of ontological languages

Concepts	RDF(S)	OWL
Bounded lists	X	X
Cardinality constraints		X
Class definitions		X
Data types	X	X
Defined classes		X
Enumerations		X
Equivalence		X
Extensions	X	X
Formal semantics	X	X
Inheritance	X	X
Inference		X
Constraints		X
Reification	X	X

In order to develop the ontology of some domain, one must define the fundamental concepts, relationships, constraints and individuals of that domain. [15] defines the configuration ontology in such a way:

- A set of components (products or services), such, that these components may be described by a set of properties and ports connecting them to other components.
- Constraints for each port describing the components that may be connected to that port, and other structural constraints.
- User requirements with the description of the desired configuration; and, possibly, some criteria for optimizing the solution.

Therefore, the main concepts that seem to be the candidates to appear in the configuration ontology are components, ports, connections, properties or attributes, constraints, and user requirements.

In Figure 1, the configuration ontology schema is presented using these concepts.

4. The process of development of a knowledge base

There are some alternatives, how to create a knowledge base. For example, Felferning et al. [1] has proposed the following process for a configuration ontology creation: firstly, a model of the configuration object is developed using a modelling language, e.g. UML with OCL constraints. After testing the syntax of concepts, model is unambiguously transformed to logical expressions, which are used by configuration engine generating domain configurations. Finally, knowledge base must be checked by expert of business domain using testing examples.

The way, considered in our work, is presented in Figure 2 [5]. Business analyst gives specifications of configuration domain, expressed in natural language, to knowledge engineer (designer of knowledge-based Information System). Designer, using some modelling tool (for example, Protege, Altova Semantic Works, or other), creates formal ontology for required domain. After that, the ontology model is transformed into DDL script with all constraints using special transforming tool. The DDL script is used to save the knowledge descriptions into a relational database. Client applications may access this relational database and render results to users (Figure 2).

5. Transformation of ontology to relational knowledge base

One of the steps of knowledge base constructing process is to transform domain ontology into relational database. For this purpose, the transformation algorithm was created, which parses OWL documents and generates DDL scripts, containing database descriptions with included domain ontology constraints.

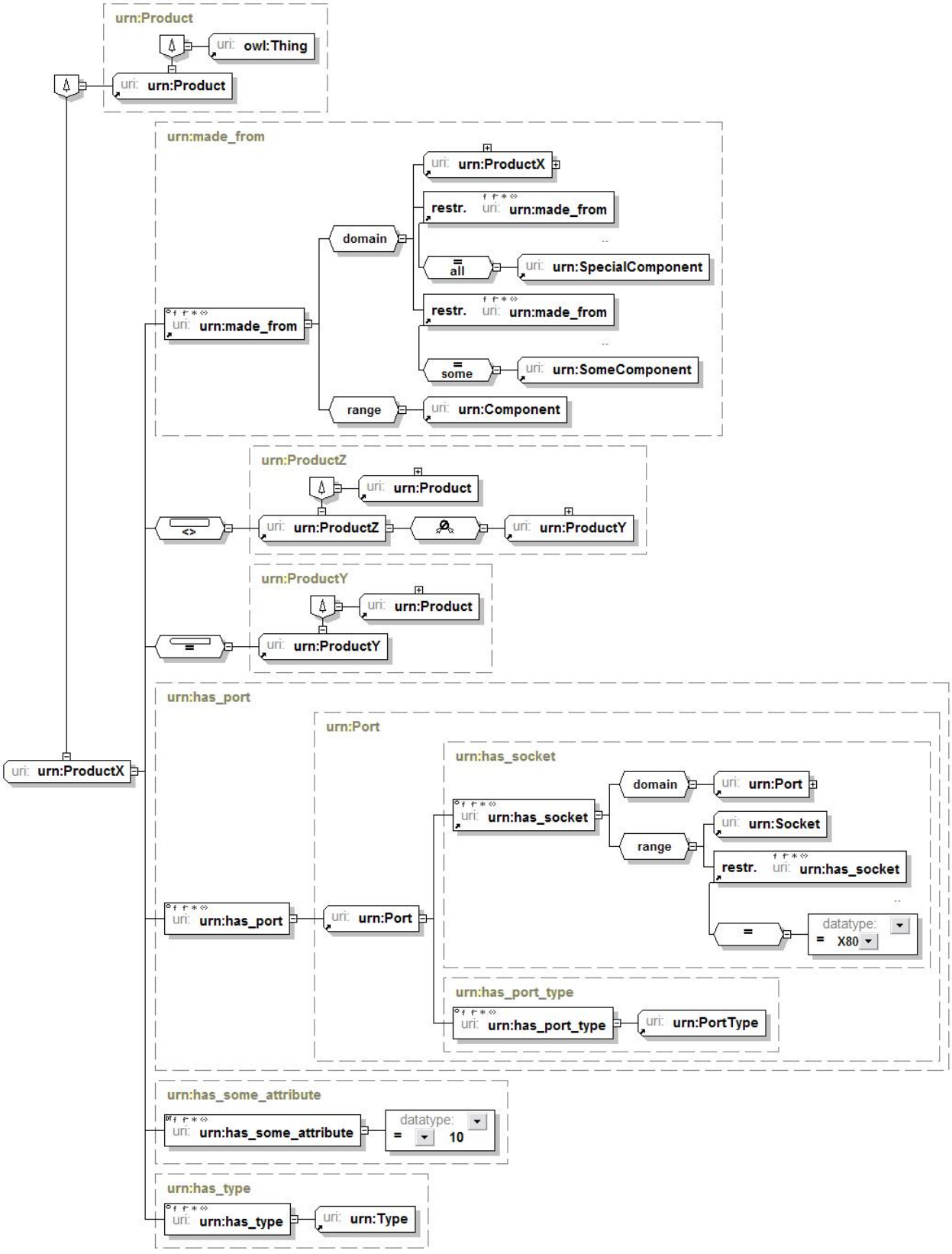


Figure 1. An example of configuration ontology

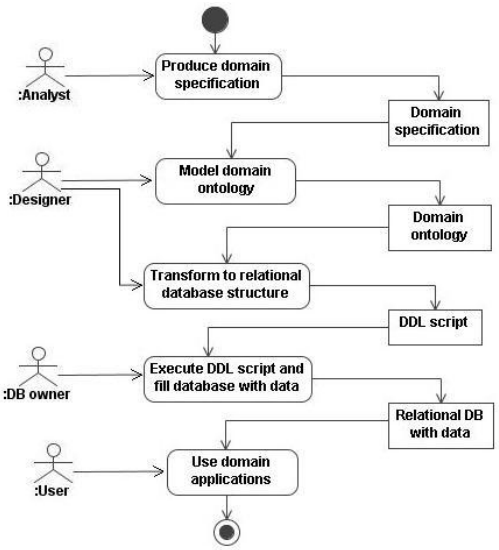


Figure 2. The proposed process for developing and transforming domain ontology to relational database

The process of transforming ontology into relational database is shown in Figure 3.

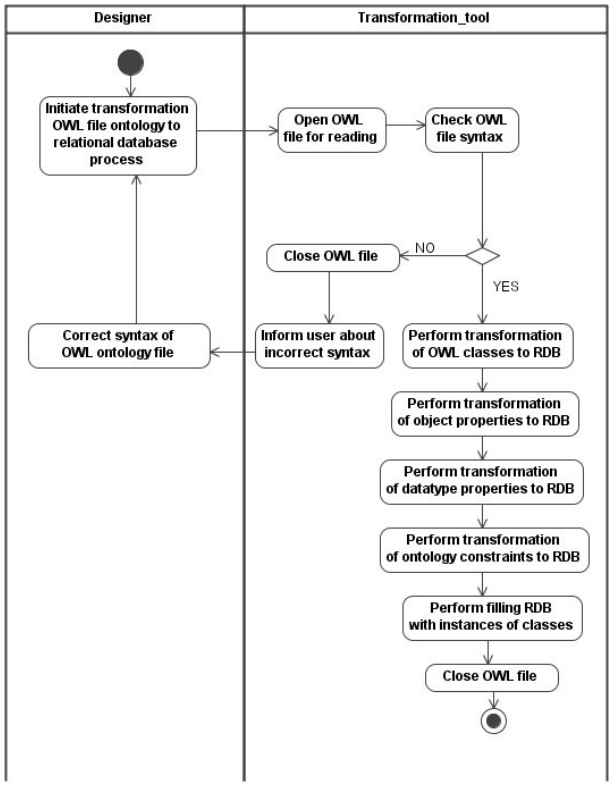


Figure 3. Algorithm for transformation of ontology into relational database

Designer initiates transformation of domain ontology described in OWL file into relational database. Transformation tool opens OWL file for reading and checks the correctness of OWL syntax. If the file syntax does not match OWL notation, system closes it and informs designer about errors. If syntax of the file matches OWL notation, transformation tool executes steps of transforming ontology to relational database. At first, system transforms ontology classes, the next steps are transformations of object and data type properties, constraints and, finally, database is filled with instances of the classes. At the end of successful transformation system closes the file.

