KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Gediminas Sakalauskas

Duomenų semantiniame pasauliniame tinkle agregavimo metodas panaudojant ontologijas

Magistro darbas

Darbo vadovas:
prof. R. Butleris

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Gediminas Sakalauskas

Duomenų semantiniame pasauliniame tinkle agregavimo metodas panaudojant ontologijas

Magistro darbas

Vadovas: prof. R. Butleris
2007-01

Konsultantas: dokt. V.Taujanskas
2007-01

Recenzentas: dr. S.Maskeliūnas
2007-01

Atliko:
IFM-1/4 gr. studentas
Gediminas Sakalauskas
2007-01-08

Kaunas, 2007

Turinys

Lentelių turinys

Paveikslų turinys

Method of Semantic Web Data Aggregation Employing Ontology

Summary

The idea of this work is to propose a method of Web data aggregation employing ontology. This method is required as a part of major method proposed by Vytautas Taujanskas and Rimantas Butleris in paper "*Categories extraction for reuse in semantic applications and profile based recommendation service*" [1]. The main task of the method aims to semantic portal users who request to get information according their profile settings based and described with meaningful semantics.

The requirements of method are created based on requirements of Semantic Web implementation and Semantic Web Services concepts.

The vision of Semantic Web, Web Ontology Language, Resource Description Framework, Semantic Query Language for RDF, Ontology tools and frameworks issues are discussed. Also the main techniques of current Web for data syndication, aggregation and publishing in various feeds formats (Atom, RSS) among users are introduced.

The proposed method integrates available techniques and creates possibility of aggregated data storing in ontology based databases. The data represented by ontology can be re-used by another system without loosing semantics and more flexible.

The project model of method implementation was created and implemented by using JAVA programming language combining available techniques described in analysis phase. The experiment with test data and use case tests shows that method can be implemented successfully with available tools as a part of Semantic Web application.

1. Įvadas

Semantinio pasaulinio tinklo (SPT) kaip naujos kartos pasaulinio tinklo poreikis atsirado kilus kokybiškesnio statinių bei dinaminių resursų valdymo poreikiui. Jis siūlomas kaip esamo pasaulinio tinklo išplėtimas, kuriame pateikiamiems duomenims, informacijai nurodoma formaliai apibrėžta prasmė – suformuojama informacija susijusi semantiniais ryšiais. Naujos kartos pasaulinio tinklo įgyvendinimu siekiama sudaryti galimybes automatinei kompiuterių tarpusavio sąveikai bendrai priimtos terminijos pagrindu. Siekiama įgyvendinimo panaudojant taikomosios srities siūlomas aprašymo kalbas, logikos formalizmus ir loginio išvedimo galimybes bei ilgą laiką analizuojamą semantinių tinklų teoriją.

Ontologijos, trumpai vadinamos tam tikros srities sąvokų visumos specifikavimu išreikštu pavidalu, aprašo semantiniame pasauliniame tinkle esančią informaciją. Ontologijų panaudojimas pasauliniame tinkle užrašant jas specialia kalba – tai tik vienas žingsnių link semantinio pasaulinio tinklo. Darbe apžvelgiamos ir analizuojamos su naujos kartos pasaulinio tinklo vizijos įgyvendinimu susijusios technologijos.

Analizuojami egzistuojantys sindikavimo (duomenų kaupimo siekiant standartizuotai publikuoti) ir agregavimo (standartizuotai publikuojamų duomenų surinkimo) metodai duomenų publikavimui pasauliniame tinkle, juose naudojamų duomenų specifikacijos. Detalizuojamas publikavimo formatų specifikacijų atvaizdavimo ontologijomis uždavinys. Praktikoje atliekant duomenų sindikavimą ir agregavimą dauguma atvejų tik pateikia taisykles publikuojamiems duomenims surinkti ir atvaizduoti. Šiame darbe pateikiamas būdas tuos duomenis surinkti išsaugant ontologijų struktūrose galimam tolimesniam pakartotiniam panaudojimui.

Metodo duomenų agregavimui poreikis ir pradiniai reikalavimai buvo iškelti Vytauto Taujansko straipsnyje „Categories Extraction for Reuse In Semantic Applications and Profile Based Recommendation Service“ [1] (principinė schema pateikta 2.2 skyriaus 10 paveiksle). Straipsnyje siūlomas portalas turėtų sukurti semantinio pasaulinio tinklo paslaugą, o tai iškelia sąlygas, kad metodas būtų įgyvendintas su SPT paslaugas realizuojančiomis technologijomis.

Atliekama su metodu susijusių įrankių analizė. Siekiant pasirinkti galimas priemones metodo realizacijai apžvelgiami duomenų sindikavimo formatų validavimo, sindikavimo ir agregavimo įrankiai, instrumentinės priemonės darbui su ontologijomis. Pastarosios lyginamos tiriant funkcionalumą pagal: programavimo sąsają; semantinių

užklausų generavimo, duomenų saugojimo galimybes, galimą panaudojamumą su kitais įrankiais ir rekomendacijas bei paplitimą tarp semantinių aplikacijų kūrėjų.

Pagal esamas technologijas ir jų panaudojimo galimybes darbe išskiriami esminiai duomenų siūlomo agregavimo metodo žingsniai, algoritmas pateikiamas 3.1 skyriuje. Toliau metodas detalizuojamas ir atliekamas sistemos realizuojančios duomenų agregavimo metodą projektavimas išskiriant panaudojimo atvejų, sekų ir būsenų diagramas. Pagal uždavinius apibrėžiami sistemos komponentai bei ryšiai tarp jų.

Realizuojant metodą pasirinktas vienas iš duomenų sindikavimo formatų parodžius, kad veiksmai su kitais formatais publikuojamais duomenimis bus atliekami analogiškai. Sukuriami eksperimentui reikalingi duomenys ir infrastruktūra pagal nustatytus reikalavimus. Eksperimentas su metodą realizuojančia sistema atliekamas pagal numatytą tvarką. Pateikiami eksperimento rezultatai ir metodo įgyvendinimo išvados parodo darbo aktualumą.

Išanalizuojamos problemos skirstomos į tris grupes: informacijos kaupimo ir paskirstymo; kategorizavimo; pateikimo vartotojui. Siūloma, kaip jas būtų galima išspręsti publikuojant duomenis pagal nustatytus standartus ir vėliau juos agreguojant šiame darbe pateiktu metodu.

2. Naujos kartos pasaulinio tinklo vizijos analizė

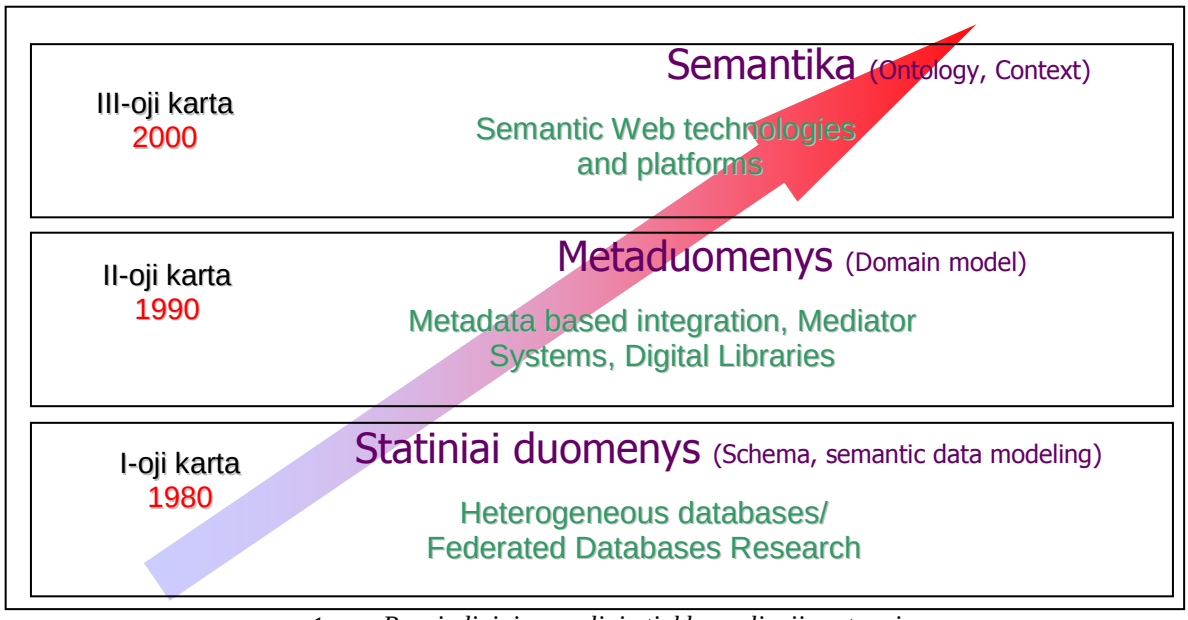
Darbo sritis siejama su semantiniu pasauliniu tinklu *(Semantic Web)* bei jį realizuojančiomis technologijomis. Apima kai kuriuos kategorizavimo uždavinio aspektus. Siejasi su informacijos kaupimu, paskirstymu ir pateikimu vartotojui.

Tolimesniuose skyriuose detalizuojamos dalykinės srities sudedamosios dalys.

2.1. Naujos kartos pasaulinį tinklą realizuojančių technologijų analizė

2.1.1. Semantinio pasaulinio tinklo (Semantic Web) kontekstas

Pasaulinio tinklo (PT) evoliucijoje galima išskirti tris pagrindinius raidos etapus pagal tai kokie duomenys ir technologijos buvo naudojami. Apibendrintai šie etapai pateikiami 1 paveiksle.



1 pav. Pagrindiniai pasaulinio tinklo evoliucijos etapai.

I-osios kartos pasauliniame tinkle (PT) buvo naudojami duomenys susieti pagal statinius ryšius. II-ojoje kartoje PT atsirado papildomi informacijos žymėjimai, padedantys specifikuoti duomenis. III-oji karta išsiskiria tuo, kad panaudojama semantika, kuri nusako semantinius ryšius tarp objektų ir suteikia jiems realaus pasaulio reikšmes.

Semantinis pasaulinis tinklas (SPT) – tai PT išplėtimas, kuriame duomenims, informacijai nurodoma formaliai apibrėžta prasmė – informacija susijusi semantiniais ryšiais. SPT skirtas automatinei kompiuterių tarpusavio sąveikai bendrai priimtos terminijos pagrindu, naudojant ontologijų aprašymo kalbas, logikos formalizmus ir loginio išvedimo

galimybes. Semantinių tinklų teorija pritaikoma pasauliniame tinkle esančių resursų aprašymams.

Semantinio pasaulinio tinklo poreikis atsirado kilus kokybiškesnio statinių bei dinaminių resursų valdymo poreikiui.

Semantinio pasaulinio tinklo terminą, pasiūlė Tim Berners-Lee [2] kaip ateities Interneto viziją. Pagal autorių Internetas turėtų tapti globalia duomenų baze – universalia terpe duomenų apsikeitimui. Siūloma infrastruktūra leistų tiek žmonėms, tiek mašinoms daryti sprendimus kaip panaudoti informaciją.

Semantinio pasaulinio tinklo sukūrimo tikslas – automatinė kompiuterių tarpusavio sąveika bendrai priimtos terminijos pagrindu, naudojant ontologijų kalbas, logikos formalizmus bei loginio išvedimo galimybes.

Idėjos įgyvendinimą prisiėmė W3C[1] konsorciumas, kuris iškėlė pagrindinius reikalavimus realizuoti XML bazėje, o semantinius ryšius užrašyti deklaratyvia RDF kalba.

Į SPT architektūrinius sprendimus įeina:

1. semantika (elementų pavadinimai),
2. struktūra (elementų hierarchija) ir
3. užklausų kalba (bendravimas).

Formalizuotos specifikacijos suteikia galimybes:

1. keistis tarpusavio informacija tiek tarp žmonių, tiek tarp mašinų [2];
2. apjungti tarpusavyje susijusį asmeninės informacijos valdymą, korporacijų programų integraciją bei globalios informacijos apsikeitimą tarp komercinių, mokslinių, kultūrinių ir kitokio pobūdžio programų., tai atliekant, be žmogaus įsikišimo.

Semantinį pasaulinį tinklą sudaro dvi pagrindinės dalys:

1. standartizuotas duomenų formatas bendram informacijos apsikeitimui (XML);
2. formalizuota kalba aprašanti kaip duomenys susiję su realiu pasauliu (RDF, OWL).

Sudarius taikomosios srities aprašą (ontologijas) bei panaudojus loginio išvedimo priemones, užtikrinančias galimybę manipuliuoti ontologijoje išreikštu taikomosios srities supratimu, sudaroma galimybė žmogui ar kompiuteriui nuosekliai perimti informaciją nagrinėjant semantinius ryšius tarp duomenų.

[1] http://www.w3.org

2.1.2. Ontologijų vaidmuo semantinio pasaulinio tinklo kontekste

Kas tai yra ontologija?

Ontologija – terminas kilęs iš filosofijos ir graikų kalboje reiškia „būtis“ kartu su „žodis“ ir „kalba“. Filosofijoje ši šaka nagrinėja būties ir egzistencijos klausimus, kategorijas, ryšius ir tipus pasireiškiančius būtyje. Kompiuterijoje šis terminas vartojamas ir daugiskaitoje: “ontologijos”. Trumpai tai vadinamas tam tikros srities sąvokų visumos specifikavimas išreikštu pavidalu *(angl. “explicit specification of a conceptualization“* T. R. Gruber, 1993) [3].

Kartais ontologija apibrėžiama, kaip tam tikros srities bendrai naudojamos sąvokų/konceptų, esybių tipų, jų tarpusavio priklausomybių, sąryšių, aksiomų, dėsningumų ir kt. visumos formalus aprašas.

Pats „ontologijos“ terminas pirmą kartą panaudotas XVII a. Maždaug nuo 1990 m. šis terminas pradėtas naudoti ir Dirbtinio intelekto srityse, kalbant apie bendrą žinių naudojimą, programinių agentų tarpusavio sąveiką, visuotinai pripažįstamų žinių atvaizdavimą, natūralios kalbos apdorojimą. Atsiradus semantinio pasaulinio tinklo idėjai, ontologijos pradėtos taikyti ir naudoti šiame kontekste.

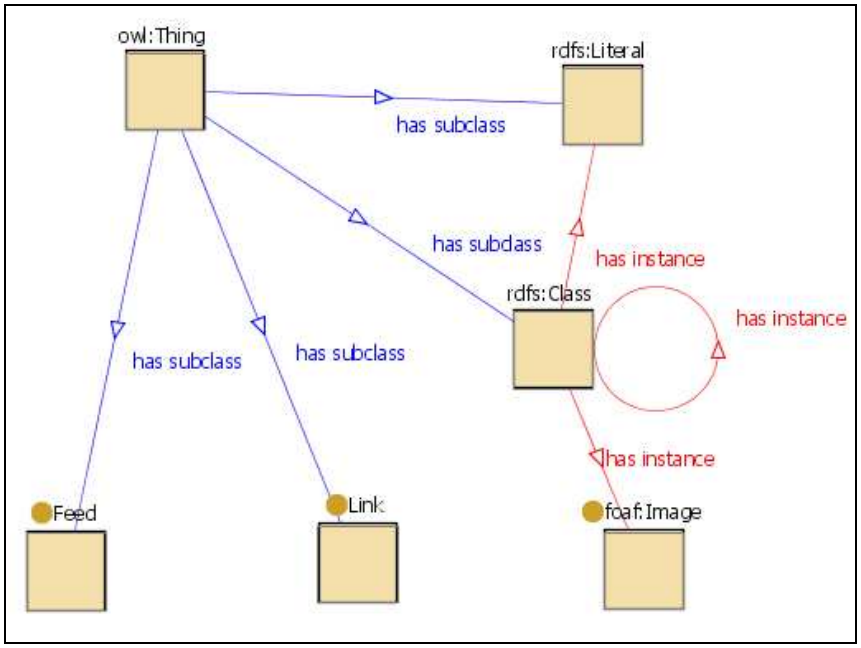
Ontologijos gali būti įvairios paskirties ir dažniausiai skirstomos į: žinių vaizdavimo ontologijas; bendrąsias ontologijas, visuotinai naudojamų sąvokų ontologijas; aukščiausio lygio ontologijas, meta–ontologijas; lingvistines ontologijas; nagrinėjimo sričių ontologijas; užduočių ontologijas, metodų ontologijas, taikomųjų programų ontologijas ir kt..

Ontologijos apibrėžia nagrinėjamos srities:

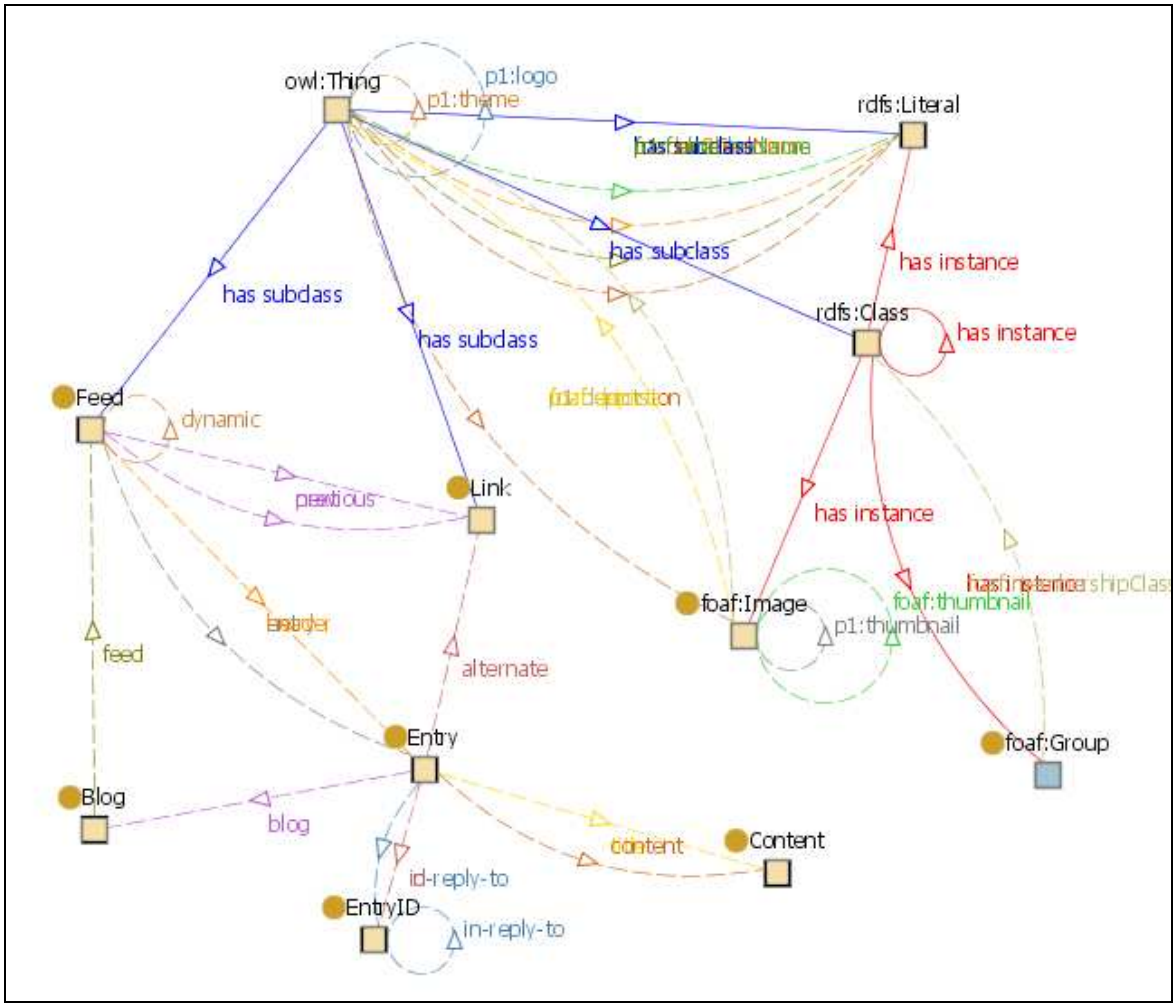
1. sąvokas, esybių (reiškinių, daiktų) tipus,
2. sąvokų hierarchijas, esybių tipų tarpusavio sąryšius, priklausomybes,
3. aksiomas, taisykles, dėsningumus apie esybių tipus ir sąryšius,
4. pavyzdinius atvejus.

Ontologijos pavyzdys su pagrindiniais elementais aprašančiais Atom 1.0 formato (žiūr. skyrių 2.1.5) struktūrą pavaizduotas 2 paveiksle.

2 paveiksle matome, kad ontologijos viršuje yra elementas iš ontologijų aprašymo kalbos `owl:Thing` (bendrinis pavadinimas – Daiktas). Šis elementas turi savybę vaikinę klasę iš resursų aprašymo kalbos `rdfs:Class` (bendrinis pavadinimas – Klasė), kurią susieja per savybę `has subclass`. Pastarasis elementas jau turi egzempliorių iš kitos ontologijos `foaf:Image`, kuris susietas per savybę `has instance`. Nagrinėjant Atom elementus kitos ontologijos nepriklausančios šiai rašomos su priešdėliu, pagal kurį buvo importuotos, o objektai priklausantys be – Feed, Link.



2 pav. Ontologijos pagrindiniai elementai aprašantys Atom 1.0 formato struktūrą



3 pav. Atom 1.0 formato ontologijos struktūra su maža ryšių ir objektų aibe

2 paveiksle pateikiami tik pagrindiniai elementai siejami su ontologijų kalbos baziniais komponentais. O 3 paveiksle pateikta jau didesnis grafas, atspindintis dalį visų objektų ir egzistuojančių ryšių tarp jų aprašomų pilnoje ontologijoje.

Darbo tema siejama su SPT, kurio vienas iš reikalavimų yra duomenų pateikimas ontologijomis. Todėl šiame darbe duomenis pasirinkta pateikti specialiai tam sudarytomis ontologijomis pilnai aprašančiomis formato struktūrą (žiūr. 2.4.1.2 skyrių).

Darbe nagrinėji ontologijų sudarymo, papildymo ir panaudojimo uždaviniai. Panaudojant instrumentinę priemonę darbui su ontologijomis, galima suformuoti duomenų grafą, saugoti duomenų bazėje, o vėliau formuojant semantines užklausas išgauti reikalingus duomenis su semantiniais ryšiais.

Šiame darbe ontologijos bus panaudojamos kuriant semantinio pasaulinio tinklo paslaugą (SPTP).

2.1.3. Technologijos įgyvendinant semantinį pasaulinį tinklą

Technologijos siūlomos, prižiūrimos ir plėtojamos W3C konsorciumo *(The World Wide Web Consortium)*, kuris prisiėmė SPT vizijos įgyvendinimo uždavinį. 2004 metų vasario mėnesį šis konsorciumas galutinai patvirtino rekomendacijas resursų aprašymo sistemos *(RDF)* ir ontologijos aprašymo kalbos *(OWL)* naudojimui.