The more detailed steps of the algorithm are presented in Figures 4, 5, and 6.

5.1 Transformation of ontology classes into RDB tables

During the process of transforming domain ontology into relational database, at the first step the ontology classes are transformed into relational database tables.

A class is the most basic concept in ontology. Class in OWL syntax is defined in this way:

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

The fundamental taxonomic constructor for classes is "`rdfs:subClassOf`". It relates a more specific class to a more generic class. If "X" is a subclass of "Y", then every instance of "X" is also an instance of "Y". The "`rdfs:subClassOf`" relation is transitive. If "X" is a subclass of "Y" and "Y" a subclass of "Z", then "X" is a subclass of "Z". Example of OWL syntax, defining class hierarchical relations:

```
<owl:Class rdf:ID="PotableLiquid">
  <rdfs:subClassOf
rdf:resource="#ConsumableThing" />
  ...
</owl:Class>
```

Every ontology class is implicitly a subclass of "`owl:Thing`" class.

An algorithm, transforming ontology classes into relational database tables, is shown in Figure 4. It uses breadth-first search. Breadth-first search algorithm goes across one hierarchical level of ontology class tree. Thus, root classes are parsed first, then their subclasses, and so on. The one table in relational database is created for every class in ontology with one-to-one relations between classes and their subclasses. Because algorithm uses breadth-first search, it is guaranteed that, when some subclass is being created, its parent class in hierarchy has already been created.

5.2 Transformation of ontology object-properties into RDB algorithm

After creating ontology class tables, object-properties are transformed into RDB relations between class tables.

Object property is a relation between instances of two classes. When we define a property, there is a number of ways to restrict the relation: the domain and

range can be specified; the property can be defined to be a specialization (sub-property) of an existing property, and so on.

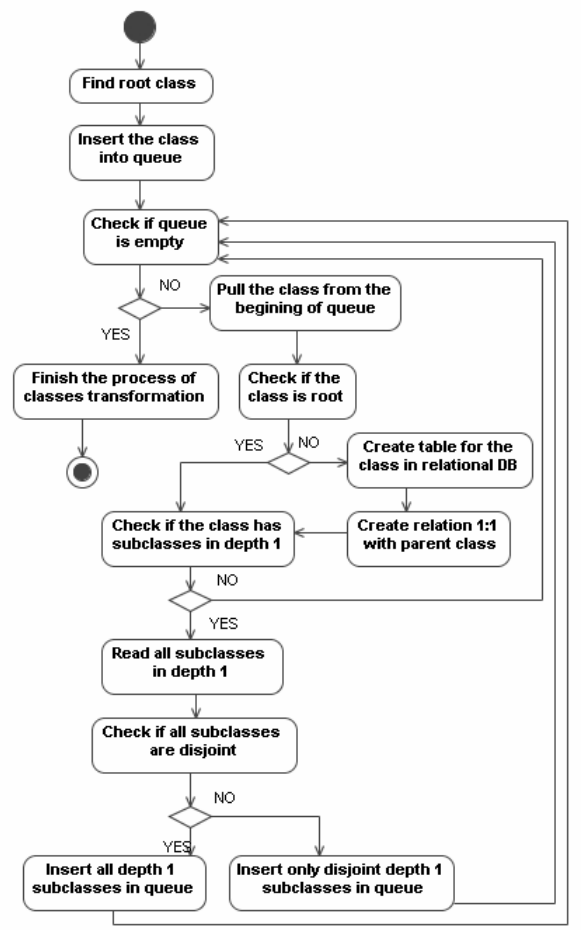


Figure 4. Transformation of ontology classes into relational database tables

Example of OWL syntax, defining object property:

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

Example of OWL syntax defining specialization of object-properties:

```
<owl:Class rdf:ID="WineDescriptor" />
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource=
      "#WineDescriptor" />
  ...
</owl:Class>
<owl:ObjectProperty rdf:ID=
      "hasWineDescriptor">
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource=
      "#WineDescriptor" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasColor">
<rdfs:subPropertyOf rdf:resource=
      "#hasWineDescriptor" />
  <rdfs:range rdf:resource="#WineColor" />
  ...
  </owl:ObjectProperty>
```

An algorithm, transforming ontology object-properties into relational database relations between tables is shown in Figure 5. This algorithm also uses breadth-first search. Firstly, it parses properties that do not have properties of the higher hierarchical level; in the next, it parses their sub-properties, and so on. Depending on the local cardinality of some class property, one-to-many or many-to-many relations between tables of classes are created. In a case of many-to-many relation, an intermediate table is created.

When the designer initiates transformation of domain ontology described in OWL file into relational database, transformation tool opens OWL file for reading and checks the correctness of OWL syntax. If the file syntax does not match OWL notation, system closes it and informs designer about errors.

If syntax of the file matches OWL notation, transformation tool executes steps of transforming ontology to relational database. Primarily, system transforms ontology classes, next steps are transformation of object and data type properties, constraints and finally database is filled with instances of the classes. At the end of successful transformation system closes the file, replacing one many-to-many relation with two one-to-many relations.

5.3 Transformation of ontology data type properties into RDB algorithm

In the process of transforming domain ontology into relational database, after transformation of object-properties into RDB relations between class tables, data type-properties are transformed into RDB data columns.

Data type properties are relations between instances of classes and RDF literals, and XML Schema data types. Example of OWL syntax, defining data type property:

```
<owl:Class rdf:ID="VintageYear" />

<owl:DatatypeProperty rdf:ID="yearValue">
  <rdfs:domain rdf:resource="#VintageYear" />
  <rdfs:range rdf:resource=
      "&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

An algorithm, transforming ontology data type properties into relational database data columns of tables is shown in Figure 6.

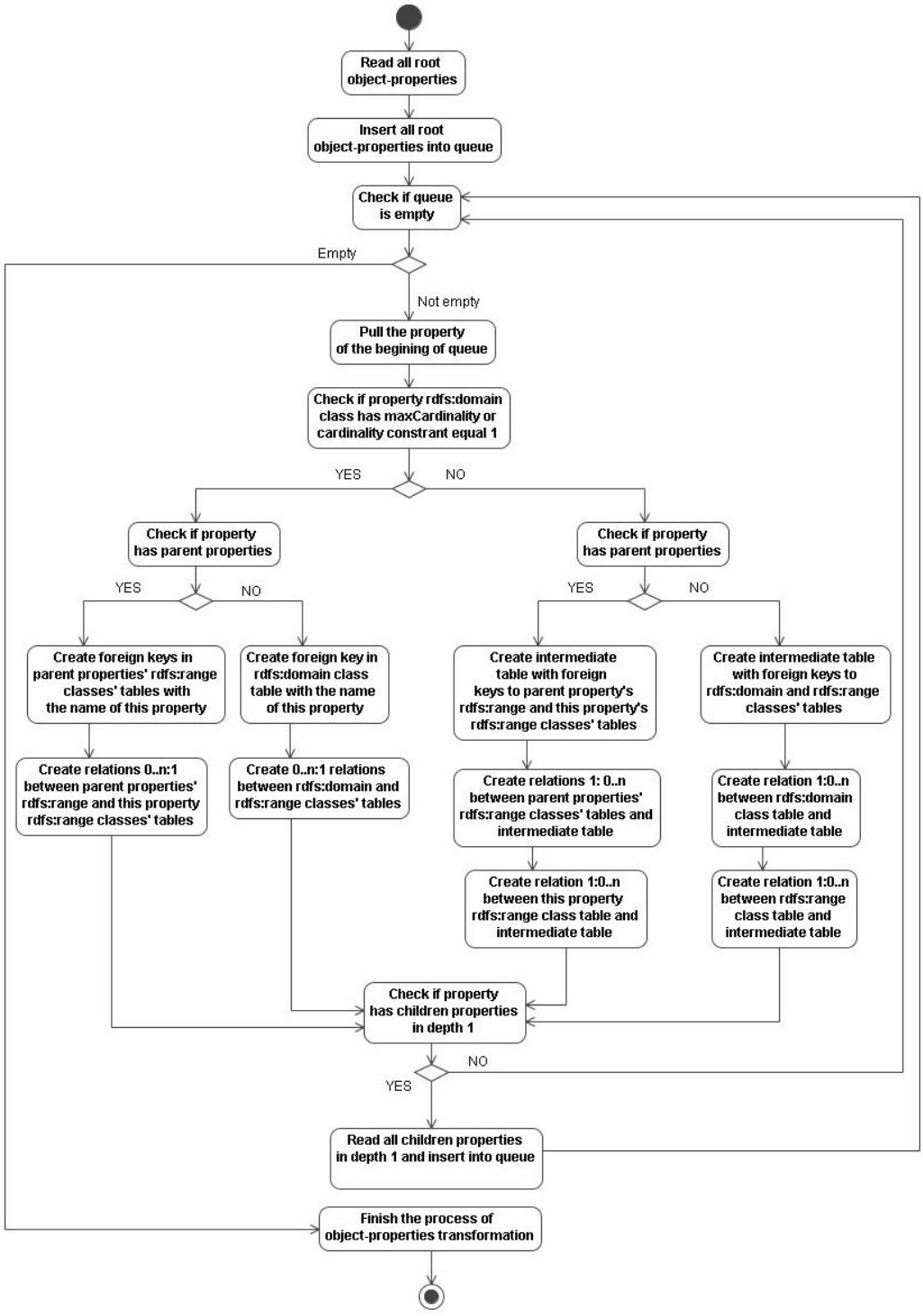


Figure 5. Transformation of object properties into relational database relations

The algorithm, transforming ontology data type properties into relational database columns of tables, searches and parses all data type properties in series. According to `rdfs:domain` value it finds database table and creates data column with the name of the property. Column data type is a set according to property `rdfs:range` value. When all data type properties are parsed, the process of data type properties transformation is finished.

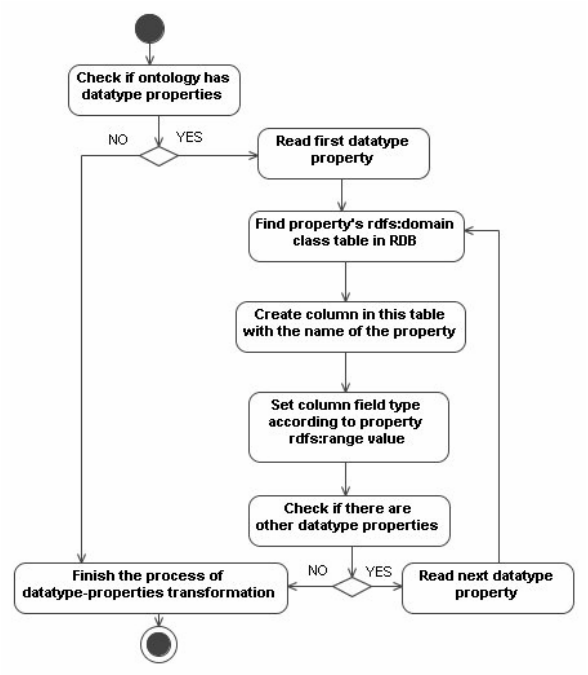


Figure 6. Transformation of data type properties into relational database table columns

2.1 Transformation of ontology constraints into RDB algorithm

In the process of transforming domain ontology into relational database, after transformation of data type-properties into RDB data columns, ontology constraints are transformed into RDB metadata tables.

In addition to designating property characteristics, it is possible to further constrain the range of a property in specific contexts in a variety of ways. It is done with property restrictions. The various forms of restrictions can only be used within the context of an `owl:Restriction`. The `owl:onProperty` element indicates the restricted property. Example of OWL syntax, defining a restriction of class property:

```
<owl:Class rdf:ID="Wine">

  <rdfs:subClassOf rdf:resource=
     "&food; PotableLiquid"/>
  <rdfs:subClassOf>
<owl:Restriction>
  <owl:onProperty rdf:resource=
     "#madeFromGrapes"/>
  <owl:minCardinality
     rdf:datatype=
     "&xsd;nonNegativeInteger">1
  </owl:minCardinality>
 </owl:Restriction>
</rdfs:subClassOf>

...