I-osios kartos pasaulinis tinklas remiasi daugiausia hiperteksto žymėjimo kalba HTML *(Hypertext Markup Language)* parašytais dokumentais. HTML yra tinkama kalba aprašymui, akcentuojant vizualinį atkūrimą, t.y. struktūruoto teksto masyvas, į kurį įsiterpia multimedijos objektai (paveiksliukai, interaktyvios formos). Tačiau HTML turi ribotas galimybes klasifikuoti teksto blokus puslapyje, išskyrus, kai tai reikalinga dokumento struktūrai nustatyti ir suteikti jam norimą išvaizdą.

Pavyzdžiui, HTML kalba sukurtas internetinis katalogas gali būti lentelės formos, kuriame prekės numeris X586172 reiškia prekę „Acme Gizmo“, kainuojančią 199 Lt. Tačiau duomenys toje lentelėje nebus susieti niekaip kitaip kaip tik lentelės formatavimu. Tai reiškia, kad „X586172“, „Acme Gizmo“ ir „199 Lt“ HTML požiūriu yra tik vienas šalia kito esantys teksto gabaliukai. HTML taip pat negali tiksliai nurodyti, kad tas puslapis yra katalogas, o tik pavadinti jį kaip katalogą suteikiant puslapio pavadinimą.

Semantiniame pasauliniame tinkle siekiama šiuos trūkumus pašalinti naudojant aprašomąsias technologijas RDF ir OWL bei į duomenis orientuotą, lanksčią žymėjimo kalbą XML. Šios technologijos yra sujungiamos taip, kad būtų galima pateikti aprašymus, kurie papildytų arba pakeistų PT dokumentų turinį.

Pasaulinio tinklo technologijos aprašytos ir specifikuojamos pagrindiniame W3C konsorciumo puslapyje http://www.w3c.org. Žemiau pateiktos šiame darbe aktualios technologijos.

2.1.3.1. Universalus resursų identifikatorius URI[2]

Universalus resursų identifikatorius *(Universal Resource Identifier)* galioja pasauliniame tinkle ir sudaro pagrindą tiek pirmos kartos, tiek ir semantiniam pasauliniam tinklui. Jo pagalba nusakoma, kur yra resursai pasauliniame tinkle, o tai reiškia, kad bet koks objektas gali būti su juo susietas, nurodytas ir išgautas jo atvaizdavimas. URI atpažinimas yra standartizuotas ir plačiai naudojamas. URI yra pagrindas ontologijų aprašymo kalbose identifikuojant resursus.

2.1.3.2. Resursų aprašymo sistema RDF[3]

W3C konsorciumas pradines specifikacijas resursų aprašymo sistemai aprašė dar 1997-aisiais.

I-osios kartos pasaulinis tinklas panaudojo hipertekstą duomenims atvaizduoti, o RDF buvo sukurta siekiant ne tik atvaizduoti, bet ir suteikti semantinius ryšius tarp duomenų.

RDF suteikia paprastą, bet efektyvų būdą aprašyti resursus (duomenis) remiantis universaliu resursu identifikatoriumi URI. RDF formatas tapo esmine konsorciumo rekomendacija 1999-aisiais, kai buvo pasiūlyta naudoti praplečiant PT funkcionalumą ir vientisumą.

Pagrindiniai konceptai naudojami RDF:

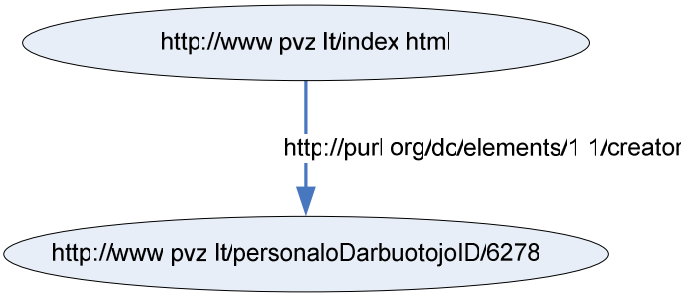
1. Grafo tipo duomenų modelis – duomenys patalpinami į grafo struktūrą, kurioje juos aprašo objektai susieti su reikšmėmis per prasmę turintį ryšį.
2. URI paremtas žodynas – visi resursai grafo struktūroje identifikuojami URI.
3. Duomenų tipai – apibrėžta tipų aibė, kuri gali būti papildoma esant poreikiui.
4. Raidės – paprasčiausia eilutė sauganti resurso reikšmę (gali būti tipizuota).
5. Į XML verčiama sintaksė – struktūra praplečiama, tačiau suderinama su XML standartais ir validuojama pagal tuos pačius principus.
6. Paprastų faktų išreiškimas – struktūra išlaikoma kaip galima paprastesnė.
7. Paveldėjimas – leidžia sudaryti duomenų hierarchijas, esamų duomenų hierarchijų praplėtimą ir pakartotinį panaudojimą.

[2] http://www.w3.org/Addressing/
[3] http://www.w3.org/RDF/

RDF suteikia galimybę aprašyti objektus naudojant URI ir suteikiant jiems prasmingus ryšius (semantiką) aprašant jų savybes su tam tikromis reikšmėmis. Tai leidžia aprašyti resursus kaip grafą sudarytą iš viršūnių ir lankų tarp viršūnių.

Grafo viršūnės atitinka resursus arba reikšmes, o lankai resurso savybę. Lankas sujungia resursą pagal jo savybę su atitinkama reikšme.

Pats paprasčiausias modelis būtų, kai turime objektą, jo savybę ir reikšmę, taip kaip pateikta paveiksle 4. Čia objektas yra pradinis PT svetainės puslapis (http://www.pvz.lt/index.html), kurio savybė yra kūrėjas (http://purl.org/dc/elements/1.1/creator), o kūrėją identifikuoja tam tikra reikšmė esanti nuorodoje (http://www.pvz.lt/personaloDarbuotojoID/6278).



4 pav. Paprasčiausias RDF grafas

Nagrinėjant RDF grafą geriausia kalbėti apie trijulę *(s, p, o)*, kurią sudaro:

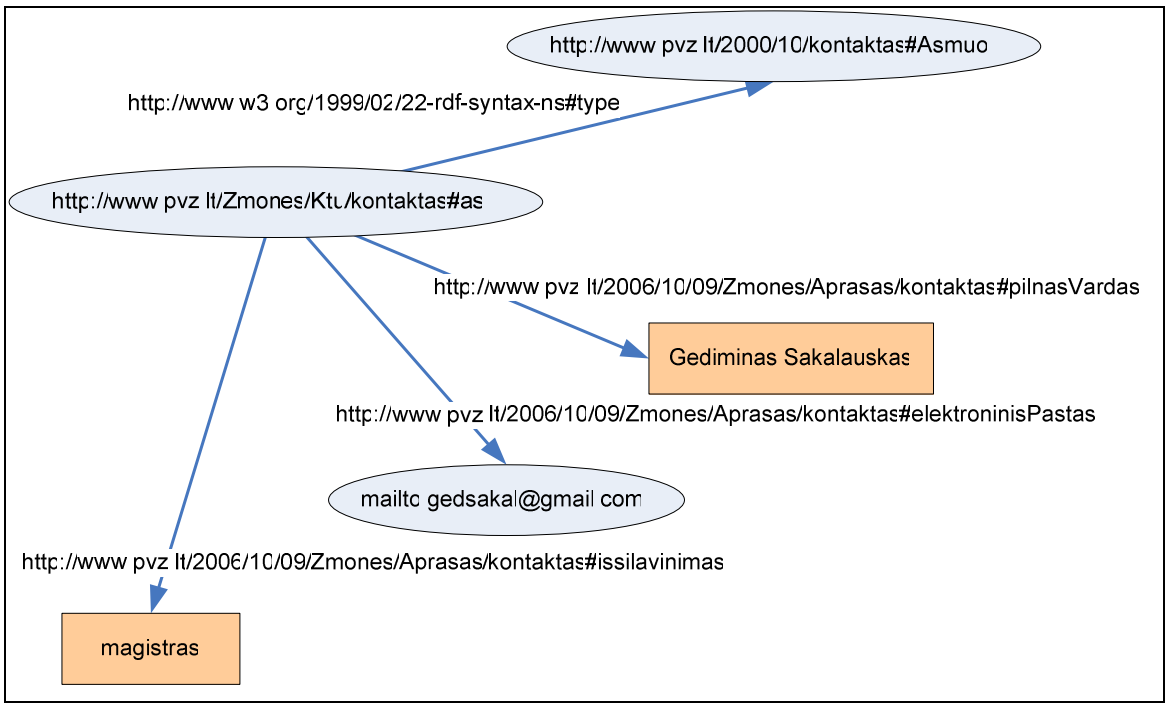
1. aprašomas subjektas (resursas) – *subject*
2. predikatas (savybė, atributas) – *predicate*
3. objektas (duomenys) – *object*

Resursas ir objektas grafe atitinka viršūnę, o lankas atitinka savybę. Viršūnė gali būti identifikuojama URI arba turėti specialų duomenų tipą ir saugoti reikšmę.

Turint tokią trijulę, turime dalį semantikos, kurią galime interpretuoti ir įvairiai pateikti. Pvz.: Tinklapis http://www.pvz.lt/index.html yra sukurtas autoriaus http://www.pvz.lt/personaloDarbuotojoID/6278.

Interpretavimo ir nuskaitymo taisykles galima parinkti ir sudaryti įvairiai nepriklausomai nuo duomenų.

Paveiksle 5 pateiktas pavyzdys analogiškas W3C RDF Primer (www.w3.org/TR/rdf-primer), atvaizduojantis resursą pavadintą „Gediminas Sakalauskas“.



5 pav. RDF grafo pavyzdys aprašantis tinklapio autorių

Grafą sudaro viršūnės ir lankai jungiantys viršūnes. Lankai ir viršūnės identifikuojančios resursus aprašomos universaliu resursų identifikatoriumi URI.

Paveiksle 5 pavaizduota:

- Objektas (individas) – žmogus aprašytas tarkime KTU adresų knygutėje, kaip kontaktas ir identifikuotas URI: http://www.pvz.lt/Zmones/Ktu/kontaktas#as
- Objekto tipas (rūšis) – žmogus, kaip asmuo ar asmenybė identifikuotas URI: http://www.pvz.lt/2000/10/kontaktas#Asmuo
- Objekto (individo) savybė – elektroninis paštas, identifikuotas URI: http://www.pvz.lt/2006/10/09/Zmones/Aprasas/kontaktas#elektroninisPastas
- Objekto (individo) reikšmės – elektroninio pašto adresas mailto:gedsakal@gmail.com, arba tiesiog vardas ir pavardė užrašyti simboliais “Gediminas Sakalauskas”

RDF sistema paprastai yra užrašoma remiantis išplečiama bendros paskirties duomenų struktūrų bei jų turinio aprašomąja kalba *(eXtensible Markup Language)* XML – vadinama XML/RDF.

XML/RDF formatu užrašytas fragmentas atspindintis paveiksle 5 pateiktą grafą:

```
<?xml version="1.0"?>
<rdf:RDF          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                  xmlns:kontaktas="http://www.pvz.lt/Zmones/Ktu/kontaktas#">
        <kontaktas:Asmuo          rdf:about="http://www.pvz.lt/Zmones/Ktu/kontaktas#as">
                  <kontaktas:pilnasVardas> Gediminas Sakalauskas </kontaktas:pilnasVardas>
                  <kontaktas:elektroninisPastas       rdf:resource=mailto:gedsakal@gmail.com  />
```

```
                <kontaktas:issilavinimas> magistras </kontaktas:issilavinimas >
        </kontaktas:Asmuo>
</rdf:RDF>
```

XML/RDF yra tik viena ne itin vaizdinga užrašymo forma. Užrašant grafą galima naudoti ir kitas notacijas, tokias kaip pavyzdžiui N3[4], kuri orientuojasi į trijulės užrašymą. Notacija ir užrašymas nedaro įtakos pačiam RDF duomenų organizavimo principui. Reikia, atsiminti, kad visą laiką operuojame duomenų grafais.

Kita labai svarbi RDF savybė yra ta, kad aprašyti grafai gali būti panaudojami kituose aprašuose, taip praplečiant ir papildant jų objektų struktūrą, savybes ir ryšius tarpusavyje.

Įvedant skirtingas vardų sritis (angl. *namespace*) galima naudoti objektus ir savybes tame grafe. Pavyzdyje žemiau pateikta, kaip bus naudojama rdf, owl, foaf, dc ir kitų struktūrų RDF modeliai.

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:vs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
</rdf:RDF>
```

2.1.3.3. RDF saugyklos ir užklausų kalba SPARQL[5]

Įsitvirtinus duomenų aprašymui RDF grafais, atsirado poreikis tokio tipo duomenų saugykloms. RDF grafe aprašoma trijulė susidedanti iš subjekto, predikato ir objekto, gali būti saugomo įvairiose terpėse.

SPARQL yra nepriklausoma nuo programavimo technikos užklausų kalba įgalinanti, išgauti RDF grafo poaibius pagal nurodytus parametrus.

Pagrindiniai uždaviniai sprendžiami panaudojant SPARQL:

1. Informacijos išgavimas URI, tuščių viršūnių ar simbolių aibės pavidalu;
2. RDF grafo poaibių išgavimas;
3. Naujų RDF grafų formavimas, pagal užklausos parametrus.

SPARQL užklausų kalba remiasi grafo šablono aptikimo principais. Pats paprasčiausias grafas yra trijulė – dvi viršūnės ir lankas tarp jų – būtent taip, kaip RDF sandara. Kombinuojant grafo šablonus suformuojama rezultatų aibė.

[4] http://www.w3.org/DesignIssues/Notation3.html
[5] http://www.w3.org/2001/sw/DataAccess/

Žemiau pateiktame pavyzdyje matome duomenis, užklausą ir rezultatus. Duomenys buvo aprašyta objektas „knyga1“, kuriam buvo suteikta savybė „title“ su reikšme „Semantinis pasaulinis tinklas“. Užklausa buvo išgauti visas objekto „knyga1” reikšmes turinčias savybę „title“ reikšmes.

Duomenys:

```
<http://www.pvz.lt/knygos/knyga1> <http://purl.org/dc/elements/1.1/title> “Semantinis pasaulinis tinklas“
```

Užklausa:

```
SELECT ?title
WHERE
{
  <http://www.pvz.lt/knygos/knyga1>     <http://purl.org/dc/elements/1.1/title>     ?title
}
```

Rezultatai:

title
“Semantinis pasaulinis tinklas“

Rezultatai pačios užklausos suformavime gali būti rikiuojami, apriboti, grupuojami ir pateikiami skirtingomis formomis.

2.1.3.4. Semantinio pasaulinio tinklo ontologijų kalba OWL[6]

Egzistuoja uždavinių, kuriems išspręsti nepakanka grafu aprašytos struktūros, bet yra reikalingas didesnis išraiškingumas. Ontologijų kalba OWL praplečia RDF suteikdama galimybę aprašyti taisykles ir apribojimus. Pagrindinė idėja yra suteikti galimybę efektyviai atvaizduoti ontologijas, patikrinti jų loginį teisingumą, nuoseklumą ir kt.

OWL papildo savybėms ir klasėms naudojamą žodyną. OWL pagalba galima aprašyti tarp klasių atsirandančius ryšius, nusakyti kardinalumą bei ekvalentiškumą, pilniau aprašyti turinį, savybių charakteristikas.

Kalbą pagal sudėtingumą ir galimybes priimta skirstyti į tris lygius:

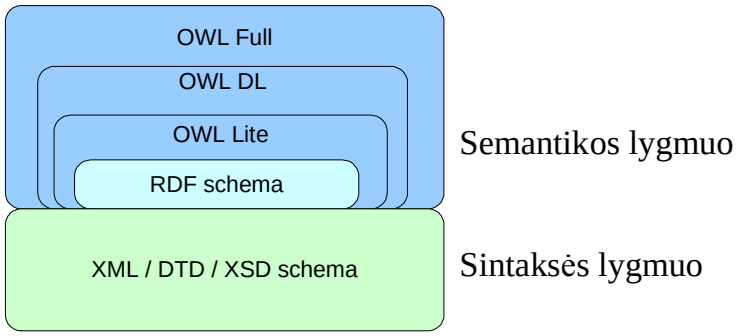
1. OWL Lite – palaiko pirmines vartotojui reikalingas klasifikavimo ir kardinalumo aprašymo galimybes. Naudojama su paprastesniais įrankiais. Lengvai pritaikoma tezaurams ir kitoms taksonomijoms.
2. OWL DL – *(DL – description logics)* palaiko maksimalų išraiškingumą tuo pačiu metu užtikrinant ribotą skaičiavimų trukmę. Šioje kalboje įtraukiamos visos OWL numatytos loginės struktūros, tačiau jos negali būti naudojamos tam tikromis sąlygomis.

[6] http://www.w3.org/2004/OWL/

3. OWL Full – skirta maksimaliam išraiškingumui ir bet kokiai sintaksinei laisvei, tokiu būdu negarantuojant, kad skaičiavimo trukmės bus baigtinės.

OWL aprašymai formuojami pagal tokius pačius principus kaip ir RDF (objektas - savybė - predikatas), tokiu būdu suteikiant galimybę ontologijas paskirstyti ir tarp sistemų ir panaudoti semantinio pasaulinio tinklo kūrimui [4].

OWL kaip ir RDF užrašoma XML sintakse, o tai įgalina laisvai keistis duomenimis Internete.



6 pav. Semantinio pasaulinio tinklo sintaksės ir semantikos lygmenys

2.1.3.5. Instrumentinės priemonės darbui su ontologijomis

Vystant RDF ir OWL aprašymo kalbas, tobulinant technologijas darbui su ontologijomis atsirado poreikis turėti pagrindines operacijas atliekančius įrankius. Siekiant unifikuoti jų funkcionalumą ir principus kuriamos instrumentinės priemonės (angl. *framework*), kurių pagalba galima atlikti pagrindinius veiksmus: sukurti ontologijos modelius, išsaugoti, nuskaityti, formuoti užklausas, valdyti duomenis ir ontologijų elementus.

Instrumentinių priemonių panaudojimas leidžia paslėpti neišvaizdžią RDF sintaksę, atliekant operacijas su objektais. Užklausoms formuluoti nereikia rašyti sudėtingos SPARQL sintaksės, kuri paslepiama po programavimo sąsaja ir leidžia sudaryti logikos sluoksnius.

Analizė pateikta [5] publikacijoje 2002-aisias metais išskiria 14 labiausiai vystomų įrankių saugojimui ir išgavimui: ICS-FORTH RDFSuite; Sesame; Inkling; rdfDB; RDFStore; Extensible Open RDF (EOR); Redland; Jena; RDF Gateway; TRIPLE; KAON Tool Suite; Cerebra®; Empolis K42; Ontopia Knowledge Suite. Dalis jų iki šių dienų jau nebevystomi, kiti ištobulėjo ir išsiplėtė iki instrumentinių priemonių. Šiame darbe panaudojama instrumenė priemonė aprašoma sprendimo analizėje (žiūr. 2.4.4 skyrių).

2.1.4. Semantinio pasaulinio tinklo paslaugas apibendrinanti schema

Ontologijų sudarymas ir valdymas yra sudėtingas procesas, nes egzistuoja skirtingi metodai ir požiūriai. Dėl diskusijų apie ontologijų vaidmenį semantiniame pasauliniame tinkle jų panaudojimas su visomis galimybėmis praktikoje vėluoja. Besivystant pasauliniam

tinklui vartotojų poreikiai kristalizavosi ir susikūrė naujo tipo paslaugos vadinamos pasaulinio tinklo paslaugomis (PTP). PTP sampratą straipsnyje apibrėžia Saulius Maskeliūnas [6].

Kitos paslaugos susijusios su socialinėmis sistemomis vadinamos – folksonomijomis *(angl. Folksonomies).* Jos remiasi duomenų sužymėjimu, organizavimu ir kategorizavimu siekiant centralizuoti informaciją. Informacija išgaunama automatiškai bendradarbiaujant Internete pagal susitartas taisykles. Siekiama kategorizuoti pasaulinio tinklo puslapius, fotografijas, nuorodas ar kitus PT resursus pagal autorių pasiūlytas ir sudarytas portalų taksonomijas.

Verta išskirti šias folksonomijomis paremtas sistemas:

- flickr[7] – fotografijų publikavimo ir apsikeitimo sistema, pasaulinio tinklo paslaugų rinkinys ir internetinės bendruomenės platforma, kuri dažnai cituojama kaip sėkmingų naujos kartos pasaulinio tinklo technologijų pavyzdys.
- del.icio.us[8] – žymeklių *(angl. bookmarks)* kategorizavimo ir apsikeitimo sistema. Pasaulinio tinklo paslauga skirta įtraukti internetines nuorodas į taksonomiją, pagal vartotojo aprašymą.
- kaip.tik.ten[9] – lietuviškoji žymeklių kategorizavimo ir apsikeitimo sistemos del.icio.us versija.

Kita vertus egzistuoja dideli projektai medicinoje, bibliotekų sistemose, inžinerijoje naudojantys ontologijas specializuotos srities aprašymams. Dažnai tokie projektai naudoja skirtingus ontologijų formavimo principų ir yra uždari. Tačiau organizuojant suderinamumą su W3C korporacijos paskelbtais SPT reikalavimais naudotiniems protokolams ir standartams būtų įmanomas jų panaudojimas ir įliejimas į SPT kaip bendrą visumą.

Egzistuoja kiti projektai, kurie daugiau orientuoti į sąlyginai siaurų ontologijų aprašymą, su tikslu jas panaudoti bendrai semantiniame pasauliniame tinkle. Keletas projektų pavyzdžių, kurie naudoja ontologijas siauros srities aprašymams:

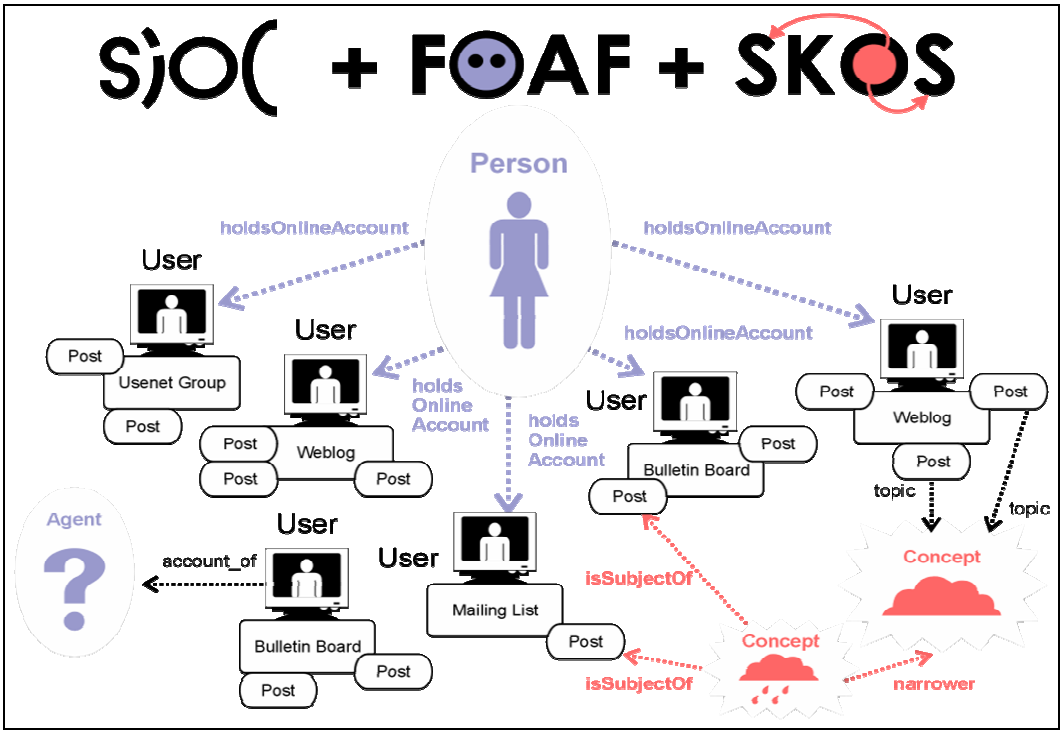
- skos[10] – supaprastintas žinių apjungimas susiejant elektronines bendruomenes, tezaurus, klasifikavimo sistemas, taksonomijas, antraštes, žodynus ir kt.
- foaf[11] – projektas apima draugų ontologijos sudarymą, saugojimą su įvairiais atributais ir valdymą.
- sioc[12] – apjungia diskusijas, dienoraščius, elektroninius susirašinėjimus ir kitus socialines bendruomenes.

[7] http://www.flickr.com
[8] http://del.icio.us
[9] http://around.puslapiai.lt/
[10] http://www.w3.org/2004/02/skos/
[11] http://www.foaf-project.org
[12] http://sioc-project.org

Vartotojų informacija aprašyta su galimybe praplėsti atributus ir panaudoti kitoms semantinėms aplikacijoms. Visos jos sukurtos ontologijų pagrindu. Jos formuoja bazinių ontologijų rinkinį.

Populiariausios bazinės ontologijos: foaf, vcard, dublin core ir kt..

Pavyzdys apjungiantis keletą sistemų ir panaudojantis bazines ontologijas pateiktas 7 paveiksle, kiti pavyzdžiai publikacijoje [7].



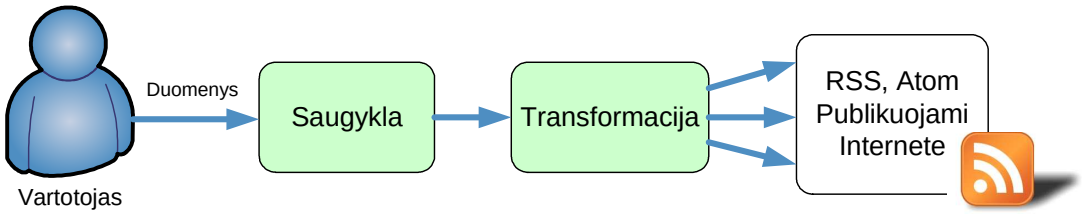
7 pav. Semantinių aplikacijų apjungimo modelis panaudojantis bazines ontologijas

Šiame darbe numatoma panaudoti esamas bazines ontologijas, duomenų transformavimą ir saugojimą ontologijomis. Norint pateikti semantinio pasaulinio tinklo paslaugą ontologijos yra neatskiriama proceso dalis. Schema ir detalesnis aprašymas pateikiamas projektavimo etapuose (žiūr. 3.1.4 skyrių).

2.1.5. *Duomenų sindikavimas ir agregavimas*

Sindikavimas – kitaip vadinamas duomenų surinkimas – ypač aktualus semantinio pasaulinio tinklo kontekste. Jis skirtas automatiniam duomenų apsikeitimui tarp svetainių bei pakartotiniam panaudojimui. Toks unifikuotas duomenų surinkimo procesas yra automatizuotas naudojant pasaulinio tinklo pranešimų formatus: RSS, Atom ar kitus naujai atsirasiančius formatus.

Plačiausiai paplitęs scenarijus ir panaudojimas duomenų sindikavimui pateiktas 8 paveiksle.

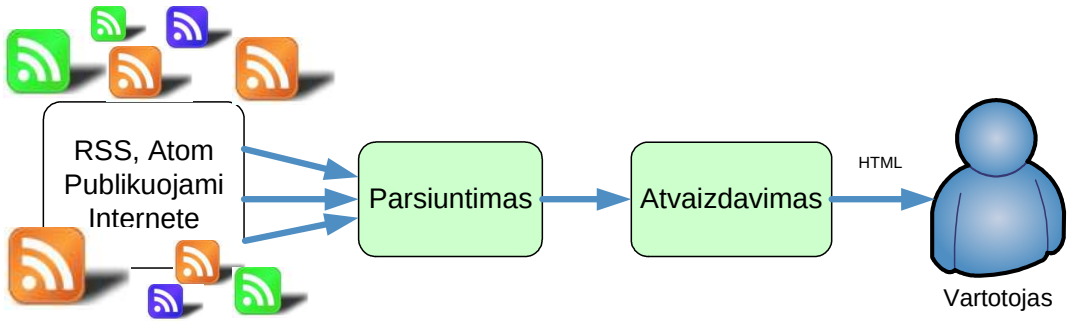


8 pav. Duomenų sindikavimo scenarijus transliavimui Internete

Šiame scenarijuje vartotojas arba aplikacija pildo saugyklą duomenimis. Tos saugyklos duomenis aplikacija transformuoja ir duomenis pateikia Internete sindikavimo formatais RSS, Atom ar kitais.

Agregavimas – duomenų surinkimas iš pasauliniame tinkle pateikiamų sindikuotų duomenų.

Dažniausiai pritaikomas scenarijus pavaizduotas 9 paveiksle.



9 pav. Internete transliuojamų duomenų agregavimo scenarijus

Šiame scenarijuje RSS, Atom ar kitais sindikavimo formatais (žiūr. 2.1.6 skyrių) Internete publikuojami duomenys parsiunčiami ir atvaizduojami vartotojui suprantama forma.

2.1.6. *Sindikavimo formatų duomenų šaltiniai semantinėms aplikacijoms*

Vienas iš esminių semantinių aplikacijų bruožų yra tai, kad duomenys pateikiami ontologijomis aprašomomis RDF schemomis arba OWL kalba. Dažniausiai apsikeitimui naudojamas XML formato resursų failai. Tobulinant informacijos apsikeitimo informacija

procesą ir siekiant jį valdyti buvo suformuota speciali XML failų formatų šeima – vadinama sindikavimo formatais. Pradinis projektas pavadintas „gausiu tinklalapių/Interneto svetainių anotavimu“ *(angl. Rich Site Summary)* sutrumpintai – RSS. Pagrindinis šio formato tikslas – duomenų rinkimas (sindikavimas) iš Interneto naujienų portalų. RSS pateiktas turinys gali būti panaudotas kitų aplikacijų, servisų kaip standartizuoti duomenys turintys tam tikrą semantiką ir pritaikomumą SPT.

Skirtingų versijų RSS veikia pagal tuos pačius principus ir suteikia galimybę atitinkamos programos pagalba automatiškai sekti naujų straipsnių norimose rubrikose publikavimą neužeinant į Interneto puslapius ar kitus įvairius Interneto informacijos šaltinius. Pvz., galima surinkti (agreguoti) ir susisteminti periodiškai atnaujinamą naujausią įvairiapusišką informaciją, naujienas išskiriant atributus: straipsnio pavadinimas; jo trumpas aprašymas (anotacija); nuoroda į straipsnį Internete bei publikavimo data.

Atom – kitas duomenų sindikavimo formatas, kurio idėja ir principai sugalvoti analogiškai RSS formatui. Atom yra RSS formato pasekėjas papildantis funkcionalumą, praplečiantis galimybes. Atom formate išsaugotos gerosios RSS savybės, ištaisytos klaidos, kurios naudojant praktikoje buvo aptiktos arba įrodytos ydingomis.

Poreikis šio formato vystymui atsirado dėl nepasitenkinimo RSS formatais. RSS nebuvo atviro kodo – visos kūrėjų teisės priklausė Harvardo Universitetui. RSS versijų įvairovė ribojo galimybes jį plėsti bei panaudoti, o vystymo darbai buvo sustabdyti. Atom buvo pradėtas vystyti su iniciatyva palengvinti duomenų sindikavimo ir agregavimo aplikacijų kūrimą sukuriant duomenų apsikeitimui skirtą protokolą.

Toliau pateikiama detalesnė Atom 1.0 formato bei šį formatą aprašančių ontologijų analizė.

2.1.7. Atom 1.0 formato istorija ir panaudojimas duomenų sindikavimui

2003-ųjų birželį Sam Ruby pradėjus atvirą projektą, prie jo prisijungė daug programuotojų bei palaikančių asmenų[13]. Pagrindiniai iškelti tikslai: sukurti nekomercinį produktą įgyvendinamą visų lengvai praplečiamą detaliai ir aiškiai specifikuotą.

Ši iniciatyva sąlygojo greitą Atom formato atsiradimą. Semantinėse aplikacijos netruko jį panaudoti daugiau nei 150 palaikytojų, tarp jų tokie kaip: Technorati (http://www.technorati.com/), Six Apart (http://sixapart.com/), LiveJournal (http://www.livejournal.com/), Creative Commons (http://creativecommons.org/) ir kt.. Tačiau dėl silpnai valdomo sprendimų priėmimo proceso diskusija pasidarė chaotiška ir

[13] http://www.intertwingly.net/wiki/pie/RoadMap

nebevaldoma. Po Atom 0.2 versijos išleidimo 2003-ųjų liepą diskusija nutrūko ir išsiskaidė ties individualiais projektais.

Diskusija atsinaujino, kai 2003-ųjų gruodį buvo paskelbta Atom 0.3 versija ir Google[14] korporacija įtraukė ją į keletą savo projektų (Blogger, Google News, Gmail). Google aplikacijų programavimo sąsaja (GData[15]) yra paremta RSS 2.0 bei Atom 1.0 specifikacijomis, kurios dar tebėra vystomos. 2004-ieji prasidėjo nuo perėjimo prie oficialaus W3C principais paremto projekto su centralizuotu valdymu, nustatytom proceso taisyklėm. Paskutinis juodraštinis variantas apie Atom sindikavimo formatą yra pripažintas ir palaikomas IETF kaip RFC 4287[16].

Šiuo metu naudojamos dvi Atom pranešimų versijos: 0.3 bei 1.0 (0.3 jau nebevystoma ir pagrindinis validatorius siūlo jos atsisakyti[17]). Platus palaikymas specifikacijoms, naudojimas portalų informacijos sindikavimui, o dauguma populiariausių pranešimų agregatorių yra pritaikyti Atom faile esančios informacijos nuskaitymui ar atvaizdavimui.

2.1.8. Atom ir RSS duomenų sindikavimo formatų palyginimas

Vystant technologiją ir atsiradus skirtingiems interpretavimams, išskiriami skirtingi RSS formatai pateikti [8] publikacijoje:

- RSS 0.91, 0.92, 2.0 – tikrai paprastas surinkimas arba išsamus tinklapio anotavimas *(angl. Really Simple Syndication arba Rich Site Summary).* Nėra ontologijos.
- RSS 1.0 – resursu aprašymo sistema paremtas puslapio aprašymas *(angl. Resource Description Framework Site Summary).* Aprašymas remiasi RDF ir sudaro tam tikrą ontologijų poaibį.

Matyti, kad RSS išsiskiria į dvi formatų šeimas, o Atom formatas turi 3 pagrindines versijas: 0.2, 0.3 ir 1.0, kuriose išlaikyti tie patys principai.

[14] http://www.google.com
[15] http://code.google.com/apis/gdata/index.html
[16] http://atompub.org/rfc4287.html
[17] http://feedvalidator.org/news/archives/2005/09/15/atom_03_deprecated.html

Atom 1.0 formatu pateikto pranešimo pavyzdys:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

 <title>Pavyzdinis pranešimas</title>
 <subtitle>Pavyzdinio pranešimo paantraštė.</subtitle>
 <link href="http://www.pvz.lt/"/>
 <updated>2006-09-01T18:30:02Z</updated>
 <author>
   <name>Gediminas Sakalauskas</name>
   <email>gediminas.sakalauskas@pvz.lt</email>
 </author>
 <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>

 <entry>
          <title>Pranešimo įrašas</title>
          <link href="http://www.pvz.lt/2006/09/01/atom1"/>
          <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
          <updated>2006-09-01T18:30:02Z</updated>
          <summary>Teksto pavyzdys.</summary>
 </entry>

</feed>
```

RSS 2.0 formatu pateikto pranešimo pavyzdys:

```
<?xml version="1.0"?>
<rss version="2.0">
   <channel>
      <title>Pavyzdinis pranešimas</title>
      <link>http://www.pvz.lt/rss"</link>
      <description>Pavyzdinio pranešimo paantraštė.</description>
      <language>en-us</language>
      <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
      <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
      <docs>http://www.pvz.lt/pranesimai.html</docs>
      <generator>Weblog Editor 2.0</generator>
      <managingEditor>editor@pvz.lt</managingEditor>
      <webMaster>webmaster@pvz.lt</webMaster>

      <item>
         <title>Pranešimo įrašas</title>
         <link>http://www.pvz.lt/2006/09/01/pranesimas1.html</link>
         <description>Teksto pavyzdys.</description>
         <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
         <guid>http://www.pvz.lt/2006/09/01/pranesimas1.html#item573</guid>
      </item>
   </channel>
</rss>
```

Palyginimas pateikiamas 1 lentelėje, kurioje matome, kad RSS 2.0 aprašo 30 elementų, o Atom 1.0 – 21.

1 lentelė. RSS 2.0 ir Atom 1.0 elementų palyginimas

RSS 2.0	*Atom 1.0*	*Komentaras*
rss	-	Specifinis tik RSS
channel	feed	Skirtingas pavadinimas
title	title	
link	link	Atom turi išplėstą struktūrą rel reikšmėms
description	subtitle	Skirtingas pavadinimas
language	-	Atom naudoja standartinį xml:lang atributą
copyright	rights	Skirtingas pavadinimas
webMaster	-	Specifinis tik RSS
managingEditor	author / contributor	
pubDate	published	Atom šį atributą perkelia į Entry klasę
lastBuildDate	updated	RSS neturi tokio Item klasės atributo
category	category	
generator	generator	
docs	-	Atom atsisakyta
cloud	-	Atom atsisakyta
ttl	-	<ttl> problematiškas, Atom atsisakyta
image	logo	Atom rekomenduojamas 2:1 santykis
-	icon	Interpretuojamas kaip favicon.ico
rating	-	Atom atsisakyta
textInput	-	Atom atsisakyta
skipHours	-	Atom atsisakyta
skipDays	-	Atom atsisakyta
item	entry	Skirtingas pavadinimas
author	author	
-	contributor	Atom pasiūlytas
description	summary / content	Priklausomai ar įtraukiamas pilnas turinys
comments	-	Atom atsisakyta
enclosure	-	Atom perkeltas į <link>
guid	id	Skirtingas pavadinimas
source	-	Atom perkeltas į <link>
-	source	Konteineris meta informacijai aprašyti

Pilnas RSS 2.0 ir Atom 1.0 palyginimas su aprašymais patalpintas adresu: http://www.intertwingly.net/wiki/pie/Rss20AndAtom10Compared.

2.1.9. Kategorizavimo uždavinys pasaulinio tinklo kontekste

Kategorizavimas – procesas, kurio metu idėjos ir objektai yra atpažįstami, diferencijuojami ir suprantami. Objektai yra sugrupuojami į kategorijas dėl kažkokios aiškios priežasties. Idealiu atveju kategorizavimas nusako subjekto ir objekto ryšį remiantis žinojimu. Šis procesas yra pagrindas kituose procesuose: nuspėjimo, interferencijos, sprendimų priėmimo ir panašiai. Egzistuoja įvairūs būdai ir metodai kategorizavimo procesui.

Su kategorizavimu praktikoje susiduriame kiekvieną dieną vartydami katalogą, ieškodami bibliotekoje ar bandydami organizuoti informaciją. Nors kategorizavimas yra natūralus procesas žmogaus mastyme, jis yra individualus ir priklauso nuo kiekvieno asmeninio supratimo, o tai sąlygoja skirtingus kategorizavimo rezultatus. Būtent todėl neturime vieningos kategorijų sistemos visoje žmonijoje.

Kaip ir standartizavimas, taip ir kategorizavimas galimas tik nustačius griežtas ribas, taisykles ir jų laikantis. Vienas sėkmingų pavyzdžių kategorizuojant internetines svetaines yra atviro katalogo projektas – DMOZ[18].

Atviro katalogo projektas (ODP) *(angl. Open Directory Project)* – tai didžiausias ir išsamiausias žmonių redaguojamas katalogas Internete. Jį kuria ir prižiūri plati pasaulinė savanorių redaktorių bendruomenė.

DMOZ duomenys saugomos RDF pavidalu ir yra atviri, lengvai panaudojami semantinių portalų kūrimui. Nustatytos taisyklės ir valdymas leidžia palaikyti vieningą kategorizavimą ir spartų vystymą.

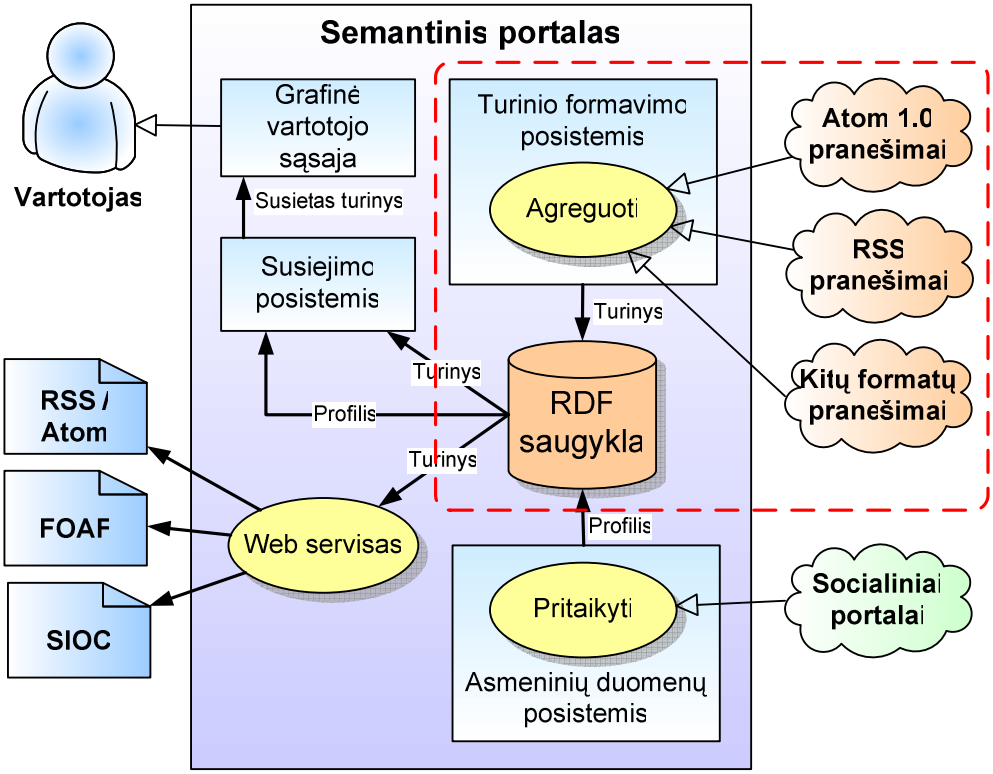
Kategorijų struktūra turi hierarchiją, bet nėra medžio tipo grafas. Viršuje esanti Top kategorija turi nusistovėjusias subkategorijas: Menas, Mokslas, Kompiuteriai, Sveikata, Pasaulis ir tt.. Šios savo ruožtu turi subkategorijas, kurios gali turėti dar keliolika lygių. Kategorija „Lietuvių“ priskiriama kategorijai „Top\World“ kaip subkategorija. Kategorija „Naujienos“ su įvairiom sritimis rastume šiek tiek giliau, o be to ji gali būti papildyta ir išplėsta pagal poreikius.

Atom 1.0, RSS 2.0 turi numatytas žymes <`category`> su atributais ir reikšmėmis nusakančias informacijos vietą kategorijų grafe. Šių žymių gali būti daugiau nei 1, todėl leidžiama nusakyti, kaip duomenys siejasi su kitomis kategorijomis. Kategorizavimo uždavinys šiame darbe nesprendžiamas, tačiau rezultatuose pateikiama informacija gali būti surūšiuota interpretuojant objektų ryšius su kategorijų hierarchija.

[18] http://www.dmoz.org/

2.2. Metodą realizuojančios sistemos duomenų agregavimui vizija

Darbo tikslas – pasiūlyti metodą ir jį realizuojančią sistemą, kuri papildytų esamas semantinio pasaulinio tinklo paslaugas aprašytas 2.1.4 skyrelyje. Siūlomas metodas panaudojamas siekiant įgyvendinti aukštesnio lygio metodą, o sistema bus viena iš semantinio portalo sudedamųjų dalių, kurio modelį pasiūlė Vytautas Taujanskas straipsnyje „Categories Extraction for Reuse In Semantic Applications and Profile Based Recommendation Service“ [1]. Principinė portalo ir metodą realizuojančios sistemos schema pateikta 10 paveiksle.



10 pav. Siūloma rekomendacinio portalo schema

Portalo kontekste sistemos paskirtis agreguoti duomenis iš įvairių pasaulinio tinklo šaltinių, apjungti į bendrą ontologiją ir pateikti duomenis ontologijų aprašu kitai posistemei. Kitos posistemės galėtų panaudoti informacijos išrinkimui ir pateikimui pagal vartotojo poreikius. Šio darbo sritis apibrėžta 10 paveiksle punktyrine linija.

Pagrindiniai uždaviniai suformuojami iš ryšių su kitomis posistemėmis:

- Agreguoti duomenis iš PT pagal nurodytą URL Atom 1.0, RSS 1.0, RSS 2.0 ir kitų formatų pranešimų pavidalu.
- Išgauti informaciją iš pranešimų ir užpildyti jų ontologiją turiniu.
- Ontologijos su turiniu patalpinimas, modifikavimas RDF saugykloje.
- Išgauti ontologijos turinį pagal semantines užklausas
- Pateikti suformuotą ontologiją su turiniu kitai posistemei tolimesniam panaudojimui.

2.3. Decentralizuotos informacijos pasauliniame tinkle agregavimo problematika

Informacija dideliais kiekiais ir įvairiais formatais nuolatos publikuojama pasauliniame tinkle. Egzistuojant didžiulei įvairovei publikuojamų dokumentų formatų ir technologijoms iškyla problemos informaciją teisingai paskirstyti ir pateikti vartotojui, tai kas jį domina, prasminga ir reikalinga. Sritys, kuriose susiduriama su problemomis priklauso nuo darbo uždavinių ir sistemai keliamų tikslų.

Žemiau pateikiami probleminės srities aprašai pagal uždavinius.