</owl:Class>
```

An algorithm, transforming ontology constraints into relational database metadata tables is shown in Figure 7.

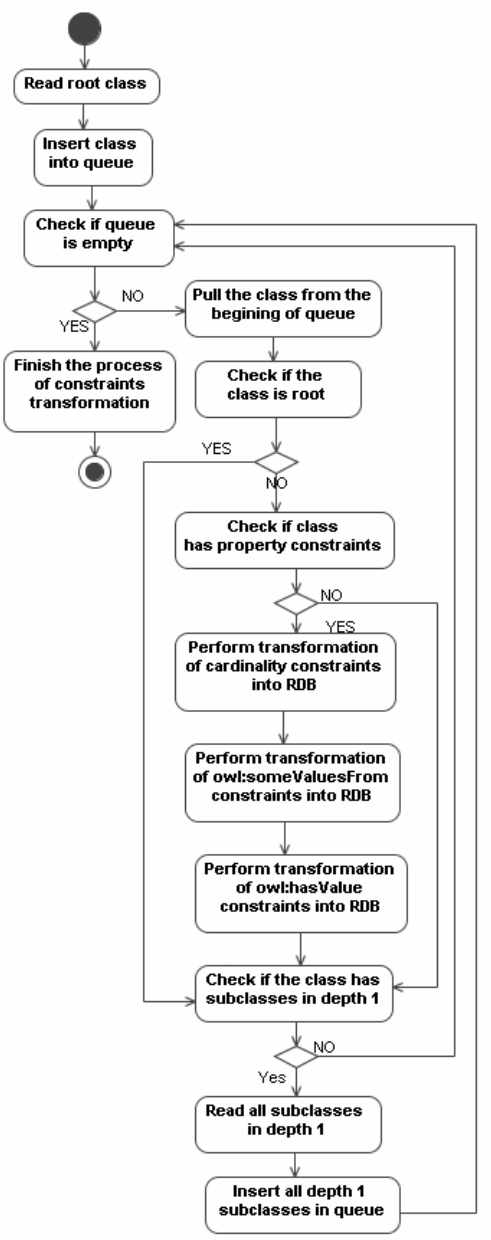


Figure 7. Transformation of ontology constraints into relational database metadata tables

Algorithm, transforming ontology constraints into relational database metadata tables performs breadth-first search. Firstly it parses constraints of root class properties, then constraints on properties of its subclasses, and so on. If a class has constraints of the property, the algorithm performs transformation of constraints into metadata tables. All constraints of particular class are parsed in series. Every type of constraint has its own table with the name of the constraint type.

At the last stage of transforming domain ontology into relational database, transformation tool inserts all instances of classes into created database.

5.5 An example of transforming ontology into RDB

We have standard wine ontology shown in Figure 8. Wine ontology is often used as an example when we talk about ontology, because it is simple and understandable for everybody. A wine itself, as a potable liquid, is not a configurable thing; however the wine as a product or a description of the product is fully configurable, because it has many different features, such as wine grapes or vintage year, restrictions, such as sugary and other properties. We can create new wine descriptions or perform the semantic search in the existing ones, so wine is the fully configurable product from this viewpoint. Using OWL2DB algorithm to transform the wine ontology presented on Figure 8 into relational database we obtain the resulting schema presented on Figure 9.

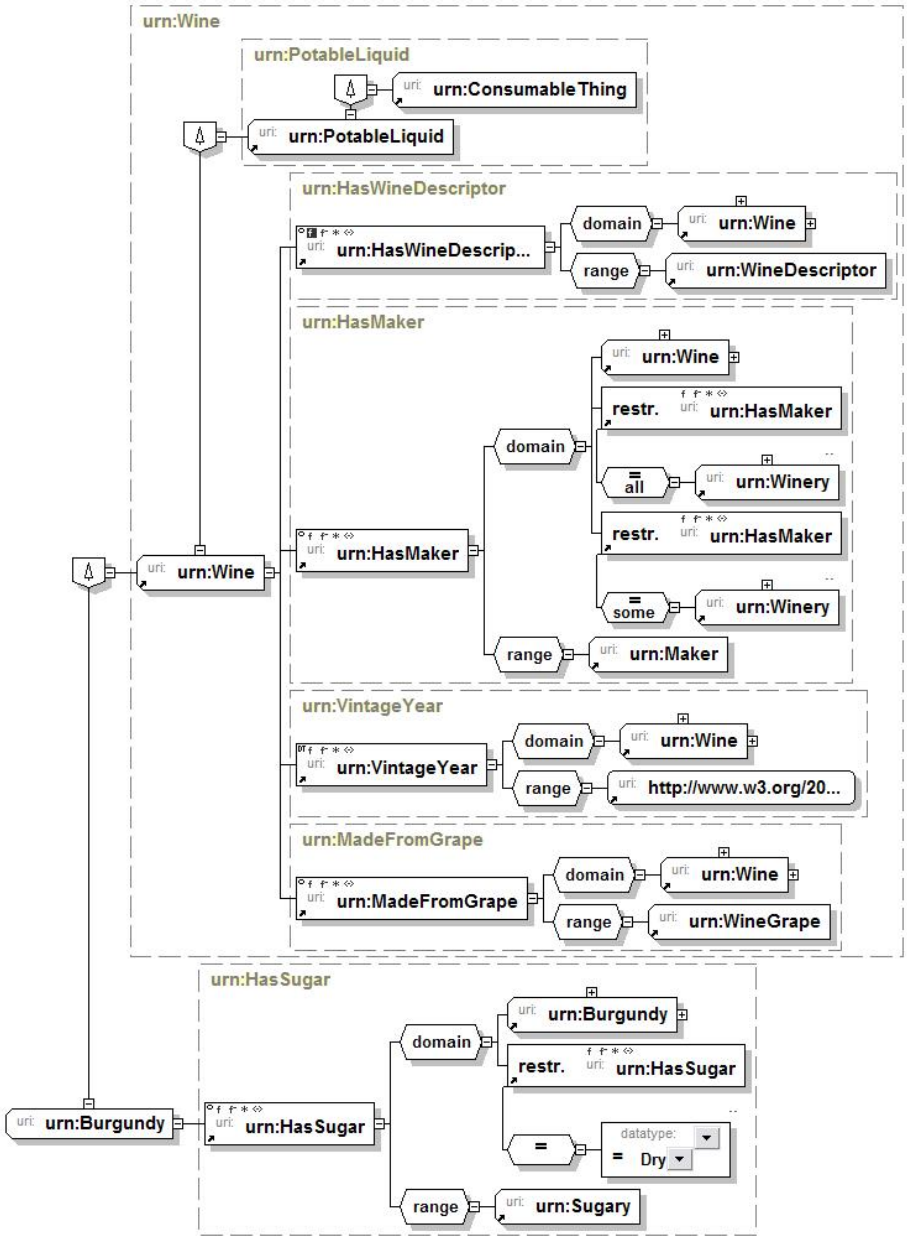


Figure 8. Example of wine ontology

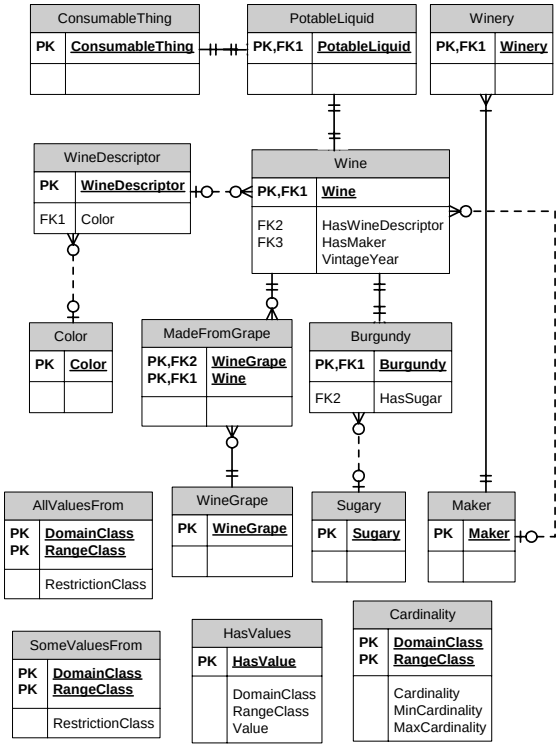


Figure 9. Example of wine ontology transformed into relational database

Meta data tables have been filled with this metadata:

Table „AllValuesFrom“

DomainClass	RangeClass	RestrictionClass
Wine	Maker	Winery

Table „SomeValuesFrom“

DomainClass	RangeClass	RestrictionClass
Wine	Maker	Winery

Table „HasValue“

HasValue	DomainClass	RangeClass	Value
1	Burgundy	Sugary	Dry

Table „Cardinality“

Domain Class	Range Class	Cardi nality	Min Cardinality	Max Cardinality
Wine	Wine Descriptor	1	Null	Null

After transformation process we have got eleven data tables and four metadata tables (Figure 9). Properties, such as “HasWineDescriptor” are transformed into foreign keys indicating to parent tables. Special OWL constraints, such as “AllValuesFrom” are kept in metadata tables. Metadata tables are filled with data according to the algorithm, described previously in the article. This minimal database structure can be used by applications, performing semantic search of special wines in general set of wines.

6 Conclusions

Ontological descriptions are gaining more and more popularity as a perspective way to enhance Information Systems in different problem domains, especially for such complex ones as configuration systems are. In this paper, we have analyzed the process how ontology of a particular domain described in OWL may be transformed and stored in a relational database. Summarizing we can make some conclusions:

– For large ontological descriptions, it is desirable to store ontological information in relational databases, although there is a lack of algorithms suitable to transform ontology concepts to RDB schema.

– The algorithm was created for transforming ontology, represented in OWL, to RDB schema. According this algorithm, ontology classes are mapped to relational tables, properties to relations and attributes, and constraints – to metadata. Such an algorithm was not created before, although it is partially based on the OWL2DB approach.

– The proposed algorithm is capable to transform all OWL Lite and part of OWL DL syntax. The further expansion of the algorithm to cover more capabilities of OWL should be based on the same principles.

– The algorithm was tested for transforming ontology examples from product configuration domain. The current version of algorithm works without a loss of information though possibilities to represent more advanced OWL features require for thorough investigation.

– A tool, performing transformations, may be implemented as add-in for ontology development tool (e.g. Protege), or as independent software, capable to import OWL documents, and to export generated DDL scripts.

References

[1] A. Felfernig, G. Friedrich, and D. Jannach. UML as domain specific language for the construction of knowledge-based configuration systems. International Journal of Software Engineering and Knowledge Engineering. Vol. 10, Nr. 4. 2000.

[2] A. Gali, C. X. Chen, K.T. Claypool, R. Uceda-Sosa. From Ontology to Relational Databases. In: Shan Wang et all (Eds.): Conceptual Modeling for Advanced Application Domains, LNCS Vol. 3289, pp. 278-289, 2005.

[3] B. Parsia, E. Sirin, A. Kalyanpur. Debugging OWL ontologies. In: The 14th international world wide web conference (www2005), Chiba, Japan, May 2005. Available at http://www.mindswap.org/papers/debuggingOWL.pdf 2005.

[4] C. Perez de Laborda, S. Conrad. Relational.OWL – a data and schema representation format based on OWL. In: Hartmann, Sven and Stumptner, Markus (Eds.): Second Asia-Pacific Conference on Conceptual

Modelling (APCCM2005)}, ACS, Australia, Vol. 43, pp.89-96, 2004.
[5] Ernestas Vysniauskas, Lina Nemuraite. The ontological modeling of product configuration knowledge base. Tarpuniversitetine doktorantu magistrantu konferencija "Informacines technologijos 2006"IT 2006, Kaunas, pp. 133-138.
[6] G. Antoniou, F. van Harmelen. Web Ontology Language: OWL. In: S. Staab and R. Studer (Eds.): Handbook on Ontologies in Information Systems, Springer-Verlag, 2003.
[7] G. Guizzardi, G. Wagner, N. Guarino, M. Sinderen. An Ontologically Well-Founded Profile for UML Conceptual Models. In: A.Person and J.Stirna (Eds.): CAISE 2004, LNCS 3084, pp. 112-126, 2004.
[8] G. Guizzardi. The role of Foundational Ontologies for Conceptual Modelling and Domain Ontology Representation. In: O.Vasilecas et all (Eds.): Proceedings of 2006 Seventh International Baltic Conference on Databases and Information systems (Baltic DB&IS 2006), July 3-6, Vilnius, Lithuania, pp. 17-25, 2006.
[9] Astrova I. Towards the Semantic Web – An Approach to Reverse Engineering of Relational Databases to Ontologies. In: Advances in Databases and Information Systems: proceedings of the 9th East-European Conference, ADBIS 2005, Tallin, September 12-15, 2005. - ISBN 9985-59-545-9. - Tallin, 2005, p. 111-122.
[10] I. Horrocks, P. F. Patel-Schneider, F. van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. Journal of Web Semantics, 1(1), pp.7–26, 2003.
[11] J. Z. Pan, I. Horrocks. OWL-Eu: Adding Customised Datatypes into OWL. Journal of Web Semantics, 2005, Vol. 4(1), pp. 1-20.
[12] L. Hotz, T. Krebs. Configuration – State of the art and New Challenges. In: L. Hots, T. Krebs (Eds.) Proc. Of 17 th Workshop Plannen, Scheduling and Configurieren, Entwerfen, (PUK2003) – KI 2003 Workshop, pp.145-157, 2003
[13] L. Hotz, T. Krebs. Supporting the product derivation process with a knowledge-based approach. In: Hots, L., Krebs, T. (Eds.) Proc. Of 17th Workshop Plannen, Scheduling and Configurieren, Entwerfen, (PUK2003) – KI 2003 Workshop, p. 24-29, 2003.
[14] L. Hvam. A Multi-perspective Approach for the Design of Product Configuration Systems – an Evaluation of Industry Applications. In: International Conference of Economic, Technical and organizational aspects of Product Configuration Systems, Technical University of Denmark, June 28-29th, pp.1-12, 2004.
[15] N. Pena, E. Garcia, J. M. Lazaro. Configuration Ontology & Multi-product Configuration Tool (I). *O*ntology-*B*ased *EL*ectronic *I*ntegration of Comple*X* Products and Value Chains IST Project IST-2001-33144 OBELIX, 2003.
[16] N. Shadbolt, T. Berners-Lee, W. Hall. The Semantic Web revisited. In: IEEE Intelligent Systems. 2006, pp. 96-101.
[17] Ontology Definition Metamodel. Preliminary Revised Submission to OMG RFP ad/2003-03-40, 18 August, 2004.
[18] R. Goodwin, J. Lee , G. A. Stanoi, M. I. Leveraging Relational Database Systems for Large-Scale Ontology Management. In: CIDR Conference, 2005. Available online: http://www.alphaworks. ibm.com /topics/semantics.
[19] S. Brockmans, P. Haase, and P. Hitzler. A Metamodel and UML Profile for Rule-extended OWL DL Ontologies. In: Y. Sure, J. Domingue (Eds): The Semantic Web: Research and Applications: 3rd European Semantic Web Conference, ESWC 2006 Budva, Montenegro, June 11-14, 2006 Proceedings, LNCS, Vol. 4011, 2006, pp. 303-316.
[20] S. Decker. Semantic Web and Databases: Relationships and some Open Problems. Proceedings of the NSF-EU Workshop on Database and Information Systems: Research for Semantic Web and Enterprises, April 3 - 5, Amicalola Falls and State Park, Georgia, pp.6-14
[21] T. Krebs, L. Hotz, C. Ranze, G. Vehring. Towards evolving configuration models. In: Hots, L., Krebs, T. (Eds.) Proc. Of 17 th Workshop Plannen, Scheduling and Configurieren, Entwerfen, (PUK2003) – KI 2003 Workshop, pp.123-134, 2003.

ONTOLOGIJOS APRAŠO TRANSFORMAVIMAS IŠ OWL Į RELIACINĘ DUOMENŲ BAZĘ

Ernestas Vysniauskas, Lina Nemuraite

Santrauka

Šį darbą paskatino didėjanti ontologijos modeliavimo svarba kuriant informacines sistemas. Derinantis prie semantinio žiniatinklio technologijų, ontologiją tikslinga modeliuoti OWL kalba. Tačiau dideliems informacijos kiekiams saugoti ir apdoroti geriau tinka reliacinės duomenų bazės, kurios ne kartą įrodė savo privalumus. Straipsnyje pateikiamas algoritmas, leidžiantis transformuoti OWL aprašytą ontologiją į reliacinės duomenų bazės schemą. Pateikiamas iliustruojantis pavyzdys.

PRODUKTO KONFIGŪRAVIMO ŽINIŲ BAZĖS ONTOLOGINIS MODELIAVIMAS

Ernestas Vyšniauskas, Lina Nemuraitė

Kauno technologijos universiteto Informatikos fakulteto Informacijos sistemų katedra

Straipsnyje aptariama standartizuoto produkto žinių modelio svarba konfigūravimo sistemos kūrimo procese. Remiantis atlikta žinių bazių modeliavimo būdų analize, šiam tikslui siūloma naudoti interneto ontologijų aprašymo kalbą OWL. Iš kitos pusės, geriausią produktų informacijos saugojimą, atnaujinimą ir manipuliavimą užtikrina reliacinių duomenų bazių technologija. Straipsnyje pateikiama konfigūravimo ontologijos kūrimo ir atvaizdavimo į reliacinę duomenų bazę metodika.

1. Įvadas