2.3.1. Informacijos kaupimo ir paskirstymo uždavinio problema

Apima pranešimų parsiuntimą. Problemos parsiunčiant kyla dėl formato neatitikimo specifikacijoms. Esant duomenų neatitikimui keliamiems reikalavimams jie bus ignoruojami, o vartotojui parodomas pranešimas. Patikrinti ar duomenų šaltiniai atitinka specifikacijas bus panaudota aplikacija FeedValidator (http://feedvalidator.org/), nagrinėjanti pranešimų sintaksę

Pranešimo informacija surašoma į ontologijos struktūrą, kuri saugoma RDF trijulės duomenų bazėje. Saugojimui pasitelkiama instrumentinė priemonė darbui su ontologijomis (procesas detalizuotas 4.1.5 skyriuje). Duomenys ir rezultatai bus saugomi panaudojant instrumentines priemones duomenų bazėje, todėl problemos gali kilti esant duomenų bazės sutrikimams. Dėl saugojimo mechanizmo klaidų suformuotos neoptimalios SPARQL užklausos duomenų talpinimas gali užtrukti per daug laiko. Esant netaisyklingoms užklausoms duomenys gali būti prarandami, todėl reikalingas mechanizmas užtikrinantis duomenų apsikeitimo procesą.

Paskirstymo uždavinys apjungia duomenų išgavimą ir perdavimą grafinei sąsajai. Problemos gali kilti užklausų generavime ir jų rezultatų pateikime. Grafinei vartotojo sąsajai duomenys pateikiami jau suformuoti todėl, kad nekiltų problemų, būtina nustatyti kokie duomenys ir kaip turi būti atvaizduoti.

2.3.2. Kategorizavimo uždavinio problema

Kategorizavimo poreikis formuojasi natūraliai, kai vartotojas naršo katalogą ar ieško jame informacijos. Paieška gali būti rankinė ar automatinė, o jos sėkmė, rezultatų kokybė, tikslumas, paieškos trukmė priklausys nuo to, kaip teisingai suorganizuota informacija – šiuo atveju sukategorizuota. Problema sudarant kategorijas ir paskirstant informaciją po kategorijas priklauso nuo autoriaus interpretavimo ir jo pasaulio suvokimo. Jei autorių yra daugiau, jie gali suvokti skirtingai ir atsiranda neaiškumas. Šios problemos darbe

nesprendžiame, bet panaudojame pranešimų atributuose esančią informaciją jų kategorizavimui.

Kategorizavimas bus atliekamas išnagrinėjus `<category>` žymę Atom 1.0 pranešimo faile. Problemos atsiras jeigu pranešimo kategorijos žymės nebus arba ji bus suformuota ne pagal DMOZ – atvirojo katalogo schemą. Tokiu atveju būtų galima pasirinkti kitą schemą.

Galimi du sprendimo būdai: 1- atmesti pranešimą kaip netinkantį į jokią kategoriją; 2- įvesti modulį pasirūpinantį pranešimo kategorijos išgavimu ir turinio jai priskyrimu. Kadangi kategorija bus ontologijos grafo viršūnė susiejama su pranešimo turiniu, negalimas tokių pačių viršūnių pasikartojimas grafe. Reikalingas viršūnių peržiūrėjimo ir suvienodinimo mechanizmas, kuris užtikrintų kategorijos unikalumą.

Naujienos pranešimo vieta kategorijų struktūroje nusakoma jau pradiniuose duomenyse. Tarpinis saugojimas ontologijoje reikalingas tam, kad galėtume formuoti semantines užklausas. Atvaizdavimui grafinėje sąsajoje šios užklausos bus supaprastintas variantas. Kategorijos bus kaip pranešimų atributai ir pilnai leis atvaizduoti kategorijų struktūrą grafinėje sąsajoje – turinys atvaizduojamas grupuojant naujienas pagal šiuos atributus.

2.3.3. Pateikimo vartotojui uždavinio problema

Pateikiamas rezultatas bus formuojamas iš nurodytų duomenų šaltinių. Reikalinga įvesti internetines nuorodas, kuriose publikuojami Atom 1.0, RSS 2.0, RSS 1.0 ar kitų sindikavimo formatų pranešimai. Eksperimentams naudosime teisingus duomenų šaltinius, suformuotus pagal visus reikalavimus duomenims. Tačiau vartotojas gali įvesti duomenis, kurie neatitinka jiems keliamų reikalavimų. Siekiant išvengti problemos, šaltiniai iš pradžių bus validuojami, o nevaliduoti ar netinkami duomenys sindikavimo proceso metu bus ignoruojami. Baigiant agregavimą apie netinkamus šaltinius vartotojas bus informuojamas.

Rezultatai bus pateikiami vartotojui grafinėje sąsajoje per Interneto naršyklę. Galimi nesuderinamumai tarp formato interpretavimų, todėl grafinė sąsaja suformuojama HTML ar XHTML interpretatoriaus turi būti validuojama pagal W3C standartus su Markup Validation Service įrankiu (http://validator.w3.org/).

Problematiška bus atvaizduoti kategorijų hierarchiją, nes kategorijų struktūra pagal plačiausiai paplitusią atvirojo katalogo sistemą DMOZ yra grafas. Tokiu atveju grafiniam atvaizdavimui reikės organizuoti medį, o prie viršūnės pateikti susijusias kategorijas, taip praplečiant medžio struktūrą.

2.4. Siūlomo problemų sprendimo analizė

Su sprendimo įgyvendinimu susijusi dalykinė sritis buvo pateikta 2.1 skyriuje. Toliau detaliau analizuojami priemonės, metodai ir įrankiai naudojami siūlomam metodui įgyvendinti.

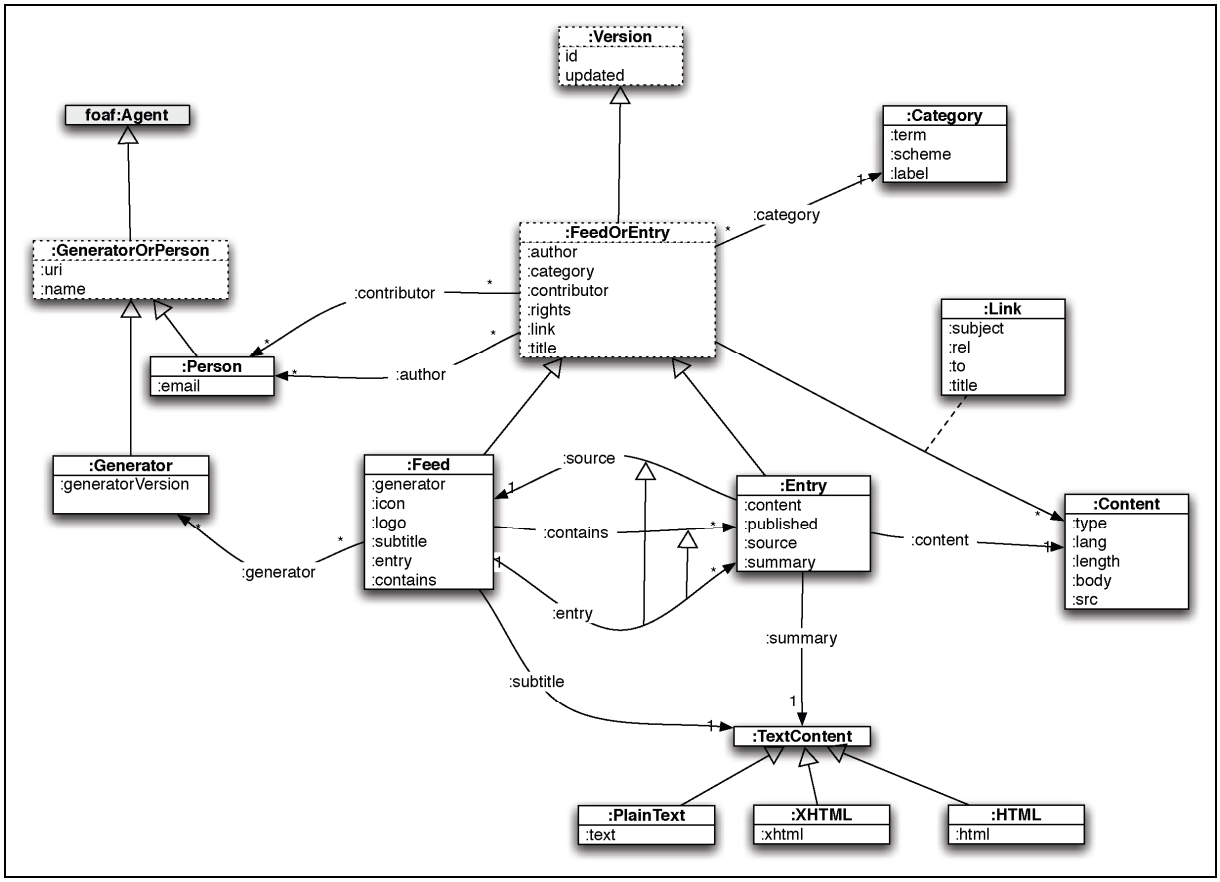
2.4.1. Atom 1.0 formato panaudojimas duomenims

Šio formato savybės ir panaudojimo galimybės buvo nagrinėtos 2.1.6 skyriuje. Būtent dėl autorių sudarytos ontologijos ir egzistuojančių priemonių darbui su šiuo formatu pateiktais duomenimis naudosime jį pradinių duomenų aprašymui, atliekant eksperimentą šiame darbe.

Kadangi kol kas nėra patvirtinta formato galutinė versija, darbo eigoje reiktų stebėti galinčius atsirasti pakeitimus ir į juos atsižvelgti. Šiuo metu vadovaujamės UML diagrama atvaizduojančia Atom 1.0 struktūrą pateiktą 11 paveiksle. Ontologijos struktūra aprašyta NG-Relax notacija (Priedas B) ir RFC 4287 dokumente pateiktomis specifikacijomis.

2.4.1.1. Atom 1.0 formato UML schema

Atom 1.0 formato elementų UML schema pateikta 11 paveiksle.



11 pav. Atom 1.0 pranešimo sandaros UML schema

Schemoje pateikta elementų, galimų atributų ryšiai ir galimi laukai pagal UML notaciją. Matome, kad Atom struktūrą būtų galima pritaikyti ir reliacinei duomenų bazei. Tačiau šiame darbe Atom struktūros elementus saugosime ne reliacinės duomenų bazės lentelėse pagal atributus ir ryšius, bet kaip ontologiją. Reliacinė duomenų bazė bus tik kaip terpė, kurioj pagal RDF struktūrą saugomi: subjektas, predikatas, objektas (žiūr. 2.1.3.2 skyrių).

Detalus Atom 1.0 formato elementų aprašymas pateikiamas RFC 4287 dokumente (žiūr. 2.1.7 skyrių).

2.4.1.2. Atom 1.0 formatą aprašančios ontologijos

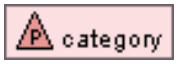
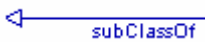
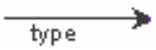
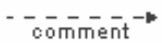
Atom 1.0 formato autoriai siekdami formaliai aprašyti kuriamą formatą aprašė jį ontologija. Formato ontologijas galima rasti skirtingomis notacijomis sudarytuose failuose:

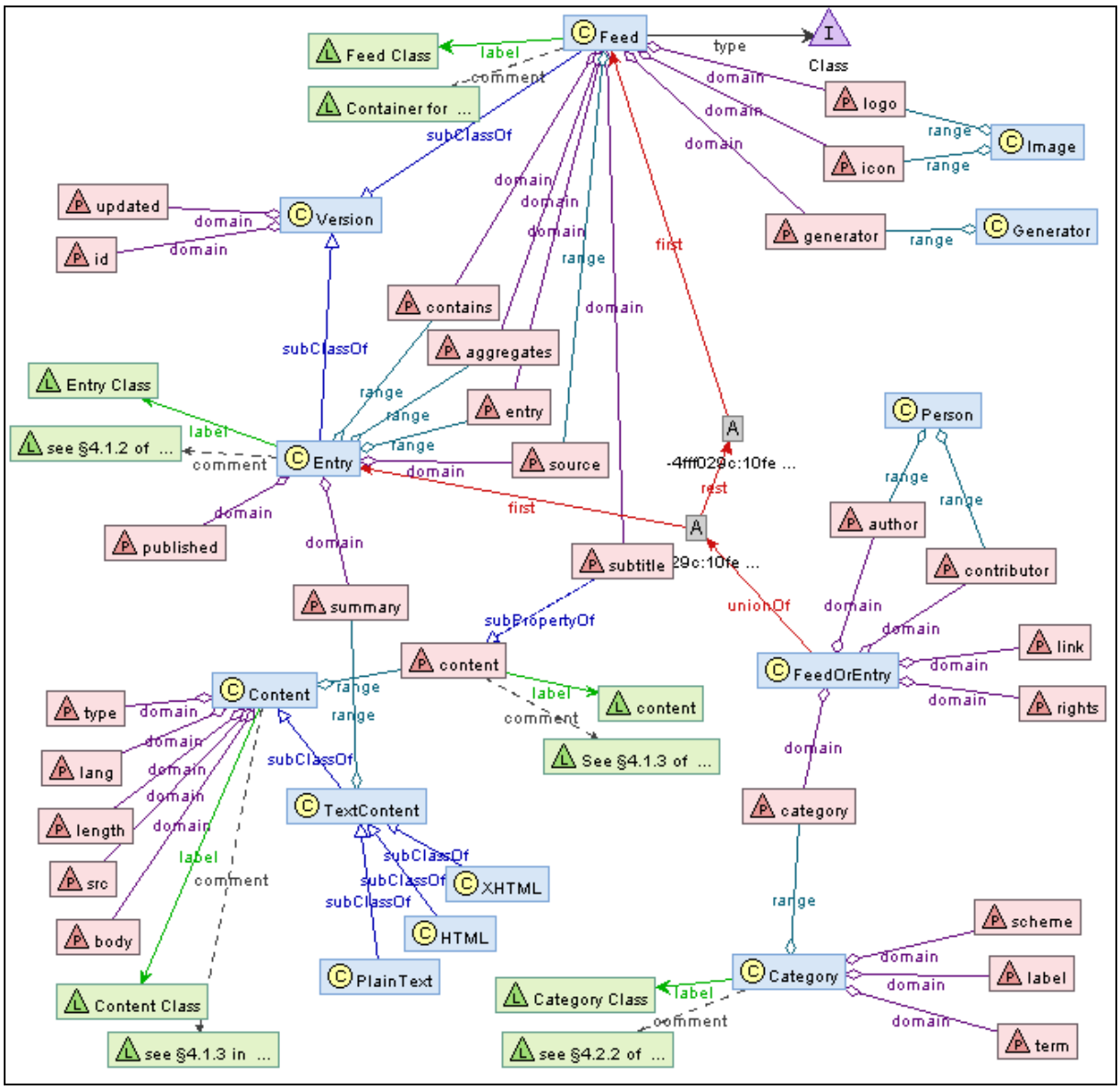
Notacija	*Nuoroda į failą*
RDF/XML	http://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.rdf
N-Triples	http://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.ntriples
Turtle	http://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.ttl
N3	http://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.n3

Fragmentas iš sudarytos ontologijos atvaizduotas grafiniame interpretatoriuje pateikiamas 12 paveiksle.

Paveiksle pateiktoje struktūroje matomų elementų notacija pateikiama 2 lentelėje.

2 lentelė. Grafiškai atvaizduotos ontologijos objektų notacija

Grafinis simbolis	*Reikšmė*
	Aprašo konceptus arba klases ontologijoje. Šiuo atveju formate apibrėžta įrašo klasė.
	Nurodo eilutės simbolių reikšmę. Gali turėti duomenų tipą (*string*, *integer*)
	Nurodo neidentifikuotą egzempliorių (tuščią viršūnę grafe)
	Nurodo konkretų egzempliorių.
	Nurodo objekto savybę. Šiuo atveju, kokia yra šaltinio ar įrašo kategorija.
	Parodo kaip susiejama objekto klasė su kita klase. Šiuo atveju subklasė.
	Parodo koks yra klasės arba koncepto tipas.
	Komentaras apie konceptą.
	Kito tipo savybės aprašytos ontologijoje ir siejančios objektus.



12 pav. Atom 1.0 formato ontologijos fragmentas

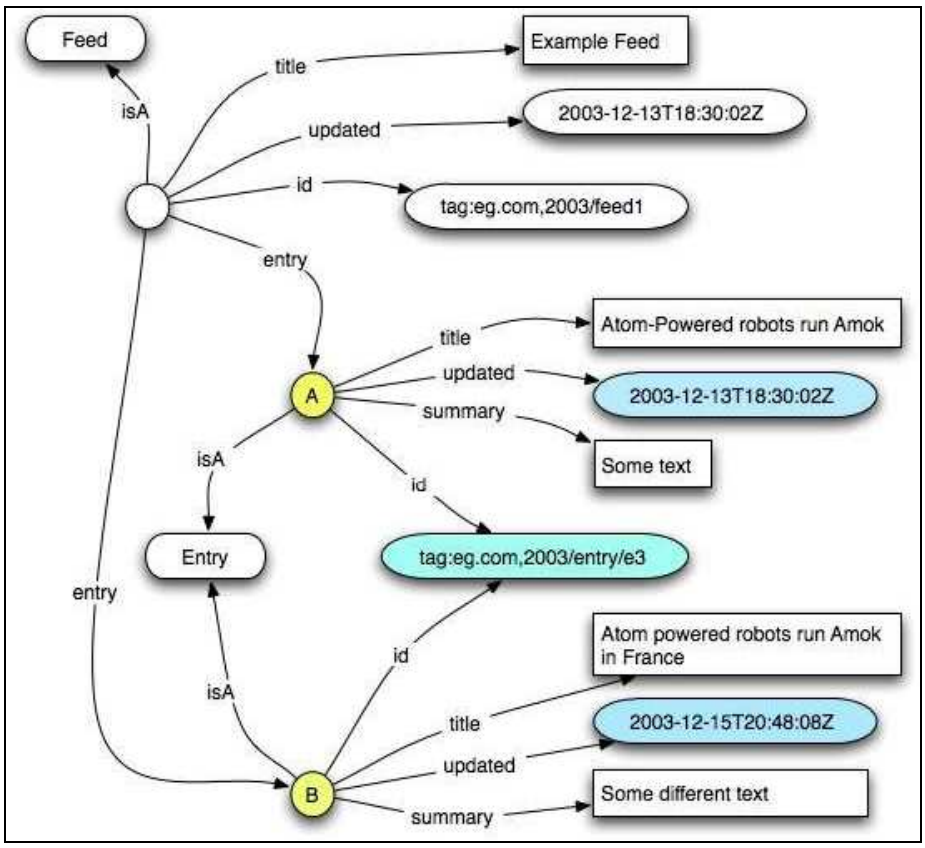
Pateikiama tik dalis ontologijos, nes pilnai aprašytą ontologiją atvaizdavus problematiška skaityti dėl ryšių ir elementų gausos.

2.4.1.3. Atom 1.0 formato ontologijos schemos pavyzdys

Atom 1.0 formato pranešimo fragmentas pagal ontologijos schemą pateiktas 13 paveiksle.

Schemoje pateiktas pavyzdys su pranešimo elementais, kurio pavadinimas „Example Feed“. Pranešimas turi du elementus A ir B, kurie susiję su savo atributais ir savybėmis. Kiti atributai ir savybės lengvai perskaitomi žmogui bei turės semantinę prasmę aplikacijai.

Elementai aprašyti ontologijos schemoje 13 paveiksle pateikiami RDF trijulėmis 3 lentelėje.



13 pav. Atom 1.0 pranešimo ontologijos pavyzdys.

3 lentelė. RDF trijulės Atom 1.0 formato pranešime

Objektas	*Savybė*	*Reikšmė*	*Komentaras*
-	isA	Feed	Duomenų šaltinis išplaukiantis iš aprašymo, ne iš savybės.
-	title	Example Feed	Pavadinimas
-	updated	2003-12-13T18:30:02Z	Datą nurodanti duomenų šaltinio atnaujinimo reikšmė
-	id	tag:eg.com.2003/feed1	Duomenų šaltinio identifikatorius
-	entry	A	Savybė siejanti duomenų šaltinį su įrašo objektu A
-	entry	B	Savybė siejanti duomenų šaltinį su įrašo objektu B
A	isA	Entry	Duomenų šaltinio įrašas išplaukiantis iš aprašymo
A	title	Atom Powered robots Run Amok in France	Pavadinimo kaip objekto B savybės resursas
A	updated	2003-12-15T20:48:08Z	Datą nurodanti įrašo atnaujinimo reikšmė
A	summary	Some text	Santrauka apie įrašą
A	id	tag:eg.com.2003/entry/e3	Įrašo A identifikatorius
B	isA	Entry	Duomenų šaltinio įrašas išplaukiantis iš aprašymo
B	title	Atom Powered robots Run Amok	Pavadinimo kaip objekto A savybės resursas
B	updated	2003-12-13T18:30:02Z	Datą nurodanti įrašo atnaujinimo reikšmė

B	summary	Some different text	Santrauka apie įrašą
B	id	tag:eg.com.2003/entry/e3	Įrašo B identifikatorius

Paaiškinimai:

- Objektas gali būti neidentifikuotas URI (lentelėje ties jo reikšme brūkšnys) – anoniminė grafo viršūnė, tačiau pagal priežastis ir susiejimą su kitais objektais jis turi prasmę ontologijoje.
- A ir B objektai identifikuoti – nėra anoniminiai (paveiksle A ir B apskritimai). Praktikoje identifikuojama URI.
- A ir B nėra tas pats objektas, nes skiriasi jų savybių reikšmės, nors jie ir rodo į tą patį identifikatorių – informacija apie versijavimą ir buvusieji duomenys neprarandami.

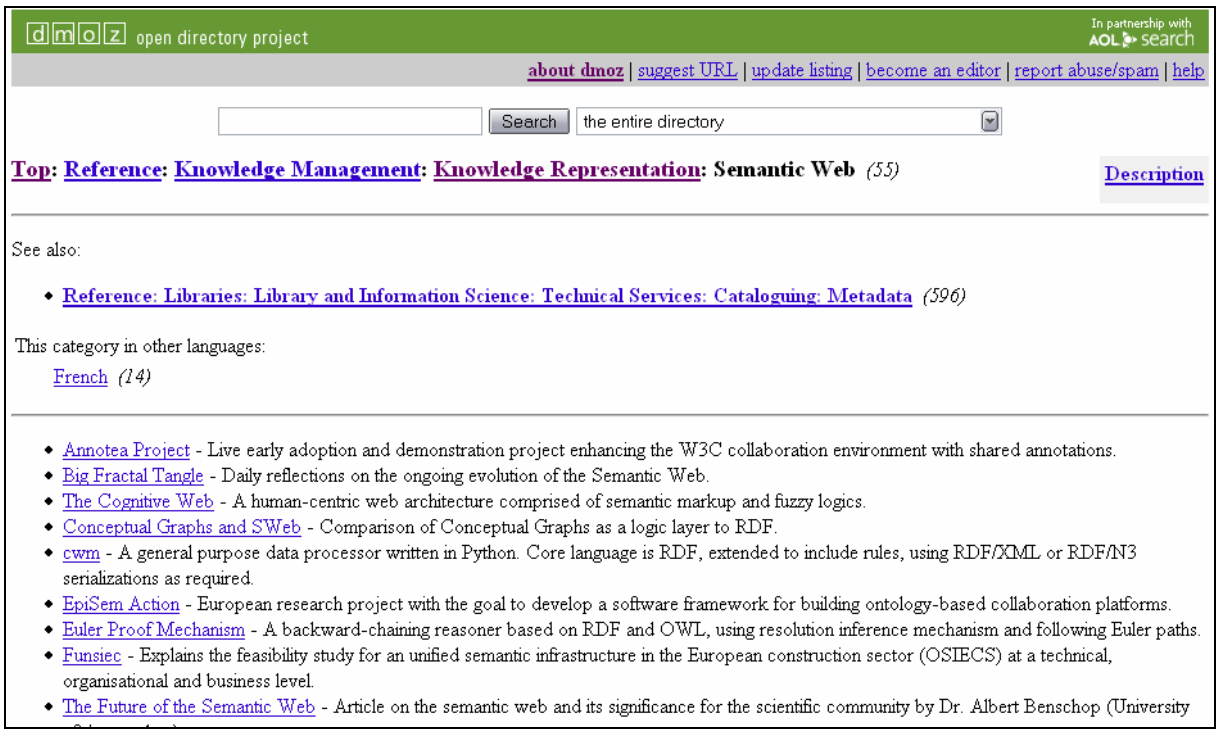
2.4.2. DMOZ kategorijų schemos panaudojimas kategorijų aprašyme

DMOZ atviro katalogo sistema aprašyta kategorizavimo uždavinio skyriuje 2.1.9. Pradiniuose duomenyse pagal šią schemą bus priskiriamos įrašų kategorijų atributų reikšmės.

Reikalingą kategorijų hierarchiją galima užrašyti URI išraiškoje ir ją galima priskirti skirtingiems atributams:

- Objektus vienijanti schema priskiriama pranešimo kategorijos atributo `<schema>` reikšmei.
- Atributui `<term>` bus paimama schemą aprašančios URI dalis esanti po dešinio brūkšnio „/“.
- `<label>` atributui suteikiama reikšmė, formuojant pranešimą arba iš atributo `<term>` reikšmės pašalinant nereikalingus simbolius.

14 paveiksle pateikiamas vaizdas DMOZ sistemoje, kurio URL http://dmoz.org/ Reference/ Knowledge_Management /Knowledge_Representation/ Semantic_Web/ . Šis URL gali būti išskaidytas ir panaudotas duomenų aprašyme sindikavimo metu.



14 pav. DMOZ kategorijos turinys

Atviro katalogo sistema yra pastoviai atnaujinama ir papildoma. Kategorijos gali prisidėti, skaidytis, todėl reikėtų stebėti ir pervadinti duomenų bazėje esančias kategorijas.

2.4.3. ROME[19] paketo panaudojimas duomenų agregavimui

ROME atviro kodo paketas naudojamas pranešimų tame tarpe ir Atom 1.0 sindikavimui, agregavimui. Buvo sukurtas siekiant apjungti įvairių formatų pranešimų agregavimą išlaikant lankstumą ir pilnumą. Paketo 0.8 versija atpažįsta šiuos formatus: RSS 0.92, RSS 0.93, RSS 2.0, RSS 1.0 ir Atom 0.3, Atom 1.0.

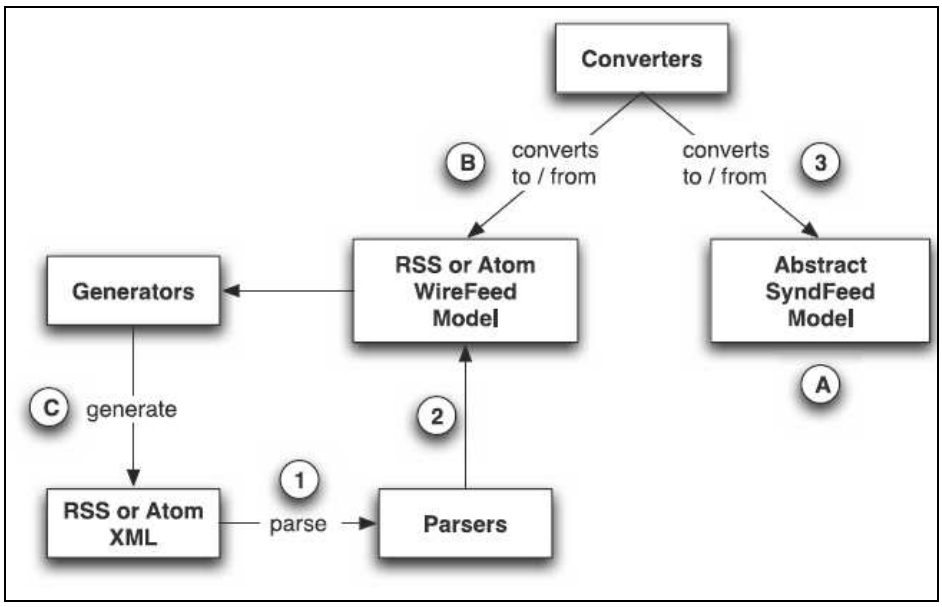
Pagrindinis trūkumas, kad XML nuskaitymas remiasi DOM metodais, kurie pasižymi prastu atminties planavimu bei netaisyklingo XML failo suformatavimo netoleravimu. Esant blogai suformuotam XML – neveiks.

15 paveikslas atvaizduoja ROME veikimo principus (paimta iš [9] psl. 145). Čia:

- A – abstraktus modelis SyndFeed, kuris gali būti konvertuojamas arba į jį konvertuojamas
- B – WireFeed modelis galintis aprašyti RSS arba Atom formato pranešimus.

Pastarasis gali būti naudojamas agreguojant arba generuojant Atom, RSS ar kitokių tipų pranešimus XML formatu.

[19] https://rome.dev.java.net/



15 pav. Pagrindiniai ROME veikimo žingsniai

ROME veikimas pagrįstas abstraktaus tarpinio modelio vadinamo SyndFeed įvedimu, kuris leidžia aprašyti visus egzistuojančius formatus išsaugant jų atributiką.

Atom 1.0, RSS 1.0, RSS 2.0 formatai dirbant su ROME yra realizuoti aprašant WireFeed objektą.

Eksperimente bus naudojami išimtinai Atom 1.0 formatu pateikiami duomenys ir SyndFeed modelis nebus reikalingas. Tiesiogiai naudosime WireFeed modelį, kuris pateiktas 16 paveiksle ir pilnai padengia Atom formato struktūrą.

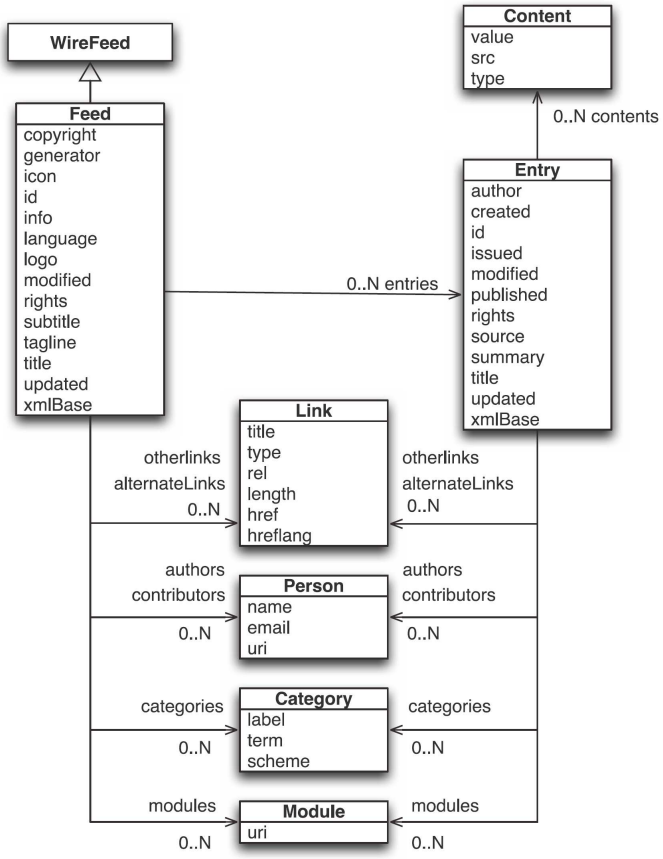
Siūlome metode bus aktualu panaudoti ROME galimybes 15 paveiksle pavaizduotas numeriais 1 ir 2:

1 (*parse*) – duomenų nuskaitymas iš nurodyto URL adreso į XML struktūrą;

2 – XML struktūroje saugomų duomenų perkėlimas į WireFeed objektą.

ROME branduolys yra lengvas procesas, tačiau priklauso nuo JDOM skaitytuvo (angl. *parser*). Branduolys gali būti praplečiamas prijungiant kitus projektus pagal poreikius, įtraukiant jį per bibliotekas. Architektūra leidžia praplėsti ir pakeisti kai kurias sindikavimo ir agregavimo taisykles.

ROME 0.8 versija dar tebėra vystymo stadijoje ir iki 1.0 versijos išleidimo gali atsirasti neesminių pakeitimų, į kuriuos derėtų atsižvelgti, jei jie daro įtaką šiam darbui.



16 pav. WireFeed modelis pilnai padengiantis Atom 1.0 formato struktūrą

2.4.4. JENA[20] instrumentinės priemonės panaudojimas darbui su ontologijomis

JENA yra atviro kodo JAVA kalba paremta instrumentinė priemonė semantinio pasaulinio tinklo įgyvendinimui. Projektas skirtas aplikacijų programavimo sąsajai (API) veiksmams su RDF kurti. Jis suteikia programavimo aplinką darbui su RDF, OWL, SPARQL ir yra valdomas remiantis SPT taisyklėmis.

Pagrindinės JENA sudedamosios dalys:

- RDF aplikacijų programavimo sąsaja
- RDF skaitymas ir rašymas RDF/XML, N3 ir N-Triples formatais
- OWL aplikacijų programavimo sąsaja
- Laikinos atminties ir mechanizmu išsaugant į RDB paremtos saugyklos
- SPARQL užklausų kalba paremtas užklausų variklis

JENA naudosime šiame darbe, nes egzistuoja projekto palaikymas; W3C remia kaip pagrindinį semantinio pasaulinio tinklo kūrimo įrankį; parašytas ir panaudojamas su JAVA technologijomis; susiejamas su kitomis atviro kodo programomis.

[20] http://jena.sourceforge.net/

Egzistuoja kitos instrumentinės priemonės darbui su ontologijomis: Sesame[21], KAON[22].

Šiuo metu labiausiai plėtojamos ir palaikomos yra Sesame ir Jena, todėl jų palyginimas pateikiamas 4 lentelėje.

4 lentelė. Instrumentinių priemonių darbui su ontologijomis palyginimas

Savybė	**Sesame**	**Jena**
Atviras kodas	Taip	Taip
Palaikoma Java	Taip	Taip
API saugykla	RDF	RDF, OWL
Ontologijų formatas	RDF	RDF, OWL (lite, DL, full)
DBVS	MySQL, PostgreSQL, Oracle	MySQL, PostgreSQL, Oracle, MS SQL
Užklausų kalbos	SPARQL	SPARQL, RDQL, ARO (SQRQL)
Saugojimas atmintyje	Pastoviojoje, laikinoje, duomenų bazėje	laikinojoje, pastoviojoje, duomenų bazėje (mechanizmas išsaugant į RDB)
Įrankis duomenų klasėms generuoti	Ne	Taip (schemagen)
W3C rekomenduoja	--	Taip

2.4.5. Schemagen[23] panaudojimas JAVA klasių generavimui

Schemagen („schemų generatorius“) yra vienas iš standartinių JENA instrumentinės priemonės įrankių skirtas JAVA klasėms generuoti iš RDF schemos, OWL ar DAML žodyno.

Atom 1.0 formato ontologijos struktūrą aprašantis RDF failas talpinamas adresu: http://bblfish.net/work/atom-owl/2006-06-06/AtomOwl.rdf . Šis failas bus panaudojamas Atom 1.0 klasei su atributais generuoti. Klasės vėliau bus naudojamos nusakant objekto tipą, savybes ir atributus.

Pritaikant metodą darbui su kitais pranešimų formatais iš jų ontologijos schemą aprašančių RDF failų taip pat labai lengvai galima sugeneruoti JAVA klases panaudojus schemagen įrankį.

[21] http://www.openrdf.org/
[22] http://sourceforge.net/projects/kaon
[23] http://jena.sourceforge.net/how-to/schemagen.html

2.4.6. Eclipse[24] SDK platformos panaudojimas sistemos implementacijai

Eclipse – atviro kodo integruota programų kūrimo aplinka. Eclipse susideda iš daugelio įskiepių kurių kiekvienas gali turėti savo langus, savo meniu punktus, savo nepriklausomą nuotolinio atnaujinimo sistemą ir pan. Nors autoriai teigia, jog parašius tinkamus modulius, su Eclipse galima daryti „bet ką“, dažniausiai ji naudojama programoms kurti. Parašyta Java kalba, Eclipse veikia visose populiariose operacinėse sistemose.

Java kodui Eclipse turi gerai išvystytą tekstų redaktorių (jau renkant kodą automatiškai nurodomos trivialios klaidos), klasių, kintamųjų ir metodų automatinį pervardinimą, palaiko Ant[25] ir JUnit[26]. Inkrementinis kompiliavimas (po nedidelio pakeitimo kompiliuojama tik pakeista dalis) sudėtinguose projektuose pagreitina darbą kelias dešimtis ir daugiau kartų. Gerai išvystyta versijų kontrolės sistema leidžia tą patį projektą vystyti net ir po visą pasaulį išsibarsčiusiems programuotojams.

Privalumas yra gausi daugiau nei tūkstančio įvairių įskiepių biblioteka ir geros galimybės pačiam tokius įskiepius kurti.

Kadangi Eclipse platinama su atviro kodo licenzija, pastaruoju metu atsirado tendencijos ją pritaikyti kaip įvairių specializuotų programų dalį, panaudojant esamas vartotojo sąsajos galimybes. Eclipse iš esmės nesiremia Swing (aplikacijų programavimo sąsaja grafinei vartotojo sąsajai kurti) galimybėmis, jas pakeisdama savo SWT[27] klasėmis. Dėl šio priežasties ji jau seniai nėra priklausoma nuo Sun[28] korporacijos ir gerai veikė su atviro kodo GNU Classpath[29] biblioteka net ir tada, kuomet jos Swing dalis dar nebuvo funkcionali.

Eclipse pasirenkame implementacijai šiame darbe būtent dėl nepriklausomumo, funkcionalumo ir panaudojimo galimybių, kurių patvirtinimas yra didelė bendruomenė naudojanti Eclipse.

2.5. Sistemos taikymo aplinkos ir vartotojų analizė

Sistema bus naudojama, kaip didesnio projekto posistemis. Sistema detaliau aprašyta V.Taujansko ir R.Butlerio straipsnyje „*Categories extraction for reuse in semantic applications and profile based recommendation service*“ [1] ir jos principinė schema pateikta 10 paveiksle. Vartotojams bus teikiama paslauga, kuri nereikalaus gilių kompiuterio ar dalykinės srities žinių apie Web 2.0 technologijas, ontologijas. Grafinės sąsajos aplinka ir funkcionalumas turėtų nesudaryti keblumų vartotojui, nes kuriamas funkcionalumas ir

[24] http://www.eclipse.org/
[25] http://ant.apache.org/
[26] http://www.junit.org/index.htm
[27] Standard Widget Tolkit, http://www.eclipse.org/swt/
[28] http://www.sun.com/
[29] http://www.gnu.org/software/classpath/

informacijos išdėstymas atitiks žmogaus mąstymą. Pateikiami rezultatai sudarys susietą tinklą, kuriame informacija siejama semantiniais ryšiais.

Kadangi šiame darbe nagrinėjamas turinio sudarymas duomenų agregavimo metodu, galutinis produktas bus ontologijos grafas aprašantis duomenis. Galutinis produktas bus pateiktas sistemos, kaip duomenys kitam posistemiui. Sistema projektuojama be vartotojo sąsajos.

Eksperimentui atlikti naudojamas programinis kodas vykdomas JAVA programavimo sistemoje Eclipse. Numatomi vartotojai bus testuotojai ir programuotojai dirbantys su projektu.

2.6. Probleminės srities sprendimo metodų literatūros šaltiniuose analizė

Ontologijų sudarymo problema įvairioje literatūroje analizuojama jau apie 30 metų. Informatikoje ši problema nagrinėjama apie 10 metų ir egzistuoja įvairių metodų bei literatūros susijusios su šia sritimi.

Literatūra susijusi su semantinio pasaulinio tinklo vystymu apibendrinta ir pateikiama knygose [10], [11], [12], [13], publikacijose [2], [14], [15] [16] [17] ir pagrindiniame internetiniame puslapyje http://www.w3.org/2001/sw/.

XML, RDF, OWL ir kitos technologijos taip pat analizuojamos daugybėje šaltinių, keletas pateikiama literatūros sąraše [18] [19].

Probleminė sritis susijusi su pranešimų sindikavimu, agregavimu ir jų valdymu plačiausiai analizuojama knygose [8] [25] [20]. Egzistuoja daugybė aplikacijų ir projektų teikiančių šias paslaugas – keletas pavyzdžių: FeednRead (fnr.sourceforge.net), Blogines (www.bloglines.com), LiveJournal (www.livejournal.com). Susijusios literatūros su šiais projektais taip pat yra labai daug, ši tematika sparčiai evoliucionuojama.

2.7. Galimų sprendimo variantų analizė

Siūlomas sprendimas siejamas su egzistuojančiais metodais skirtais pranešimų sindikavimui, agregavimui ir atvaizdavimui.

Sprendimai surenkant informaciją panaudojant sindikavimo formatu aprašytus šaltinius siūlomi [21] [22] magistro tezėse, [23] [24] straipsniuose. Dauguma skirti informacijai atvaizduoti ir pateikti vartotojui. Pateikti metodai nuo metodo siūlomo šiame darbe skiriasi tuo, kad nepanaudoja ontologija aprašytos struktūros duomenų saugojimui. Saugant duomenis ontologijos struktūra patogu perduoti kitoms posistemėms. Panaudojamų formatų specifikacijos, išsaugotos ontologijomis, informaciją talpinamą duomenyse leidžia pateikti interesantams. Tuo šis metodas yra unikalus.

Darbas remiasi jau išvystytų projektų, technologijų panaudojimu ir apjungimu, todėl galimos alternatyvos būtų kitų projektų ar kitų technologijų apjungimas. Kadangi naudojami metodai ir technologijos jiems realizuoti yra šiuo metu plačiausiai išvystyti, alternatyva nebūtų geresnė už šiame darbe siūlomą kombinaciją.

2.8. Analizės išvados

Atlikus analizę prieita prie apibendrintų išvadų apie esamą padėtį ontologijų sudarymo srityje. Išanalizuota kaip galima jas panaudoti duomenų agregavimo iš pasaulinio tinklo metodui realizuoti.

Išnagrinėta dalykinė sritis, pateikti pagrindiniai apibrėžimai ir veikimo principai. Numatyti įrankiai panaudojami įgyvendinimo etape.

Nustatyti vartotojai ir pagrindiniai veiklos proceso žingsniai, kurių pagrindu sistema bus projektuojama. Specifikuoti reikalavimai bei uždaviniai, kuriuos sistema turės atlikti. Apibrėžti pagrindiniai sistemos tikslai bei kontekstas.

3. Pasauliniame tinkle tiekiamų RSS, Atom ir kitų formatų duomenų agregavimo metodo įgyvendinimas

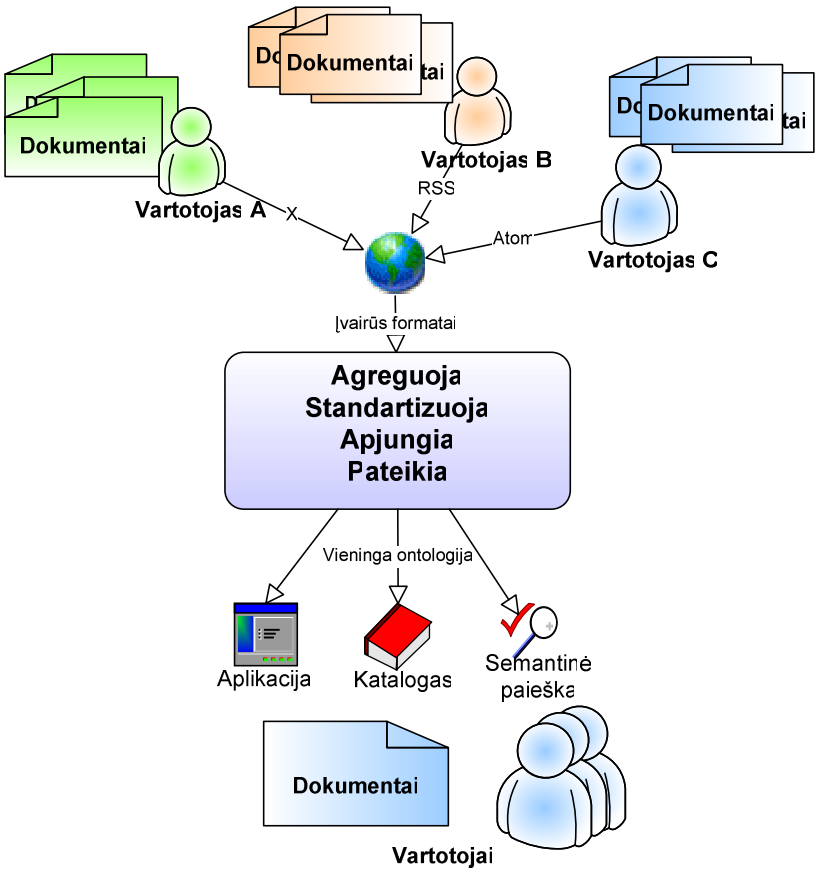
Keliama užduotis – įgyvendinti metodą sukuriant sistemą kaip didesnio semantinio portalo sudedamąją dalį. Metodas skirtas duomenų esančių semantiniame pasauliniame tinkle agregavimui ir apjungimui į ontologiją.

Šioje dalyje pateikiamas metodo įgyvendinimo projektas, patikslinami reikalavimai, tikslai ir veikimo principai. Kuriamos sistemos veikimas detalizuojamas veiklos, panaudojimo atvejų ir sekų diagramomis. Aprašomos dalykinės srities esybės, taisyklės. Pateikiami veikimo principai, sistemos architektūra, komponentai ir sąveiką tarp jų.

3.1. Aukščiausio lygio panaudojimo atvejis

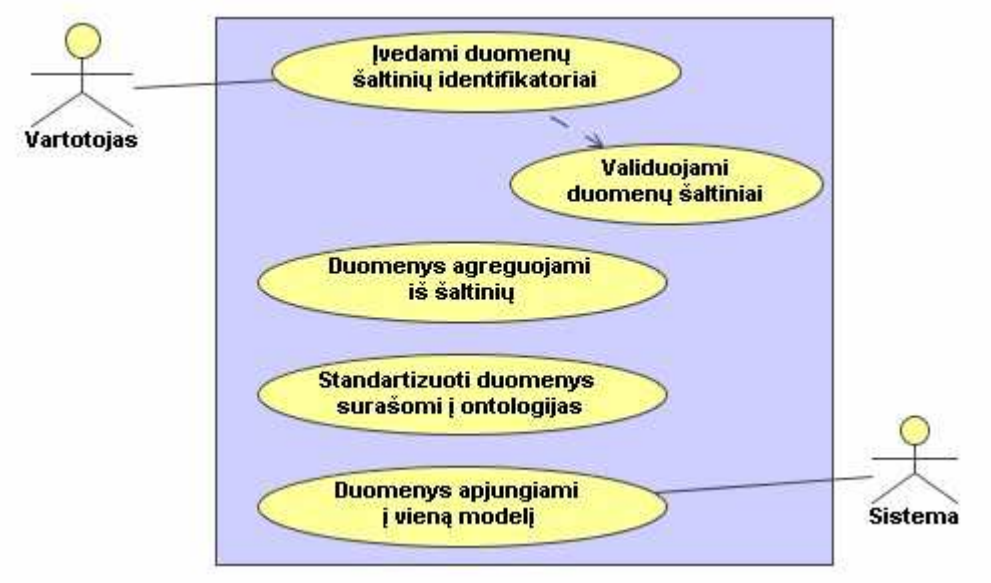
Panaudojimo atvejai pateikiami sistemos veikimui paaiškinti formaliai ir vaizdžiai. Pereinama nuo aukščiausio lygio bendro atvejo iki žemesnių, labiau specifinių uždavinių.