Visi šiuolaikiniame pasaulyje pageidauja produktų, kuo nors išsiskiriančių iš bendros produktų masės, labiau atitinkančių jų norus, skonį ar interesus. Vartotojai nori jiems asmeniskai skirtų individualių produktų ar sprendimų. Vienas iš būdų kurti tokius produktus yra konfigūravimos sistemos [1]. Produktų konfigūravimo sistemos arba konfigūratoriai yra informaciniai įrankiai, kurie leidžia automatizuoti užsakymų priėmimo procesą, surinkdami kliento užklausas ir reikalavimus be žmogiškų tarpininkų, lengvai konstruoti naujus gaminius ir atlikti semantinę produktų paiešką.

Visiems šiems darbams atlikti konfigūravimo sistemoms reikalingos žinios apie produktus, jų sudedamąsias dalis, įvairios konstravimo taisyklės ir apribojimai. Šios žinios saugomos konfigūravimo sistemų žinių bazėje.

2. Ontologijos apibrėžimas ir pagrindinės sąvokos

Ontologija − tai probleminės srities esybių ir jų semantinių ryšių aprašymas. Ontologija turi daug apibrėžimų. Trumpiausias apibrėžimas: ontologija yra aiškiai išreikšta konceptualizacijos specifikacija. Ji apibrėžia tam tikroje žinių srityje naudojamas išraiškas ir konceptus (prasmę) bei jų tarpusavio ryšius. Ontologija gali būti laikoma:

- Taksonomija, turinti minimalią žinių hierarchiją – tėvo/vaiko ryšius;
- Tezauras (išplėstas žodynas), apimantis žodžius ir sinonimus;
- Konceptualus modelis, turintis sudėtingesnius ryšius;
- Loginė teorija, apimanti labai sudėtingas, išraiškingas, suderintas ir prasmingas žinias.

Taisyklinga ontologija turi būti išreikšta aiškiai apibrėžta sintakse, turinčia aiškią mašininę interpretaciją, atitinkančią ankstesnį ontologijos apibrėžimą.

Konfigūracija − tai abstraktus sistemos elementų ir jų komponavimo taisyklių aprašas. Viena iš svarbiausių konfigūravimo valdymo sistemos savybių – semantinis produkto konstravimas remiantis konfigūravimo žinių bazėje saugoma informacija. Darbe nagrinėjamos konfigūravimo sistemos veikimas yra paremtas meta duomenų bazėje saugomais konfigūracijų modeliais.

Konfigūracijos ontologija – tai konfigūraciją aprašančių esybių ir jų semantinių ryšių aprašymas. Formalizuotas ontologijos aprašymas šiame darbe vadinamas modeliu. Produkto konfigūracijos ontologijos esybės gali būti tokios:

- Produktas;
- Produktas::produkto_atributas;
- Produktas::produkto_elementas;
- Produktas::būtinas_elementas::produkto_elementas;

- Produktas::produkto_elementas::nesuderinami_elementai;
- Produktas::panašūs_produktai;
- Produktas::nesuderinami_produktai.

Konfigūracijos modelis - tai griežčiau aprašyta (formalizuota) konfigūracijos ontologija. Konfigūracijos modelis yra meta informacija apie produktų klasę. Šiame darbe konfigūracijos modeliai yra dviejų lygių – formalus ir reliacinis. Formalus modelis yra aprašytas interneto ontologijų aprašymo kalba OWL (angl. *Web Ontology Language*) sintakse ir gali būti transformuojamas į reliacinį modelį. Reliacinis modelis aprašytas DDL (angl. *Data Definition Language*) sintakse, jis gali būti patalpintas į meta duomenų bazę. Meta duomenų bazė šiuo atveju − tai duomenų bazė, sauganti konkrečių konfigūracijos modelių reliacinius aprašus.

3. Produkto konfigūracijos valdymo sistemos

Produkto konfigūravimo sistemos arba konfigūratoriai yra laikomi vienomis iš sėkmingiausių programų, kuriose taikomos dirbtinio intelekto technologijos [1]. Konfigūratoriai yra informaciniai įrankiai, kurie leidžia įvesti duomenis apie produktus ir vykdyti paiešką konkrečioje produktų klasėje pagal bet kokią tai klasei būdingą savybę. Pagrindinė jų savybė yra semantinis produkto konstravimas remiantis konfigūravimo žinių bazėje saugoma informacija. Taip pat konfigūracijos valdymo sistemos leidžia automatizuoti užsakymų priėmimo procesą, surinkdamos klientų užklausas ir reikalavimus be žmogiškųjų tarpininkų.

Daugumoje atvejų konfigūratorius palaiko vartotojo sąsają tarp tiekėjo ir jo internetinio kliento. Jo pagrindinė užduotis yra padėti klientams patiems susikonfigūruoti produktus pagal savo pačių individualius poreikius [3]. Pavyzdžiui, klientams gali būti suteikiama galimybė keisti bazinį produktą, tuo pat metu šiuos pokyčius pavaizduojant grafiškai.

Konfigūratoriai palaiko patį konfigūravimo procesą. Šį procesą turi atlikti toks asmuo ar kompiuterinis agentas, kuris tiksliai suprastų klientų poreikius ir sugebėtų sukurti pilną produkto aprašą, kuris atitiktų individualius kiekvieno kliento reikalavimus. Konfigūravimo užduotis yra turint aibę kliento reikalavimų ir produktų šeimos aprašą, iš visų alternatyvų surasti veiksmingą, efektyvų ir pilnai specifikuotą atskirą produkto atvejį, kuris tenkintų bendrą visų šeimos produktų struktūrą.

Techninėje literatūroje sutinkama įvairių produkto konfigūratoriaus apibrėžimų. Bendru atveju dirbtinio intelekto bendruomenė konfigūratoriumi laiko programinę įrangą [6]. Pavyzdžiui, produkto konfigūratorius apibrėžiamas kaip „...programa, turinti logines galimybes kurti, palaikyti ir naudoti produktų modelius, kurie duoda pilną produktų pasirinkimų ir produktų variantų kombinacijų aprašą su minimalia įvedamų ir saugomų duomenų aibe“. Pagrindinis techninis konfigūratoriaus komponentas yra žinių bazė, kuri susideda iš dviejų dalių – duomenų bazės ir konfigūravimo logikos.

Duomenų bazė apima visą komponentų tipų ir egzempliorių aibę, tuo tarpu konfigūracijos logika apibrėžia tarp skirtingų komponentų egzistuojančius apribojimus, kas leidžia sudaryti tik taisyklingus ir pilnai struktūrizuotus produkto variantus.