Bendra veikimo principą ir metodo panaudojimą atspindi schema ne UML notacijoje pateikiama 17 paveiksle. Šioje schemoje vaizduojama, kad problemos prasideda kai skirtingi vartotojai sindikuodami duomenis kuria dokumentus ir publikuoja pasauliniame tinkle. Dokumentai gali būti publikuojami skirtingais formatais: Atom, RSS skirtingomis versijomis ir kitais sindikavimo formatais. Todėl atsiranda problema tuos duomenis vieningai nuskaityti, nes nuolatos keičiasi formatai, atsiranda naujų. Vartotojai neretai skirtingai interpretuoja panaudojimo galimybes ir pateikia formatus šiek tiek iškraipytus, pritaikytus savo poreikiams. Aplikacijos, kurios agreguoja duomenis, jų nestandartizuoja, o tik suskirsto pagal tipą ir naudoja skirtingus metodus jiems pateikti. Dokumentų ir jų informacijos panaudojimas tolimesniems duomenims tokiu atveju yra apribojamas. Šios aplikacijos pagrindinis tikslas yra agreguoti skirtingus formatus, juos standartizuoti, apjungti, ir pateikti įvairiems tolimesniems panaudojimams aplikacijoms, katalogui sudaryti ar semantinėms užklausoms grafe vykdyti. 15 paveiksle tai pavaizduota išėjimo rodykle iš sistemos su pavadinimu „Vieninga ontologija“. Tokiu būdu galinis vartotojas gaus visus Internete publikuotus dokumentus, nesvarbu kokiu formatu ir kokio autoriaus jie buvo sudaryti.



17 pav. Duomenų agregavimo metodo apibendrinta schema

Sistemos modelis atvaizduotas panaudojimo atvejų diagrama UML notacijoje 18 paveiksle. Jame atvaizduoti 4 pagrindiniai veiksmai atliekami sistemos realizuojant metodą.



18 pav. Sistemos aukščiausio lygio panaudojimo atvejis

Panaudojimo atvejai:

- **PA 1:** Įvedami duomenų šaltinių identifikatoriai – tai atvejis, kai vartotojas į sistemą įveda URL adresą pasauliniame tinkle identifikuojantį duomenų šaltinį. Tokių identifikatorių gali būti įvedama daugiau nei 1, gali būti importuojami iš sąrašų. Tik įvedus duomenų šaltinius, jie turi būti validuojami ir toliau bus naudojami tik validūs duomenų šaltiniai.
- **PA 2:** Duomenys agreguojami iš šaltinių – šiame etape sistema panaudoja ROME įrankį (žiūr.: 2.4.3 skyrių) duomenims iš pasaulinio tinklo parsiųsti ir nuskaityti patalpinant juos į aprašančius objektus.
- **PA 3:** Standartizuoti duomenys surašomi į ontologijas – šiame etape duomenys iš juos aprašančių objektų perkeliami į ontologijos schemas pagal kiekvieno formato sudarytas ontologijų schemas.
- **PA 4:** Duomenys apjungiami į vieną modelį – panaudojimo atvejis, kai duomenis saugantys ontologijų grafai suliejami į vieną panaudojant apjungimo operaciją.

Sekančiuose poskyriuose detalizuojami panaudojimo atvejai.

3.1.1. Duomenų šaltinių identifikatorių įvedimas

Įvedami duomenų šaltinių identifikatoriai, kurie nurodo kokiu adresu Internete patalpinami duomenų šaltiniai. Šaltinio identifikatorių aprašo URL adresas.

URL adresų aprašančių duomenų šaltinius pavyzdžiai:

```
http://www.tbray.org/ongoing/ongoing.atom
http://groups.google.com/group/IFM-14/feed/atom_v1_0_msgs.xml
http://bblfish.net/blog/blog.atom
http://www.delfi.lt/rss/feeds/lithuania.xml
```

Duomenų šaltinių skaičius nėra ribojamas ir programa veiks tol, kol parsiųs duomenis iš visų šaltinių iteruojant per jų sąrašą. Kartą įvedus šaltinį jis pasilieka sąraše ir antrą kartą įvedinėt nereikia. Prieš agreguojant duomenis, jų duomenų šaltiniai patikrinami. Tuo tikslu naudojama kita sistema kaip įrankis – FeedValidator (žiūr.: 2.3.1 skyrių).

5 lentelė. PA Duomenų šaltinių identifikatorių įvedimas

PA ID	PA1	
PA pavadinimas	Įvedami duomenų šaltinių identifikatoriai	
Aktoriai	Vartotojas, Sistema	
Aprašymas	Vartotojas įveda duomenų šaltinių URL adresą	
„Po“ sąlygos	Duomenų šaltinį identifikuojantis adresas atsiduria identifikatorių sąraše	
Pagrindinis scenarijus		
Žingsnio Nr.	*Vartotojo veiksmas*	*Sistemos reakcija*
P-1.	Įvesti šaltinio URL	Siųsti užklausą į FeedValidator ar teisingai suformuota šaltinis
P-2.		Jei FeedValidator patvirtina validumą, įtraukti URL į sąrašą.
Išimtys (klaidos)		
Klaidos sąlyga		*Sprendimas*
Neteisingai suformuotas, nevaliduojamas FeedValidator		Pranešti vartotojui ir neįtraukti į sąrašą

3.1.2. Duomenų agregavimas iš šaltinių

Duomenys agreguojami iš šaltinių panaudojant ROME įrankio galimybes.

Perduodant duomenų šaltinio identifikatorių per parametrus, duomenų failas yra parsiunčiamas iš Interneto ir nuskaitoma jame esančių duomenų struktūra patalpinama į duomenų objektus atitinkančius formato specifikacijas (žiūr. 1 ir 2 žingsnis 2.4.3 skyriuje).

6 lentelė. PA Duomenų agregavimas

PA ID	PA2	
PA pavadinimas	Duomenys agreguojami iš šaltinių	
Aktoriai	Sistema	
Aprašymas	Duomenys parsiunčiami, nuskaitomi ir sudedami į aprašančius objektus	
„Prieš“ sąlygos	Perduodami validuoti duomenų šaltinių identifikatoriai	
„Po“ sąlygos	Suformuoti duomenis aprašantys objektai	
Pagrindinis scenarijus		
Žingsnio Nr.	*Vartotojo veiksmas*	*Sistemos reakcija*
P-1.		Duomenys parsiunčiami iš Interneto į XML struktūros failą
P-2.		Elementai perskaitomi iš XML naudojant ROME įrankį
P-3.		Elementams sukuriami objektai pagal ROME struktūrą
P-4.		Objektai su duomenimis gražinami kaip rezultatas
Išimtys (klaidos)		
Klaidos sąlyga		*Sprendimas*

3.1.3. Duomenų standartizavimas surašant į ontologijas

Standartizuoti duomenys surašomi į ontologijas – šiame etape duomenys iš juos aprašančių objektų perkeliami į ontologijos schemas pagal kiekvieno formato sudarytas ontologijų schemas. Ontologija apibendrinanti ir objektai priskiriami pagal tam tikras taisykles, todėl atsiranda standartizavimas.

Šiame panaudojimo atvejyje nėra įrankio atliekančio duomenų perkėlimą iš ROME objektų į ontologijas pagal JENA instrumentinės priemonės taisykles. Kyla poreikis kurti modulį atliekantį šį perkėlimą. Ontologijos modelis su duomenimis išsaugomas automatiškai, panaudojant JENA galimybes.

7 lentelė. PA Duomenų įtraukimas į ontologiją

PA ID	PA3	
PA pavadinimas	Standartizuoti duomenys surašomi į ontologijas	
Aktoriai	Sistema	
Aprašymas	Panaudojant JENA instrumentinę priemonę ir duomenų aprašą JAVA klasėmis, ontologijos užpildomos duomenimis	
„Prieš“ sąlygos	1. Ontologijos tuščia struktūra turi egzistuoti duomenų bazėje 2. Sugeneruotos JAVA klasės turi atitikti ontologijos struktūrą 3. Duomenų objektai turi būti užpildyti	
„Po“ sąlygos	Ontologija praplečiama naujais egzemplioriais	
Pagrindinis scenarijus		
Žingsnio Nr.	*Vartotojo veiksmas*	*Sistemos reakcija*
P-1.		Ontologijos modelis nuskaitomas iš duomenų bazės JENA instrumentinėmis priemonėmis į laikiną modelį
P-2.		Duomenys analizuojami ir egzistuojančių analogai pagal standartus kuriami ontologijos modelyje
P-3.		Ontologijos modelis automatiškai išsaugomas duomenų bazėje su užpildytais duomenimis
Išimtys (klaidos)		
Klaidos sąlyga		*Sprendimas*

3.1.4. Duomenų ontologijų apjungimas į vieną modelį

Duomenų apjungimas į vieną ontologijos grafą – duomenis saugantys ontologijų grafai suliejami į vieną panaudojant apjungimo operaciją.

Su objektais saugančiais RDF grafus (žiūr. 2.1.3.2) JENA instrumentinėje priemonėje atliekama apjungimo operacija. Išgavus modelį MA, kuris aprašo Atom formato pranešimų ontologiją, išgauname kito formato pranešimų ontologijos modelį MB, apjungę juos su operacija UNION gausime modelį M, kuris ir bus metodo realizacijos rezultatas.

$$M = MA \cup MB$$

8 lentelė. PA Duomenis aprašančių ontologijų apjungimas

PA ID		PA4
PA pavadinimas		Duomenų apjungimas į vieną ontologijos grafą
Aktoriai		Sistema
Aprašymas		Duomenis saugantys ontologijų grafai suliejami į vieną grafą panaudojant apjungimo operaciją.
Pagrindinis scenarijus		
Žingsnio Nr.	*Vartotojo veiksmas*	*Sistemos reakcija*
P-1.	Užklausa bendrai reikalingai informacijai	Išgaunami visų formatų ontologijos grafai
P-2.		Grafai apjungiami į vieną
P-3.		Apjungtas ontologijų grafas grąžinamas kaip rezultatas
P-4.	Vartotojas naršo nagrinėja informaciją	
Išimtys (klaidos)		
Klaidos sąlyga		*Sprendimas*

3.2. Reikalavimų specifikavimas metodo realizacijai

Pagrindinius reikalavimus ir sistemos tikslus padiktavo semantinio pasaulinio tinklo samprata aptariama straipsnyje [2]. Semantinio pasaulinio tinklo idėja yra sukurti bendrą universalią terpę duomenų apsikeitimui. Apjungti tarpusavyje susijusį asmeninės informacijos valdymą, korporacijų programų integraciją bei globalios informacijos apsikeitimą tarp komercinių, mokslinių, kultūrinių ir kitokio pobūdžio programų. Patenkinti poreikį atsirandantį semantiniame pasauliniame tinkle – kompiuteriams suprasti informaciją, be žmogaus įsikišimo.

Plintančios paieškos technologijos tarp vis didesnio informacijos kiekio reikalauja informacijos semantikos bei konkretumo. Atsiradę RSS [25], Atom ir kiti sindikavimo formatai standartizuotam informacijos apsikeitimui pasauliniame tinkle naudojant sindikavimo-agregavimo mechanizmą, kelia informacijos standartizavimo reikalavimus.

Duomenys semantiniame pasauliniame tinkle dažniausiai pateikiami ontologijomis grįstomis duomenų schemomis. Ontologijų formalus aprašas paremtas RDF ir OWL [26] kalba, kuria turi būti aprašomi siūlomą metodą realizuojančios sistemos duomenys.

Detalizuojant bendrus semantinio pasaulinio tinklo reikalavimus suskirstome juos į keletą punktų.

3.2.1. Reikalavimai sistemai

Sistema turi turėti aiškiai apibrėžtas ribas bei turi būti sudaryta iš posistemių su aiškiai apibrėžtais interfeisais, per kuriuos jos funkcijos ir rezultatai būtų įjungiami į semantinį portalą arba pateikiamos kaip paslaugos kitoms sistemoms.

Duomenys aprašomi ontologijų kalba RDF schemomis su OWL (*Web Ontology Language*) elementais.

Pradiniai duomenys talpinami pasauliniame tinkle ir yra formuojami pagal Atom, RSS ir kitų formatų specifikacijas.

Veikimas turi būti kiek galima automatizuotas – galutinis produktas pateikiamas vartotojui, be žmogiškųjų resursų sąnaudų.

Sistema vartotojui grafinės sąsajos neteikia, bet atliekant eksperimentą leidžia stebėti pagrindinius veikimo žingsnius ir pateikia skaičiavimų įverčius.

Sistema pateikia kitoms programoms aiškius interfeisus ir duomenų tipus.

3.2.2. Reikalavimai sistemos architektūrai

Architektūra turi būti realizuota paskirstant jos komponentus į veiklos sluoksnius pagal n-sluoksnių metodiką. Išskiriami 3 sluoksniai, kuriuose numatomi pagrindiniai reikalavimai sistemai pateikiami 9 lentelėje.

9 lentelė. Reikalavimai pagal veiklos sluoksnius

Sluoksnis	**Reikalavimas**
Atvaizdavimo	Kliento aplikacija turi būti pateikiama per Interneto naršyklę.
	Visose naršyklėse atvaizdavimas ir galimybės turi būti vienodos.
	Grafinė sąsaja projektuojama pagal naujas Internetines technologijas.
	Grafinėje sąsajoje nėra realizuota veiklos logikos. Visi veiksmai atliekami panaudojant veiklos sluoksnio komponentus per jų interfeisus.
Veiklos logikos	Sistema realizuota JAVA technologijom turi būti talpinama Servletų pagrindu.
	Pradiniai duomenys turi būti validuojami.
	Panaudojamas įrankis duomenų agregavimui
	Panaudojama instrumentinė priemonė darbui su ontologijomis.
	Dirbama su duomenų objektais, panaudojant duomenų sluoksnį.
Duomenų	Duomenys publikuojami Pasauliniame tinkle Atom 1.0, RSS 1.0, RSS 2.0 ar panašios paskirties pranešimų formatais.
	Duomenys ir tarpiniai rezultatai saugomi RDF duomenų bazėje.
	Duomenys aprašomi klasėmis, kurios gali būti generuojamos ir modifikuojamos.

3.2.3. Reikalavimai duomenis aprašantiems failams

Siūlomas metodas skirtas agreguoti ir formuoti skirtingų ontologijų grafams ir pradiniame žingsnyje duomenys turi būti parsiunčiami iš PT esančių resursų failų.

Kadangi ROME paketas – naudojamas pranešimų agregavimui – neveikia su neteisingai suformuotais resursų failais, tai duomenys turi būti validuoti. Šis reikalavimas galioja bendrai visiems duomenų failams nesvarbu kokiu formatu jie pateikti.

Validavimui, bet kurio sindikuoto duomenų formato failams tinka Internete arba lokaliai veikianti FeedValidator aplikacija (http://feedvalidator.org/), kuri patikrina ar pranešimo failas suformuotas taisyklingai. Tikrinama ar sintaksė neprieštarauja taisyklėms bei ar duomenys surašyti pagal formatų specifikacijas.

FeedValidator apjungia trijų tipų pranešimų specifikacijas: RSS 1.0, RSS 2.0 ir Atom 1.0. Atsiradus naujam formatui šio įrankio kūrėjai esamą programos kodą numato keisti pagal specifikacijas.

Siūlomas metodas numato galimybę agreguoti įvairius duomenų šaltinių failų formatus. Failo struktūroje pateikiamiems duomenims ypatingų reikalavimų nėra. Jei jie buvo suformuoti remiantis specifikacijomis ir validatorius juos patvirtino, kaip teisingus, vadinasi juos galima apdoroti. Validatoriaus reikalavimai sindikavimo formatų failams pateikiami nuoroda http://feedvalidator.org/docs/ .

3.2.4. Nefunkciniai sistemos reikalavimai ir apribojimai

Sistemai keliami nefunkciniai reikalavimai aptariami 10 lentelėje.

10 lentelė. Sistemai keliami nefunkciniai reikalavimai

Reikalavimas	**Išpildymas**
Reikalavimai standartams	Sistemos darbo rezultatas turi atitikti ontologijų bylų standartus (OWL, RDF ar panašiai), nustatytus W3C konsorciumo. Pagrindinis duomenų standartas yra Atom 1.0 apibrėžtas RFC 4287.
Reikalavimai veikimui	Sistemos darbo kokybė svarbesnė už darbo laiką, tačiau turi būti proto ribose apibrėžtas laiko intervalas (minučių skaičius tūkstančio žodžių tekstui apdoroti). Taip pat turi būti galimybė pasirinkti pageidaujamą ontologijos išsamumą (sudėtingumo lygį), nuo kurio priklausytų taisyklių išskyrimo laikas.
Reikalavimai sąveikai/suderinamumui su kitomis sistemomis	Sistema nebus derinama su kitomis sistemomis. Ji bus kaip vieno portalo posistemė ir turi būti galimybė prijungti kitas posistemes tiek pateikiant duomenis, tiek gaunant rezultatus. Aiškiai apibrėžti

	interfeisai, duomenų klasės.
Reikalavimai vartotojo sąsajai	Kadangi sistema reikalaus minimalaus vartotojo įsikišimo ontologijų išskyrimo procese – vartotojo sąsajai nėra apibrėžiami reikalavimai. Vartotojo sąsajoje bus pateikiami rezultatai. Svarbiausias rezultatų pateikimo reikalavimas, kad būtų galima intuityviai surasti informaciją.
Reikalavimai plėtimui	Sistema privalo turėti interfeisus papildomų modulių prijungimui ateityje.

3.2.5. Reikalavimai rezultatui ir kokybės kriterijai

Metodą realizuojančios sistemos rezultatas – vieninga ontologija pateikiami duomenys informacijos pakartotiniam panaudojimui.

Rezultato kokybę atspindės šie faktoriai:

- Pilnumas – ar visa informacija, kuri buvo pateikta duomenyse, yra pateikiama ir vartotojui sistemos rezultatuose.
- Teisingumas – informacija nėra iškraipoma ir pateikiama taip, kaip buvo suformuota.
- Informatyvumo – naujienų pranešimų kategorizavimas atliekamas siekiant suteikti daugiau informacijos vartotojui. Jei rezultatas yra informatyvus tai vadinasi yra kokybiškas.
- Panaudojamumas – priklauso nuo to, kaip rezultate gautą informaciją bus galima pakartotinai panaudoti integruojant sistemą į aukštesnius lygius.

Šie rezultato kokybės kriterijai ne visada gali būti teisingai apibrėžti kiekybiškai. Jie nustatomi atsižvelgiant į iškeltus funkcinius ir nefunkcinius reikalavimus.

Kiekybiškai įvertinami rezultato kokybės kriterijai:

- Sistemos sugaištas laikas ir sunaudoti resursai rezultatui pasiekti
- Užklausų kiekis ir duomenų bazės apkrovimas atliekant uždavinį

3.3. Duomenų agregavimo metodą realizuojančios sistemos projektas

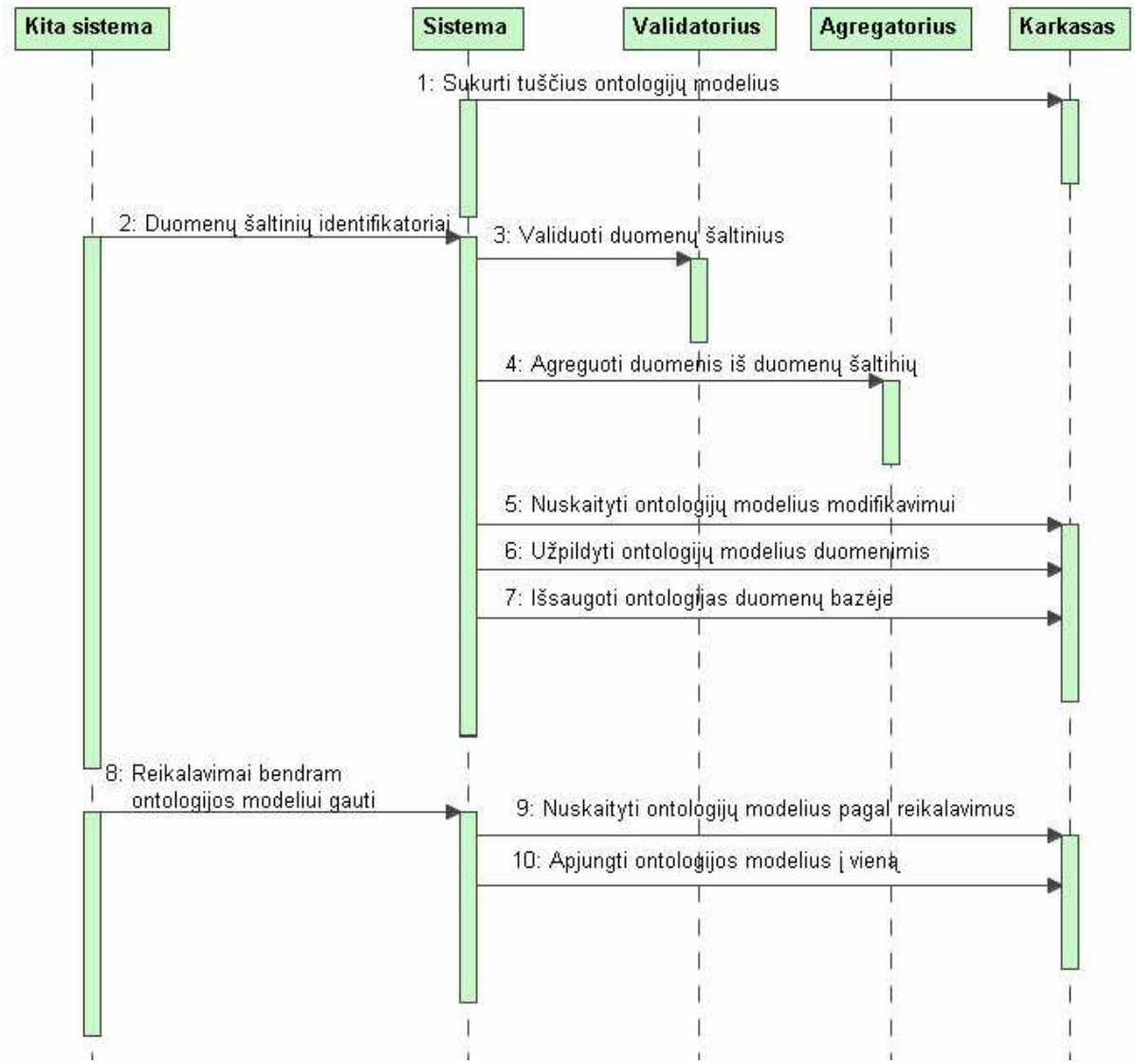
3.3.1. Projekto tikslas

Suprojektuoti sistemą duomenų agregavimui iš skirtingų sindikavimo formatų pagal juos aprašančias ontologijas. Sistema turi realizuoti siūlomą metodą ir atitikti visus sistemos funkcionalumui keliamus reikalavimus.

3.3.2. Panaudojimo atvejų sekų diagramos

Panaudojimo atvejų raiškumą praplečia sekų diagramos. 19 paveiksle pateikiamoje sekų diagramoje vaizduojami trys pagrindiniai panaudojimo atvejai:

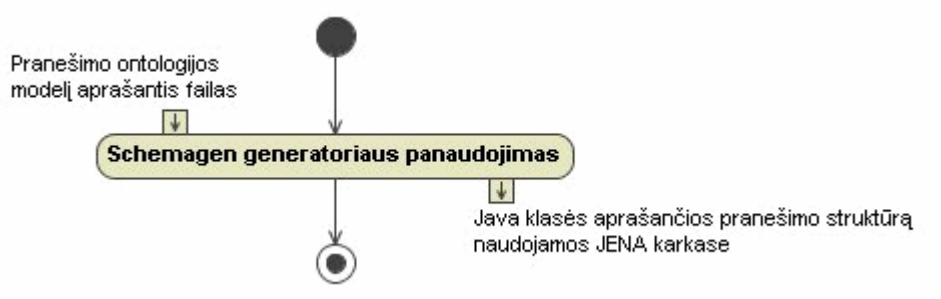
- Sukurti tuščius ontologijų modelius panaudojant instrumentinę priemonę darbui su ontologijomis.
- Įvedamos duomenų šaltinių nuorodos ir sekantys veiksmai bei atsakomoji reakcija iš sistemos.
- Sistemai pateikiami bendri reikalavimai bendram modeliui ir sekančiais veiksmais suformuojamas bei pateikiamas atsakas su rezultatais.



19 pav.: Sekų diagrama pagal aukščiausio lygio panaudojimo atvejį

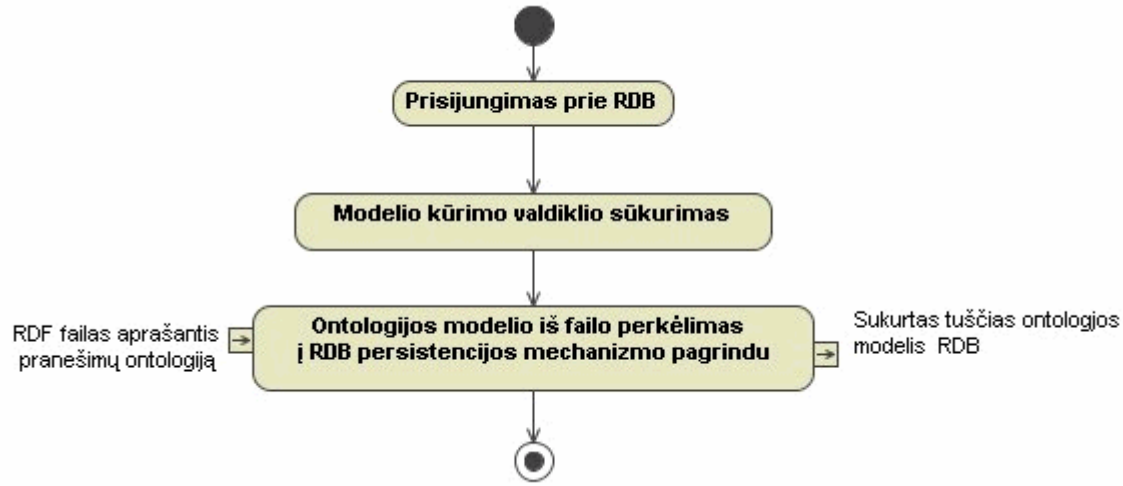
3.3.3. Bendri pranešimų gyvavimo ciklo modeliai

Sistema projektuojama taip, kad galėtų agreguoti įvairių formatų pranešimuose publikuojamus duomenis. Diagramose 20, 21 ir 22 pavaizduotas apibendrintas visų formatų pranešimų gyvavimo ciklas.



20 pav. Pranešimo objektus aprašančių klasių sugeneravimas

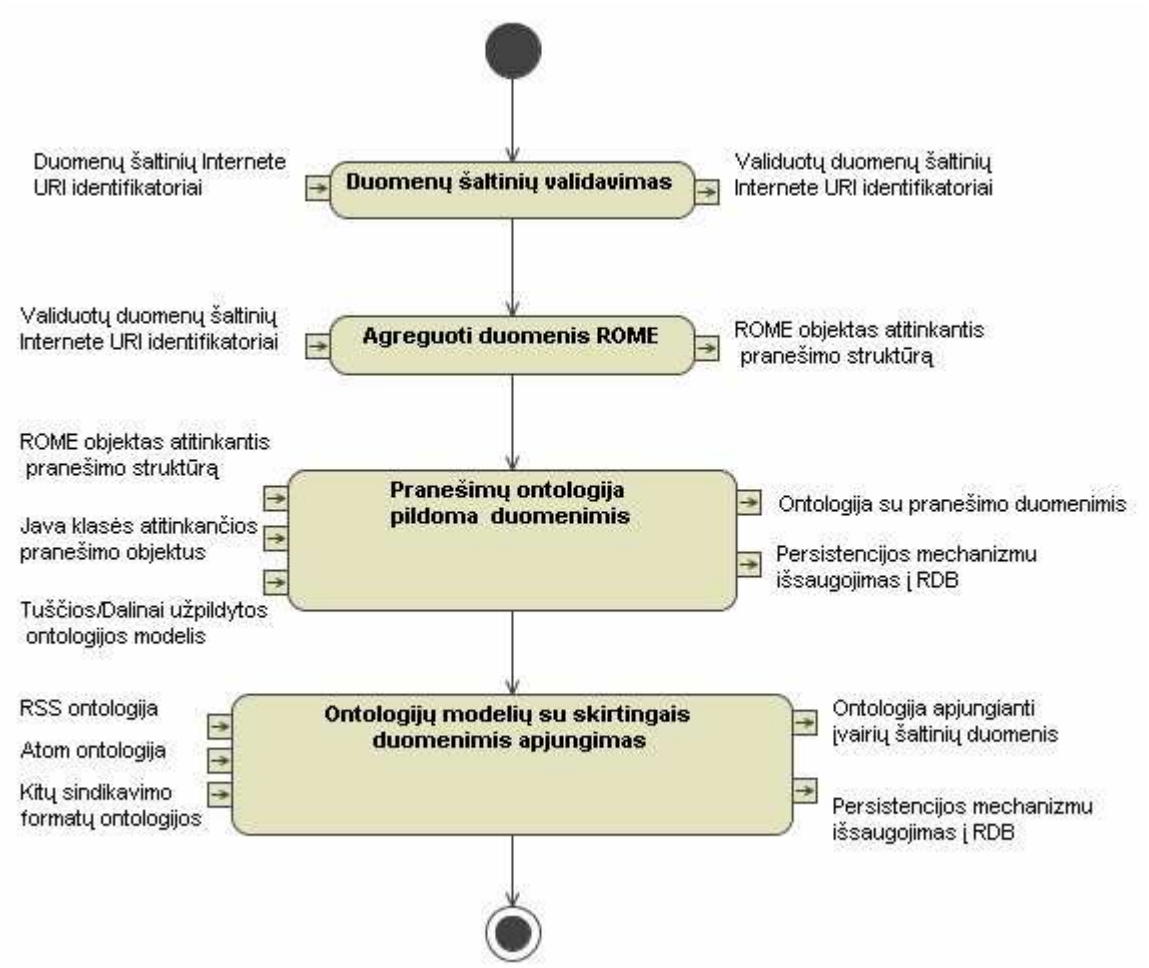
Iš specifikacijų aprašytų RDF grafu gaunamos Java klasės naudojamos JENA instrumentinėje priemonėje aprašant pranešimų objektus.



21 pav.: Ontologijos struktūros atitinkančios pranešimo formatą sukūrimas RDB

Iš tų pačių pranešimo specifikacijų RDF aprašo sukuriamas tuščias ontologijos modelis reliacinėje duomenų bazėje panaudojant JENA instrumentinės priemonės funkcijas ir duomenų objektų saugojimo į RDB (angl. *persistency*)[30] mechanizmą.

[30] http://jena.sourceforge.net/DB/creating-db-models.html

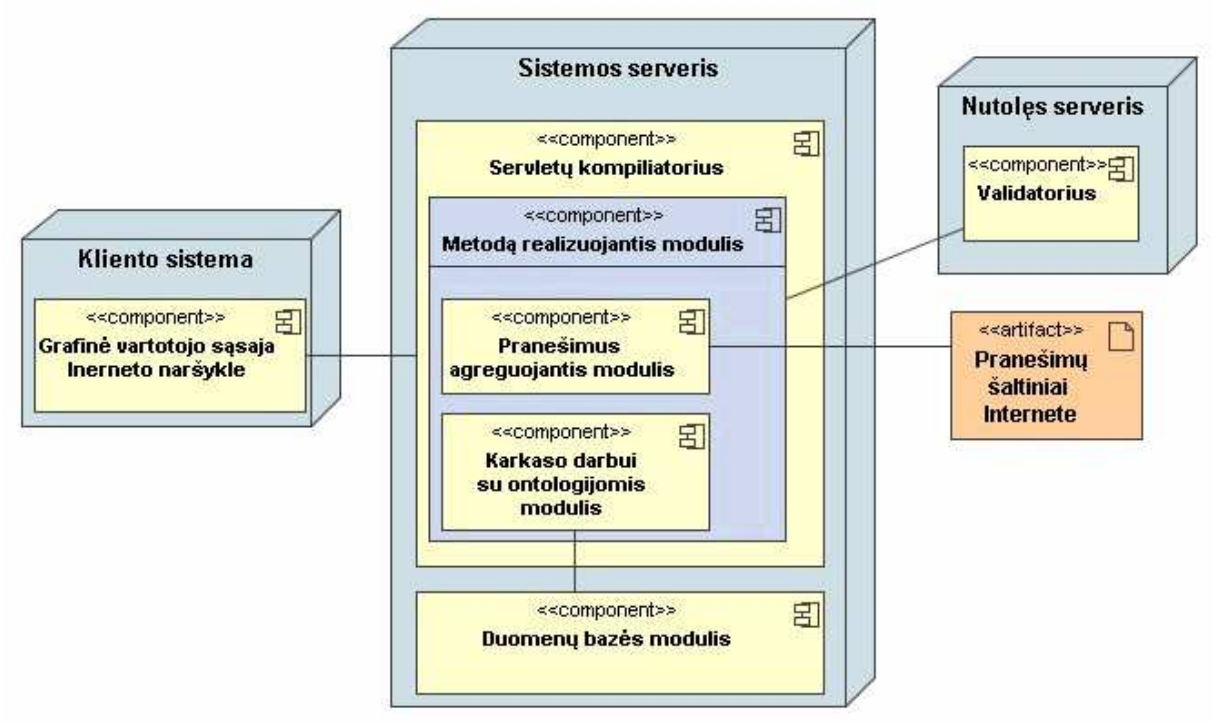


22 pav.: Pranešimo parsiuntimas ir ontologijos suformavimas

Po duomenų šaltinio validavimo sistema inicijuoja duomenų parsiuntimą ROME įrankio pagalba ir gauna užpildytus objektus atitinkančius skirtingų pranešimų struktūrą. Duomenų objektai analizuojami ir kuriami jų atitikmenys nuskaitytame ontologijos modelyje. Ontologijos modelis nuskaitytas naudojant objektų saugojimo į RDB mechanizmą, todėl pildomas naujais objektais automatiškai juos išsaugo duomenų bazėje. Vėliau kiekvienas iš modelių apjungiamas ir gaunama vieninga ontologija su tais pačiais duomenimis, kurie buvo parsiųsti iš pasaulinio tinklo.

3.3.4. Sistemos architektūra

Specifikavus panaudojimo atvejus 23 paveiksle pateikiama bendra architektūros diagrama.



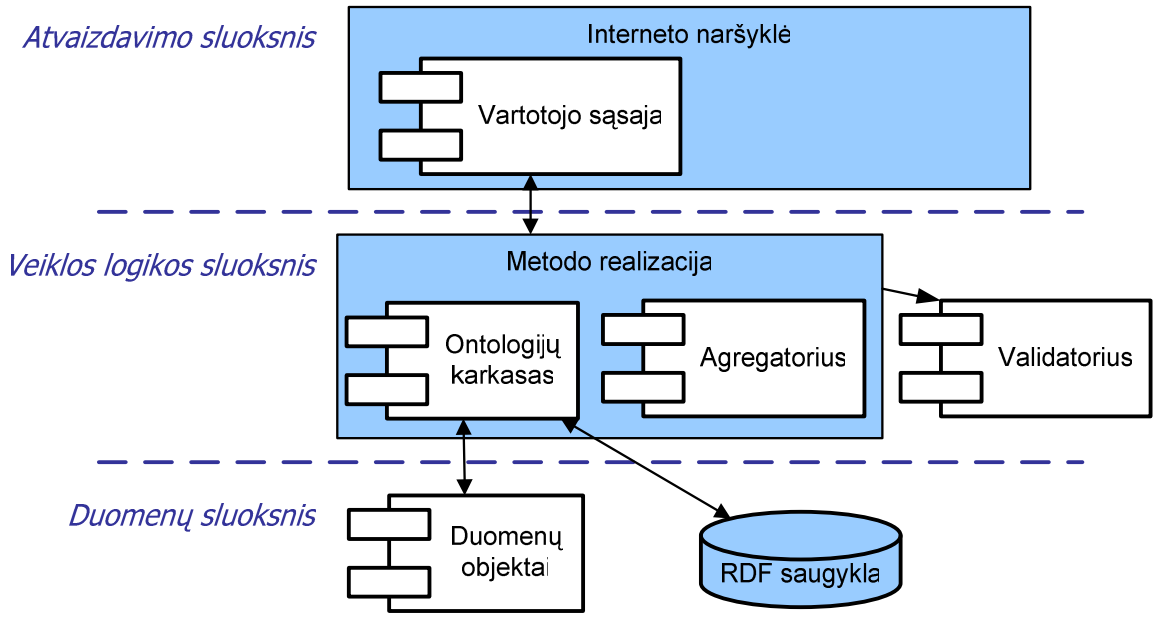
23 pav.: Sistemos architektūros modelis

Pagrindiniai architektūriniai sprendimai atsispindi 23 paveiksle pateiktame modelyje:

- Kliento sąsajos komponentas naudoja serveryje esančio servletų kompiliatoriaus pateikiamus duomenis
- Sistema realizuota serveryje, kuriame vykdomas programinis kodas sudarantis sistemą
- Instrumentinė priemonė darbui su ontologijomis atlieka veiksmus: sukuria, atnaujina, nuskaito, naudojant saugojimo į RDB mechanizmą išsaugo ontologijas duomenų bazėje
- Pranešimus agreguojantis modulis parsiunčia Internete esančius duomenis
- Sistema turi galimybę patikrinti pranešimus prieš parsiunčiant ar jie suformuoti teisingai, kad išvengti trikdžių. Tam per interfeisą panaudojamas validatorius.

3.3.5. *Trijų sluoksnių architektūros panaudojimas*

Pagal reikalavimus architektūrai iškeltus 3.2.2 skyriuje, sistemos komponentai paskirstomi į veiklos sluoksnius taip kaip parodyta 24 paveiksle.



24 pav.: Sistemos komponentų paskirstymas pagal trijų sluoksnių architektūrą

Pagal N-sluoksnių architektūros principus išskiriame 3 lygius:

1. **Atvaizdavimo sluoksnis** – atsakingas už vartotojo paslaugas. Realizuoja vartotojo sąsają, kad vartotojas galėtų pasirinkti norimas atlikti funkcijas ir matytų pasirinktą informaciją.
2. **Veiklos logikos sluoksnis** – atsakingas už veiklos paslaugas. Užtikrina sistemos funkcionalumą, t.y. vartotojo sistemoje pasirinktų funkcijų atlikimą. Vartotojo paslaugų elementai – ribinės klasės sąveikauja su atitinkamais valdikliais, kurie realizuoja metodus, atitinkančius tam tikrą sistemos funkciją.
3. **Duomenų sluoksnis** – atsakingas už duomenų paslaugas. Saugo sistemai reikalingus duomenis. Duomenų įterpimas, atnaujinimas, šalinimas ir nuskaitymas iš duomenų bazės realizuojamas programiniame kode operuojant duomenų objektais ir taip užtikrinant programinį tvarkingumą.

Uždavinys nėra tradicinės IS kūrimo atvejis, nes veiklos lygyje susiduriama su ontologijos specifikacijomis. Duomenų sluoksnis turi būti specialiai pritaikomas darbui su ontologijomis paremta duomenų baze. Tam generuojant duomenų objektus aprašančias klases turėtų būti įrankis pateikiamas kartu su JENA instrumentine priemone. Turint objektų RDF

aprašą (ontologiją) ir atlikus transformaciją gaunamos objektus aprašančios klasės, kurios naudojamos veiklos logikos sluoksnio komponentuose atliekant operacijas su ontologijos grafu, transformuojant objektus ir kt.

3.3.6. Pagrindiniai sistemos komponentai

Metodą realizuojančią sistemą tikslinga išskaidyti į komponentus – dekompozicijos principais. Tokiu būdu įvedamos aiškios veikimo ribos bei apibrėžiami ryšiai tarp posistemių ir sudaroma galimybė programuoti ir testuoti atskiras sistemos dalis.

Pagrindinius sistemos komponentus galima paskirstyti pagal veiklos sluoksnius:

11 lentelė. Pagrindiniai sistemos komponentai

Sluoksnis	**Komponentas**	**Aprašymas**
Atvaizdavimo	Vartotojo sąsajos	Realizuoja vartotojo sąsają, kad vartotojas galėtų pasirinkti norimas atlikti funkcijas ir matytų pasirinktą informaciją. Pateikia pranešimus, priima įvedamus duomenis
Veiklos logikos	Metodo realizacija	Apjungia vidinius komponentus ir organizuoja darbą, aukščiausiame logikos lygmenyje.
	Instrumentinė priemonė darbui su ontologijomis	Pateikia priemones darbui su ontologijos modeliais: sukurti, nuskaityti, atnaujinti. Atsakingas už duomenų lygio kontrolę.
	Agregatoriaus įrankis	Parsiunčia pranešimus iš pasaulinio tinklo ir sukuria objektus, kurie vėliau bus perkeliami į ontologijos modelį.
	Validatoriaus įrankis	Pagal nurodytą URL patikrina ar pasauliniame tinkle esantis pranešimo šaltinis suformuotas teisingai.
Duomenų	Duomenų klasės	Duomenų objektus aprašančios klasės. Naudojamos kuriant objektus veiklos logikos lygmenyje ir atitinkančios RDF modelio reikalavimus instrumentinėje priemonėje darbui su ontologijomis.
	RDF saugykla (duomenų bazė)	Duomenų bazė, kurioje saugomi RDF modeliai.

3.3.7. Komponentą realizuojantys įrankiai

Šiame darbe numatomiems realizuoti komponentams parinkti įrankiai išnagrinėti analitinėje dalyje. Komponentai su jiems numatytais įrankiais pateikiami lentelėje.

12 lentelė. Moduliams realizuoti naudojami įrankiai

Modulis	**Įrankis realizacijoje**
Metodo realizacija	JAVA programavimo kalba Eclipse SDK aplinkoje
Instrumentinė priemonė darbui su ontologijomis	JENA
Agregatoriaus įrankis	ROME
Validatoriaus įrankis	FeedValidator
Duomenų klasės	Sugeneruotos su Schemagen (JENA) generatorium JAVA programavimo kalba
RDF saugykla (duomenų bazė)	Postgre SQL

3.3.8. Duomenų bazės modelis ontologijos struktūrai išsaugoti

Dalykinės srities duomenys bus saugomi ontologijos pavidalo RDF grafe. RDF grafą galima saugoti faile, tačiau toks būdas nėra efektyvus dėl laiko sąnaudų skaitant, modifikuojant grafą. Todėl parenkama duomenų bazė, kurios palaikymas ir visos operacijos su duomenimis realizuotos JENA instrumentinėje priemonėje.

Duomenų bazėje pagrindą sudaro lentelė, kurios įrašai turi 4 atributus:

1. subjektas;
2. predikatas;
3. objektas;
4. grafo numeris.

Čia 1, 2 ir 3 atitinka RDF esminius atributus aprašytus 2.1.3.2 skyrelyje. O grafo numeris identifikuoja, kuriam iš duomenų bazėj saugomų grafų šis subjektas su predikatu ir objekto reikšme priklauso.

SQL sakinys kuriantis duomenų bazės lentelę:

```
CREATE TABLE jena_g1t1_stmt
(
  subj varchar(250) NOT NULL,
  prop varchar(250) NOT NULL,
  obj varchar(250) NOT NULL,
  graphid int4
)
```

Kadangi objekto reikšmė gali būti kur kas didesnė už 250 simbolių, įvedama kita lentelė sauganti ilgas reikšmes. Įrašas turi 4 atributus:

1. Id – pirminis raktas;
2. head;
3. chksum;
4. tail – ilgą reikšmę (dažniausiai tekstą) talpinantis atributas;

SQL sakinys kuriantis duomenų bazės lentelę:

```
CREATE TABLE jena_long_lit
(
  id serial NOT NULL,
  head varchar(250) NOT NULL,
  chksum int8,
  tail text,
  CONSTRAINT jena_long_lit_pkey PRIMARY KEY (id)
)
```

Grafams aprašyti duomenų bazėje lentelė, kurioje įrašai turi atributus:

1. Id – pirminis raktas
2. name – grafo pavadinimas

SQL sakinys kuriantis duomenų bazės lentelę:

```
CREATE TABLE jena_graph
(
  id serial NOT NULL,
  name varchar(1024),
  CONSTRAINT jena_graph_pkey PRIMARY KEY (id)
)
```

Kiekvienam grafui sukuriama nauja lentelė kurioje saugoma jo objektai su savybėmis ir jų reikšmėmis.

Kitos lentelės skirtos JENA instrumentinės priemonės objektų saugojimo į RDB mechanizmo sisteminėms reikšmėms talpinti ir metodo realizacijai jokios įtakos neturi.

Tokia paprasta duomenų bazė leidžia išsaugoti bet kokio sudėtingumo RDF modelius, OWL kalba aprašytas ontologijas ir duomenis jose.

3.4. Metodą realizuojančios sistemos projektavimo išvados

Projektavimo objektas buvo pasirinktas dalykinėje srityje išnagrinėjus naujas informacines technologijas ir naujos kartos technologijas naudojamas įgyvendinant SPT viziją.

Suprojektuotas objektas – siūlomą metodą realizuojanti sistema.

Sistema buvo projektuojama kaip viena iš sudedamųjų portalo dalių. Bendrai apibrėžti metodą realizuojantys apibendrinti uždaviniai ir taikymo sritis. Uždaviniai bus detaliai aprašomi realizacijos stadijoje, priklausomai nuo įrankių panaudojimo ir pranešimų formatų savybių.