Kataloge esančiame produkto aprašyme yra daugybė žinių ir taisyklių apie produktus ir jų tarpusavio ryšius, tačiau visa tai būna paslėpta duomenyse, kuriuos sunku ištraukti ir panaudoti programiniuose įrankiuose [1]. Dėl tokios produkto duomenų struktūros, kur savybės ir ryšiai nėra tiksliai išreikšti standartizuota terminologija, yra labai sudėtinga ir brangu manipuliuoti šiais duomenimis, naujai interpretuoti juos kitokiam panaudojimui arba kurti naują katalogo struktūrą, taip pat dalintis jais tarp skirtingų padalinių (kiekvienas jų gali turėti skirtingą duomenų reikšmės ir naudojimo tikslų supratimą).

Todėl bendrinės ontologijos ir ontologijų kalbos gali būti perspektyvus ir įgyvendinamas būdas patobulinti ir supaprastinti produktų katalogų kūrimą ir jų palaikymą viso produkto gyvavimo ciklo metu.

4. Ontologijų taikymas konfigūravimo žinių bazės ontologiniam modeliavimui

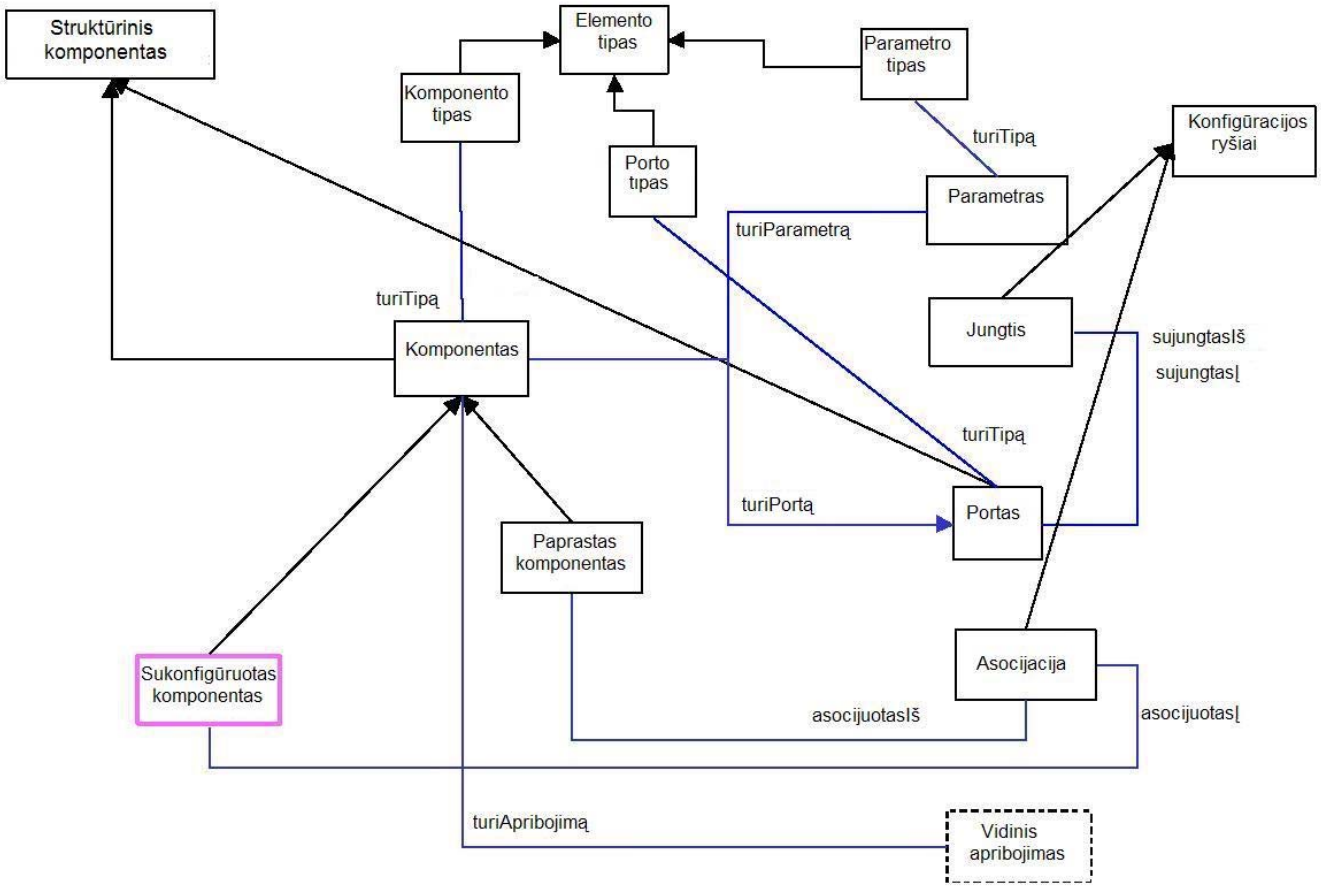
Produkto ontologinis modelis supaprastina konfigūravimo žinių bazės kūrimą ir palaikymą [7]. Ontologijos nustato pastovią produktų terminologiją, taip pat suteikia galimybę generuoti įvairius tų pačių produktų vaizdus priklausomai nuo esančio konteksto.

Norint sukurti konfigūravimo ontologiją, pirmiausia reikia išsiaiškinti, iš ko susideda konfigūracija. Pagal Mittal ir Frayman [4] apibrėžimą, konfigūracijos ontologija yra:

- Rinkinys komponentų (produktų), kurie gali būti aprašyti rinkiniais savybių ir jungčių (angl. *Connector*), jungiančių juos su kitais komponentais.
- Apribojimai kiekvienai jungčiai, apibrėžiantys, kokius komponentus prie jos galima jungti, ir kiti struktūriniai apribojimai.
- Vartotojo reikalavimai su pageidaujamos konfigūracijos aprašymu ir galimi sprendimo optimizavimo kriterijai.

Taigi, pagrindiniai konceptai, naudojami konfigūracijos ontologijoje, yra komponentai, jungtys, ryšiai, savybės arba atributai, apribojimai, vartotojo reikalavimai.

Žemiau esančiame modelyje (1 pav.) pateikiama konfigūracijos ontologija, sukurta naudojant šiuos konceptus.



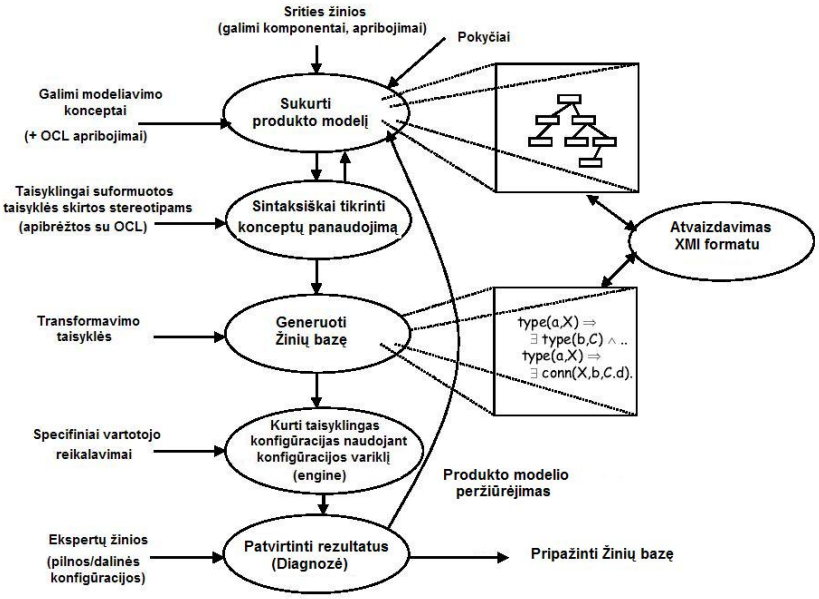
1 pav. Konfigūracijos ontologija

5. Produkto konfigūratoriaus žinių bazės kūrimo procesas

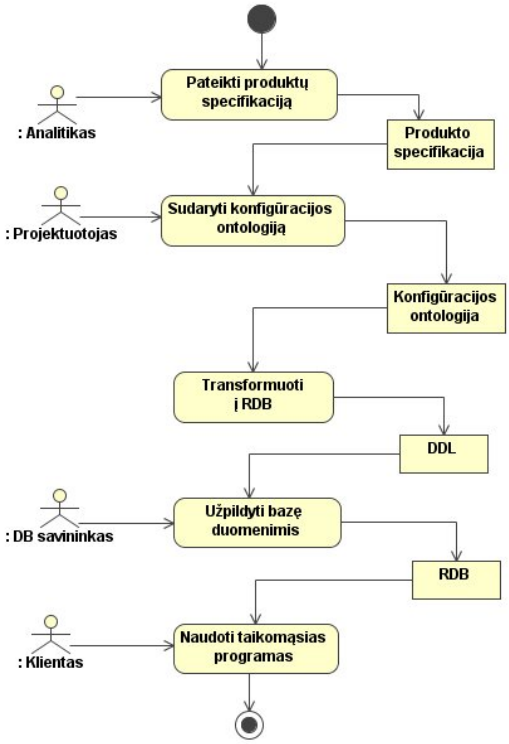
Aleksander Felferning [5] siūlomas konfigūracijos žinių bazės kūrimo procesas yra pateikiamas 2 pav. Pirmiausiai, naudojant kokią nors modeliavimo kalbą, pavyzdžiui, UML, yra sukuriamas konfigūruojamo produkto modelis. Po sintaksinio konceptų korektiškumo tikrinimo šis modelis nedviprasmiškai transformuojamas į logines išraiškas, kurias naudoja konfigūravimo variklis, generuojantis produktų konfigūracijas. Galiausiai žinių bazę tikrina probleminės srities ekspertas, testavimui naudodamas kontrolinius pavyzdžius.

Produkto konfigūracijos ontologinio modelio bei paties konfigūratoriaus kūrimo veiklos proceso modelis pateikiamas 3 pav. Probleminės srities analitikas pateikia IS projektuotojui produktų specifikacijas, t.y. produkto

konfigūraciją natūralia kalba. Projektuotojas, naudodamas modeliavimo įrankį, sukuria formalų produktų konfigūracijos modelį (ontologiją). Naudojant transformacijos įrankį, produkto konfigūracija su visais apribojimais atvaizduojama į DDL skriptą, kuris naudojamas išsaugoti produkto konfigūraciją reliacinėje duomenų bazėje. Kliento taikomoji programa kreipiasi į duomenų bazę ir naudodamasi ten esančiais produkto duomenimis bei metaduomenimis pateikia rezultatus vartotojui (4 pav.).



2 pav. Taisyklingos konfigūracijos žinių bazės konstravimo modelis



3 pav. Produkto konfigūratoriaus kūrimo proceso modelis

6. Žinių bazės ontologinio modelio saugojimas reliacinėje duomenų bazėje

Vienas iš darbo tikslų yra sukurtą konfigūracijos ontologiją išsaugoti reliacinėje duomenų bazėje. Tam reikia sukurti transformavimo algoritmą, kuris, nagrinėdamas OWL failą, galėtų generuoti DDL skriptą.

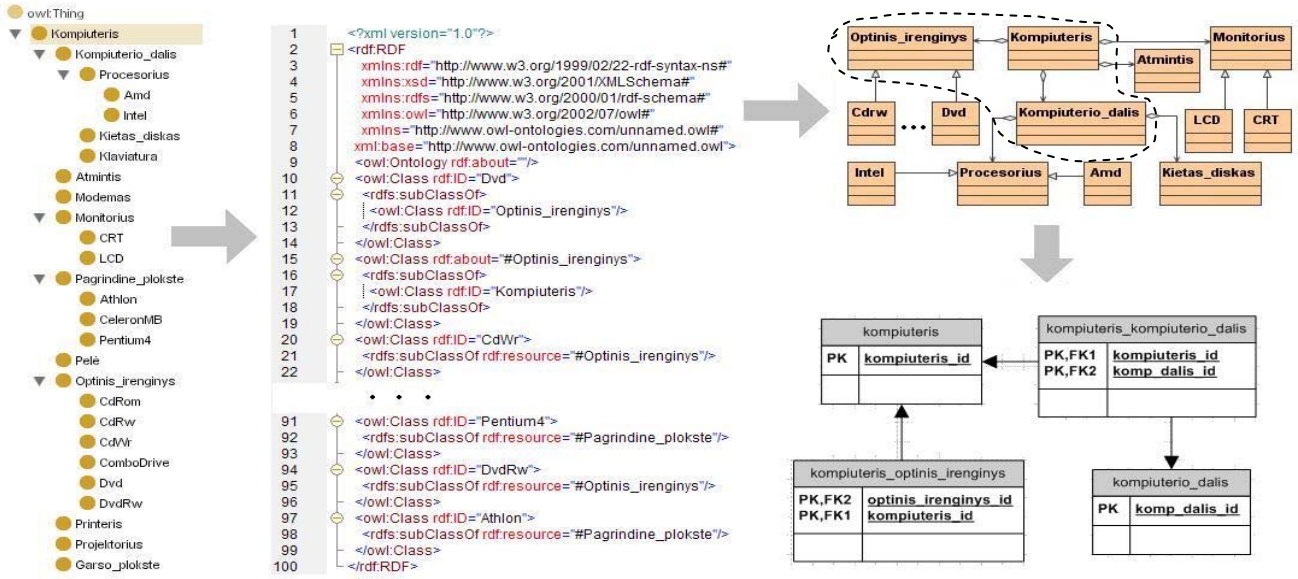
Anuradha Gali [2] pasiūlė požiūrį, kaip sukurti sistemą, kuri atvaizduotų OWL ontologiją į reliacinių duomenų bazių schemas, išsaugodama apribojančią informaciją. Sistema susideda iš trijų dalių – ontologijų modeliavimo, dokumentų valdymo ir ontologijų resursų valdymo.

Ontologijų modeliuotojas paima OWL dokumentus kaip įeigą ir sukuria ontologijos modelį, kuris yra panašus į DOM (angl. *Document Object Model*), naudojamą XML dokumentuose. Nagrinėjant OWL dokumentus visi apribojimai turi būti aptikti ir išsaugoti. Dokumentų valdytojas padeda apdoroti OWL dokumentus. Jis sukuria ryšius tarp ontologijos modelio elementų ir įkeltų dokumentų, taip pat naujus modelius įkeltiems dokumentams. Ontologijų resursų valdytojas suteikia metodus, skirtus resursų tipų gavimui ir nustatymui. Savybė rdf:type apibrėžia daugybę įvairių ontologijų kalbų semantinių modelių taisyklių.

Konfigūracijos ontologijų atvaizdavimui į reliacinę duomenų bazę bus naudojamas [2] pasiūlytas OWL2DB algoritmas, kuris atvaizduoja OWL dokumentus į reliacinės duomenų bazės lenteles tokiu būdu:

1. Pirmiausia nagrinėjamas OWL failas ir nustatomos šakninės (angl. *Root*) klasės. Grąžinamas rezultatas yra šakninių klasių skaičius.
2. Atliekama iteracija per visas šaknines klases, jos metu nustatomas šakninių klasių palikuonių gylis.
3. Iškviečiamas ShowClass metodas, kuris tikrina *i*-tos poklasės gylį šaknies atžvilgiu. Kai grafo gylis pasiekia 3, sukuriamos reliacinės lentelės visoms poklasėms, esančioms virš *i*-tosios. Atributų vardai/egzemplioriai tampa stulpelių vardais.
4. Ankstesnis žingsnis kartojamas pradedant nuo paskutinės poklasės ir einama skersai grafo tol, kol visoms poklasėms, esančioms „3“ gylyje nuo pradinio paieškos mazgo, surenkami lentelių ir atributų vardai. Tuomet pereinama prie žemesnio lygio ir kartojama tol, kol atvaizduojamos visos klasės.
5. Klasėms, kurios yra nesikertančios (disjoint), sukuriami atskiri ryšiai su tėvine klase. Nesikertančios yra tos klasės, kurios yra tame pačiame lygyje tačiau viena kitai priešingos, pavyzdžiui: daiktas:valgomas ir daiktas:nevalgomas.
6. Kai lentelių vardų ir atitinkamų atributų informacija nustatyta, reikia surinkti egzempliorių reikšmes ir užpildyti duomenų bazę. Grafas vėl yra nagrinėjamas, išrenkamos reikšmės, kurios saugomos duomenų struktūrose.
7. Galiausiai prisijungiama prie duomenų bazės, sukuriamos visos lentelės ir įrašomos reikšmės.

Taigi šio darbo procesas susideda iš produkto konfigūracijos ontologinio modelio sudarymo, modelio formalaus aprašymo OWL kalba, bei konfigūracijos aprašo transformavimo į reliacinę duomenų bazę. Paprasčiausias modelis, aprašantis asmeninio kompiuterio konfigūraciją, pavaizduotas 4 pav. ***Ontologinio modelio transformavimas į reliacinę duomenų bazę***.



4 pav. Ontologinio modelio transformavimas į reliacinę duomenų bazę (parodytas tik DB schemos fragmentas)

7. Išvados

Straipsnyje buvo apžvelgtos problemos, susijusios su produkto konfigūracijos modelio sudarymu, apibrėžta kas yra produkto konfigūravimo žinių bazės modeliavimas, nustatyti konfigūracijos modelio sudarymo proceso etapai ir transformavimo į reliacinę duomenų bazę metodas. Analizės metu padarytos tokios išvados:

1. Ontologijos yra perspektyvus ir įgyvendinamas būdas patobulinti ir supaprastinti produktų konfigūracijos kūrimą ir jų palaikymą viso produkto gyvavimo ciklo metu. Ontologijos suteikia pastovią produktų terminologiją, taip pat suteikia galimybę generuoti įvairius tų pačių produktų vaizdus priklausomai nuo esančio konteksto.

2. Išnagrinėjus literatūros šaltinius galima daryti išvadą, kad šiuo metu nėra standartų, kaip konstruoti konfigūracijos žinių bazę remiantis ontologijomis. Tačiau yra pavyzdžių, kuriais galima remtis kuriant savo konfigūracijos žinių bazės konstravimo ciklo modelį.

3. Analizės metu sukurtas veiklos proceso modelis, kurio metu iš probleminės srities analitiko pateiktų produktų specifikacijų gaunama produktų konfigūracijos žinių bazė, naudojama taikomųjų programų.

4. Produktų konfigūravimo ontologijas ir pačių produktų informaciją tikslinga saugoti reliacinėje duomenų bazėje, kuri užtikrina korektišką duomenų saugojimą, atnaujinimą ir manipuliavimą. Pateiktas algoritmas, kuriuo galima atlikti OWL kalba aprašytos produkto konfigūracijos ontologijos transformavimą į reliacinėje DBVS saugomą konfigūratoriaus žinių bazės schemą.

5. Ontologijų naudojimas informacinėse sistemose yra perspektyvus ir gana naujas dalykas, panašių taikomųjų programų nepavyko aptikti, todėl atlikta analizė duoda vertingas gaires tolimesniems praktiniams taikymams ir tyrimams sprendžiant konfigūravimo ir panašių sričių problemas.

Literatūros sąrašas

[1] Hvam, Lars. A multi-perspective approach for the design of product configuration systems – an evaluation of industry applications. In: Conference Proceedings, International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems, June 28.-29. 2004, Technical University of Denmark.

[2] Anuradha Gali, Cindy X. Chen, Kajal T. Claypool, Rosario Uceda-Sosa. From Ontology to Relational Databases. In: International Conference on Conceptual Modeling ER 2004: 278-289

[3] Lothar Hotz, Thorsten Krebs. Configuration – State of the art and New Challenges. In: Workshop "Planen, Scheduling und Konfigurieren, Entwerfen 2003", 15-18.09.2003, Hamburg, Germany.

[4] Pena, N., Garcia, R., Lazaro, J.M. Configuration Ontology & Multi-product Configuration Tool, Part 1. In: Ontology-based electronic integration of complex products and value chains, 2003.04.30.

[5] Felfernig, A., Friendrich, G.E., Jannach, D. UML as domain specific language for the construction of knowledge-based configuration systems. In: International Journal on Software Engineering & Knowledge

Engineering (Best Papers from SEKE'99), Vol. 10(4), 2000, pp. 449-470, World Scientific Publishing Ltd., 2000.

[6] Hotz, L., Krebs, T. Supporting the product derivation process with a knowledge-based approach. In: ICSE 2003: WS SVM, 03.-11.05.2003 , Portland, Oregon, USA.

[7] Krebs,T., Hotz, L., Ranze, H., Vehring, G. Towards evolving configuration models. In: Workshop "Planen, Scheduling und Konfigurieren, Entwerfen 2003", 15-18.09.2003, Hamburg, Germany.

ONTOLOGY MODELLING FOR PRODUCT CONFIGURATION KNOWLEDGE BASE

Ernestas Vyšniauskas, Lina Nemuraitė

The current paper is addressed to the importance of knowledge base modelling in product configuration system development. The performed analysis of knowledge modelling methodologies revealed that it is desirable to base knowledge model on Web Ontology Language OWL. From the other side, the relational database technology ensures the best facilities for storage, update and manipulation with product information. The process and methodology for configuration ontology development is proposed where ontology is transformed and stored in relational database.