Numatytus panaudojimo atvejus pilnai galima realizuoti JAVA programavimo kalba ir pasirinkta instrumentine priemone darbui su ontologijomis JENA bei jos įrankis Schemagen. Taip pat panaudojamas ROME pranešimų sindikavimo-agregavimo įrankis, validatorius FeedValidator ir aprašyta kuriame sistemos lygyje jie bus panaudojami siekiant metodo realizacijos.

Grafinės sąsajos komponentai sistemoje nenumatomi. Tačiau apibrėžta 3-jų sluoksnių architektūra ir jų vieta sistemoje.

Atliktas metodą realizuojančios sistemos projektavimas leidžia pereiti prie tolimesnės įgyvendinimo stadijos, kurioje detalizuosime metodą siauresnei realizacijai su Atom 1.0 formato pranešimais ir realizuosime sistemą eksperimentui atlikti.

4. Metodo įgyvendinimas JAVA programavimo kalba

Metodas įgyvendinama JAVA programavimo kalba (jdk_1.5_07) panaudojant Eclipse SDK aplinką. Tai lėmė keli faktoriai:

- JAVA ir Eclipse SDK yra atviro kodo nemokamos programos
- Pagrindiniai įrankiai realizuojantys semantinį duomenų agregavimą ir sistemos numatomos naudoti darbe suprogramuotos naudojant šias technologijas.

Pagrindiniai įrankiai: JENA instrumentinė priemonė, ROME, FeedValidator, Postgre SQL serveris ir jų apjungimas pagrindinėje aplikacijoje detalizuojama tolimesniuose skyriuose.

Įgyvendinant metodą eksperimento implementacija atlikta panaudojant Atom 1.0 formato savybes. Implementacija eksperimentams kitiems sindikavimo formatams bus analogiška.

4.1. Detalizuotas realizuojamo metodo veikimas įgyvendinime

Prieš pradedant programavimą detalizuojama metodo diagrama pateikta 3.1 skyriuje.

Schemoje pavaizduotoje 25 paveiksle detalizuotas metodas su įrankiais, kuriais jis realizuojamas, bei įvairiais formatais aprašytais duomenų šaltiniais.

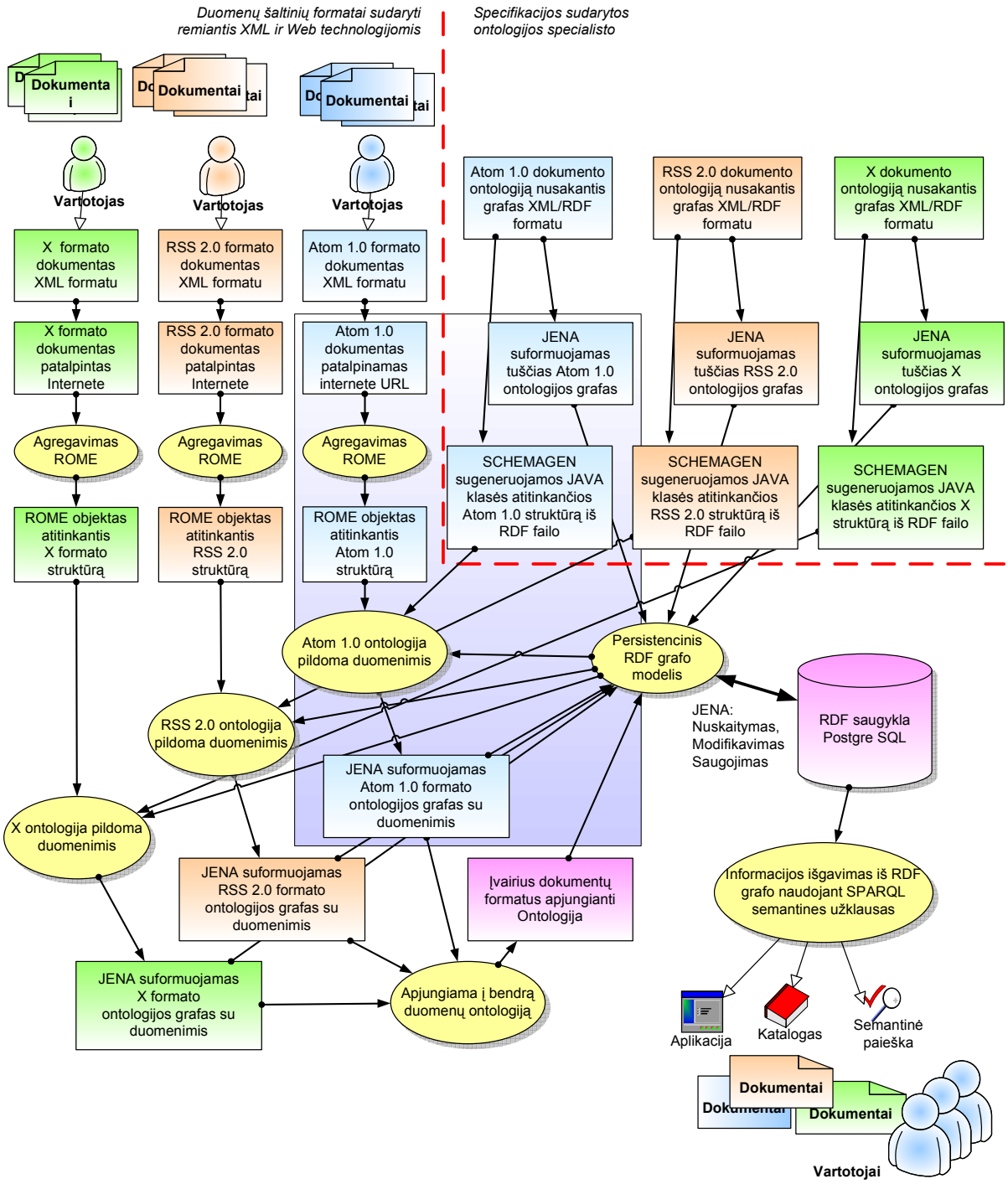
Sistemoje naudojamas ontologijas, kurios aprašo kiekvieno formato elementus, savybes ir ryšius tarp jų sudaro tos srities specialistai. Jie naudodami bendrus RDF ir OWL kalbos elementus, sudaro XML/RDF failus. Sistemoje šie ontologijas aprašantys failai panaudojami prieš vykdant sistemos darbą ir tada kai modelis yra sukurtas naujas arba reikalingas atnaujinimas. Vienas iš failo panaudojimų susijęs su klasių aprašų generavimu panaudojant Schemagen įrankį, per parametrus jam pateikiant XML/RDF failą, o išeigoje gaunami klasės aprašai. Kitas panaudojimas –ontologijos modelio, paremtu JENA instrumentinės priemonės mechanizmu objektų saugojimui į RDB, formavimas tokiu būdu sukuriant ontologijos kopiją duomenų bazėje, kuri tik aprašo schemą, bet neturi savyje jokiu egzempliorių. Panaudojant sugeneruotus klasių aprašus ontologija pildoma egzemplioriais.

Metodo aktualumas atsiskleidžia kai skirtingi vartotojai dokumentus publikuoja įvairiais formatais. Schemos viršuje pavaizduoti vartotojai turintys savą aibę dokumentų. Kiekvienas iš jų dokumentus surenka ir publikuoja pasauliniame tinkle skirtingais formatais: Atom 1.0, RSS 2.0 ar kitais – schemoje pažymėtais X. Publikavimas susideda iš dviejų etapų: 1) sugeneruojamas XML pagrindu sudarytas failas; 2) failas patalpinamas arba atnaujinimas tam tikru URL adresu. Tokių URL sąrašas yra pirminiai duomenys sistemai identifikuojantys

duomenų pasauliniame tinkle vietą. Sistemoje numatoma duomenų šaltinius validuoti panaudojant įrankį FeedValidator. Schemoje priimta, kad šaltiniai suformuoti teisingai ir juos galima apdoroti. Tokiu atveju atliekamas duomenų agregavimas panaudojant ROME įrankį, kuris nuskaito duomenis esančius pasauliniame tinkle ir suformuoja duomenų objektus su visomis to formato savybėmis.

Toliau sistema atitinkamai pagal duomenų formatą surašinėja duomenis iš objektų į to formato pranešimų ontologiją. Panaudojami ROME objektai su informacija, jų sugeneruoti klasių aprašai pagal JENA instrumentinę priemonę ir tuščias ar dalinai užpildytas ontologijos modelis iš duomenų bazės, sudarytas pagal automatinio objektų saugojimo į RDB mechanizmą. Užpildžius ontologijos modelį šis automatiškai išsaugomas duomenų bazėje. Šis funkcionalumas yra suprogramuotas.

Turint užpildytus ontologijų modelius kiekvienam formatui galimas jų apjungimas į vieną modelį panaudojant JENA funkcijas. Šis modelis taip pat išsaugomas duomenų bazėje. Pakartotinis duomenų panaudojimas ir pridėtinė vertė atsiranda naudojant bendrą pranešimų ontologiją, kurioje esantys egzemplioriai gali būti pateikiami įvairiais pavidalais: aplikacija, katalogu ar semantinės paieškos rezultatų suvestine. Tokiu būdu suformuojami vieningi dokumentai, o ne skirtingų formatų kokie jie buvo prieš patenkant į sistemą.



25 pav. Detalizuota duomenų agregavimo metodo schema

4.1.1. Schemagen panaudojimas generuojant JAVA klases

Schemagen aprašymas pateiktas 2.4.5 skyriuje. Detali naudojimo instrukcija su parametrų lentelėmis pateikiama adresu: http://jena.sourceforge.net/how-to/schemagen.html

Šiame darbe programa aktyvuojama nurodant vykdomąjį failą Ant[31] įrankiui. Vykdomojo failo tekstas:

[31] http://ant.apache.org/

```
set CP=C:\Jena\lib\antlr-2.7.5.jar;C:\Jena\lib\arq.jar;C:\Jena\lib\concurrent.jar;C:\Jena\lib\iri.jar;
C:\Jena\lib\icu4j_3_4.jar;C:\Jena\lib\jena.jar;C:\Jena\lib\jenatest.jar;C:\Jena\lib\json.jar;C:\Jena\lib\junit.jar;
C:\Jena\lib\log4j-1.2.12.jar;C:\Jena\lib\commons-logging-api.jar;C:\Jena\lib\commons-logging.jar;
C:\Jena\lib\xercesImpl.jar;C:\Jena\lib\xml-apis.jar;C:\Jena\lib\wstx-asl-2.8.jar;C:\Jena\lib\stax-api-1.0.jar;

java
        -classpath %CP%
        jena.schemagen
        -i AtomOwl.rdf
        -a http://bblfish.net/work/atom-owl/2006-06-06/
        --ontology --owl
        -o atom.java
```

Komanda SET nustato kintamajam CP visus ClassPath reikalingus įtraukti įrankius darbui su šia priemone. Toliau vykdoma Schemagen programa su JAVA komanda ir kintamasis priskiriamas parametrui –classpath. JENA įrankiui Schemagen paduodami parametrai:

- -i AtomOwl.rdf – ontologijos schemą aprašantis failas;
- -a http://bblfish.net/work/atom-owl/2006-06-06/ - vardo sritis;
- --ontology – nurodo, kad schema aprašo ontologiją;
- --owl – kad RDF schemoje naudojami OWL sintaksės intarpai;
- -o atom.java – failas į kurį išvedama sugeneruota Atom klasė.

Klasė Atom.java sugeneruota darbui su Atom 1.0 formato pranešimais formuojant ontologiją pateikiama priede C.

4.1.2. Tuščio ontologijos modelio sukūrimas pagal ontologiją aprašantį failą

Programos kodas realizuojantis tuščio ontologijos modelio sukūrimą. Veiksmų aprašymai komentaruose.

```
public void createEmptyAtomOntologyDatabase() {

        // ATOM struktūrą aprašantis RDF failas
        String fileName = "file:AtomOwl.rdf";

        // Prisijungimas prie duomenų bazės, išvalymas
        // Sukuriamas objektas skirtas ontologijos
        // modeliui formuoti duomenų bazėje
        ModelMaker maker = getRDBMaker(true);

        // Sukuriamas bazinis modelio objektas su specifikacijomis
        Model base = maker.createModel(s_base_model_name, true);
        OntModelSpec spec = new OntModelSpec(OntModelSpec.OWL_MEM);
        spec.setImportModelMaker(maker);

        // Sukuriamas ontologijos modelis
        OntModel model = ModelFactory.createOntologyModel(spec, base);

        // suformuojamas įvedimo srautas iš failo
        s_source = OntDocumentManager.
```

```
                                        getInstance().
                                        doAltURLMapping(fileName);

            // formuojamas ontologijos modelis, automatiškai išsaugomas DB
            model.read(s_source);
}
```

Atom 1.0 formato struktūrą aprašantis RDF failas buvo sukurtas šio formato kūrėjų. Analogiškai turint kitų sindikavimo formatų aprašą ontologijomis jiems bus sukuriami tušti ontologijų modeliai ir išsaugomi duomenų bazėje.

4.1.3. Agregavimas ROME

ROME aprašymas pateiktas skyrelyje 2.4.3.

Panaudojamas WireFeed objektas pranešimui parsiųsti, o vėliau konvertuojamas į Feed objektą ROME pakete pilnai atitinkantį Atom 1.0 specifikacijas.

Iš bendro ROME funkcionalumo pavaizduoto 15 paveiksle panaudojama 1 ir 2 dalys, kur:

- 1 – nuskaito duomenis iš Interneto
- 2 – sudeda duomenis į WireFeed modelį.

```
private WireFeed parseFeed(String fURL) throws Exception {

            URL feedUrl = new URL(fURL);
            WireFeedInput winput = new WireFeedInput();
            WireFeed wfeed = winput.build(new XmlReader(feedUrl));
            return wfeed;
}
```

4.1.4. Atom 1.0 ontologijos pildymas duomenimis

Viešas metodas, kuriam nurodoma iš kur imti duomenų šaltinį (URL). Funkcionalumas paaiškintas komentaruose.

```
public void addFeedToDatabaseFromURL(String feedURL) throws Exception
{
            /**
             * Parsiųsti Atom pranešimą iš nurodyto URL
             * paversti jį į objektą (com.sun.syndication.feed.atom)
             * atitinkanti Atom 1.0 formato specifikacijas
             */
            WireFeed wireFeed = parseFeed(feedURL);
            Feed feed = (Feed) wireFeed;

            /**
             * Nuskaityti persistencinį modelį iš duomenų bazės
             */
            Model base = loadAtomPersistentModel();
            OntModelSpec spec = new OntModelSpec(OntModelSpec.OWL_MEM);
            OntModel model = ModelFactory.createOntologyModel(spec, base);

            /**
             * Pridėti duomenis iš objekto į ontologijos modelį sukurtą naudojant
```

```
            * persistencijos mechanizmą
            */
           AtomFeedInstanceMaker feedMaker = new AtomFeedInstanceMaker();
           feedMaker.makeFeed(feed, model);

           /**
            * Atspausdinti modelį į failą testo rezultatams patikrinti
            */
           FileOutputStream outF = new FileOutputStream("debugas.rdf");
           PrintStream p = new PrintStream(outF);
           base.write(p);
           p.close();
}
```

Šis metodas naudoja privatų metodą ir specialią klasę duomenų sutapatinimui tarp ROME objekto ir JENA ontologijos modelio. Jų aprašymai pateikiami sekančiuose poskyriuose.

4.1.5. Ontologijos modelio nuskaitymas iš duomenų bazės

Metodas nuskaitantis modelį iš duomenų bazės. Tokiu būdu nuskaityto modelio pakeitimai automatiškai išsaugomi į duomenų bazę panaudojant persistencijos mechanizmą.

- M_DB_URL, M_DB_USER, M_DB_PASSWD, M_DB – konstantos nusakančios prisijungimą prie duomenų bazės.
- s_base_model_name – modelio pavadinimą sauganti konstanta

```
public Model loadAtomPersistentModel() {

           IDBConnection conn = new DBConnection(
                                       M_DB_URL, M_DB_USER, M_DB_PASSWD, M_DB);
           ModelMaker maker = ModelFactory.createModelRDBMaker(conn);
           Model base = maker.openModel(s_base_model_name);
           return base;
}
```

4.1.6. Duomenų perkėlimo iš ROME į Ontologijos modelį klasė

Klasė pavadinta AtomFeedInstanceMaker atlieka veiksmus perkeliant duomenis iš ROME objekto į ontologijos modelį.

Kuriant egzempliorių (individą) naudojamas metodas nusakant jo resursą identifikuojantį ontologijoje ir jo klasės tipą. Jeigu egzempliorius turi savybę, tai ta savybė jam priskiriama su tam tikra reikšme.

Pagrindinis metodas makeFeed(feed, model) pereina per visą objekto struktūrą ir formuoja ontologijos modelį.

Parametrai:

- feed – pranešimo duomenis saugantis objektas (deklaracija ROME pakete com.sun.syndication.feed.atom);
- m – ontologijos modelis (deklaracija JENA instrumentinės priemonės pakete com.hp.hpl.jena.ontology).

Pateikiamas metodo kodas rodo, kad tai yra viešas metodas per kurį atsiskleidžia klasės AtomFeedInstanceMaker funkcionalumas.

```
public void makeFeed(Feed feed, OntModel m) {

	/*
	 * Sukuriamas pranešimo privatus metodo egzempliorius
	 * su sugeneruotu identifikatoriumi
	 */
	Individual _feed = m.createIndividual(
				generateUrn(feed.getId(), feed.getUpdated()),
				Atom.Feed);
	/*
	 * Pranešimui prisikiriami visos galimos jo savybės
	 * pereinant per vidinę jo struktūrą
	 */
	makeFeedAlternateLinks(m, feed, _feed);
	makeFeedAuthors(m, feed, _feed);
	makeFeedCategories(m, feed, _feed);
	makeFeedContributors(m, feed, _feed);
	makeFeedEntries(m, feed, _feed);
	makeFeedGenerator(m, feed, _feed);
	makeFeedIcon(feed, _feed);
	makeFeedId(feed, _feed);
	makeFeedLogo(feed, _feed);
	makeFeedRights(feed, _feed);
	makeFeedSubtitle(feed, _feed);
	makeFeedTitle(feed, _feed);
	makeFeedUpdated(feed, _feed);
	makeFeedXmlBase(feed, _feed);
}
```

Siekiant paaiškinti patį principą pateikiamas pagal analogiškus principus metodui makeFeed metodo makeFeed pagalbinis metodas makeEntryAuthors. Šis metodas formuoja įrašo (Entry) autorius (Authors), kur pagal argumentus:

- m – ontologijos modelis,
- entry – pranešimo įrašas,
- _entry – egzempliorius jau sukurtas modelyje.

```
private void makeEntryAuthors(OntModel m, Entry entry, Individual _entry) {
	Individual _author;
	List authors = entry.getAuthors();
	if (authors != null) {
		Iterator i = authors.iterator();
		while (i.hasNext()) {
			Person author = (Person) i.next();
			_author = makePerson(m, author);
			if (_author != null)
				_entry.addProperty(Atom.author, _author);
		}
	}
}
```

Su šiuo metodu susijęs kitas pagalbinis metodas sukuria autorių reprezentuojantį egzemplioriaus klasę (Person). Metodo pavadinimas makePerson su argumentais:

- m – ontologijos modelis;
- person – iš formato ontologijos aprašo sugeneruotos schemos abstrakti klasė atvaizduojanti asmenį (Priedas C).

```
private Individual makePerson(OntModel m, Person person) {
        String name = person.getName();
        String uri = person.getUri();
        String email = person.getEmail();

        if (email == null)
                return null;

        Individual _person = m.createIndividual(
                                        "http://zmones.lt/" + email,
                                        Atom.Person);
        _person.addProperty(Atom.email, email);
        if (name != null)
                _person.addProperty(Atom.name, name);
        if (uri != null)
                _person.addProperty(Atom.uri, uri);

        return _person;
}
```

Atom.email, Atom.name ir kiti – sugeneruotos Schemagen iš Atom formato ontologijos aprašo klasės atributai.

Visų kitų klasių ar konceptų kūrimo, bei reikšmių jų savybėms priskyrimo principai bus analogiški pastarąjam.

4.1.7. Ontologijų apjungimas į bendrą modelį

Nuskaitomi visų formatų modeliai su metodais analogiškais pateiktam 4.1.5 skyriuje ir apjungiami į vieną panaudojant JENA instrumentinės priemonės funkciją union(Model m).

```
public Model allModelUnion()
{
        /**
         * Nuskaitomi visų formatų modeliai
         */
        Model mRSS10 =          loadRSS10Model();
        Model mRSS20 =          loadRSS20Model();
        Model mAtom10 =         loadAtom10Model();
        Model mX =              loadXModel();

        /**
         * Sukuriamas tuščias ontologijos modelis
         */
        Model m = ModelFactory.createDefaultModel();

        /**
         * Nuskaityti modeliai apjungiami panaudojant apjungimo operaciją.
         * m = m U mRSS10 U mRSS20 U mAtom10 U mx
         */
```

```
        m = m.union(mRSS10).union(mRSS20).union(mAtom10).union(mX);

        return m;
}
```

4.1.8. Informacijos išgavimas iš RDF grafo panaudojant SPARQL

Duomenų išgavimui iš onotlogijomis aprašytų saugyklų naudojam SPARQL užklausų kalba. Jos trumpas aprašymas pateikiamas 2.1.3.3 skyriuje.

Žemiau pateiktas metodo fragmentas, kaip metodą realizuojančioje programoje panaudojant JENA instrumentinės priemonės funkcionalumą suformuota semantinė užklausa pateikiama per parametrus yra įvykdoma ieškant informacijos duomenų bazėje.

```
public void queryDatabase( String queryString ) throws IOException  {

        /**
         * Modelis nuskaitomas iš duomenų bazės
         */
        Model model = getModelFromDatabase();

        /**
         * Sukuriamas užklausos objektas pagal paduotą SPARQL užklausą.
         */
        Query query = QueryFactory.create(queryString);

        /**
         * Vykdoma užklausa iš ontologijos modelio duomenų
         */
        QueryExecution qe = QueryExecutionFactory.create(query, model);
        ResultSet results = qe.execSelect();

        /**
         * Atspausdinami užklausos rezultatai.
         * Gali būti panaudojami kitokiems pateikimo būdams
         */
        ResultSetFormatter.out(System.out, results, query);

        // Atlaisvinami užklausos resursai
        qe.close();
}
```

4.2. Metodo realizacijos testavimas

4.2.1. Pasiruošimas eksperimentui

Ruošiantis eksperimentui buvo atliekami šie veiksmai:

- sukurti testiniai duomenys aprašomi Atom 1.0 formato failais.
- Failai patalpinti Pasauliniame tinkle.
- Sukonfigūruotas Postgre SQL serveris duomenų saugojimui.
- Sukurta JAVA aplikacija atliekanti pagrindinius metodo žingsnius.
- Sukonfigūruotas grafinis interpretatorius (RDF Gravity[32]) gautų rezultatų teisingumui tikrinti

4.2.2. Testavimo veiksmų seka

Testavimui atlikti numatyta tokia pagrindinių veiksmų seka:

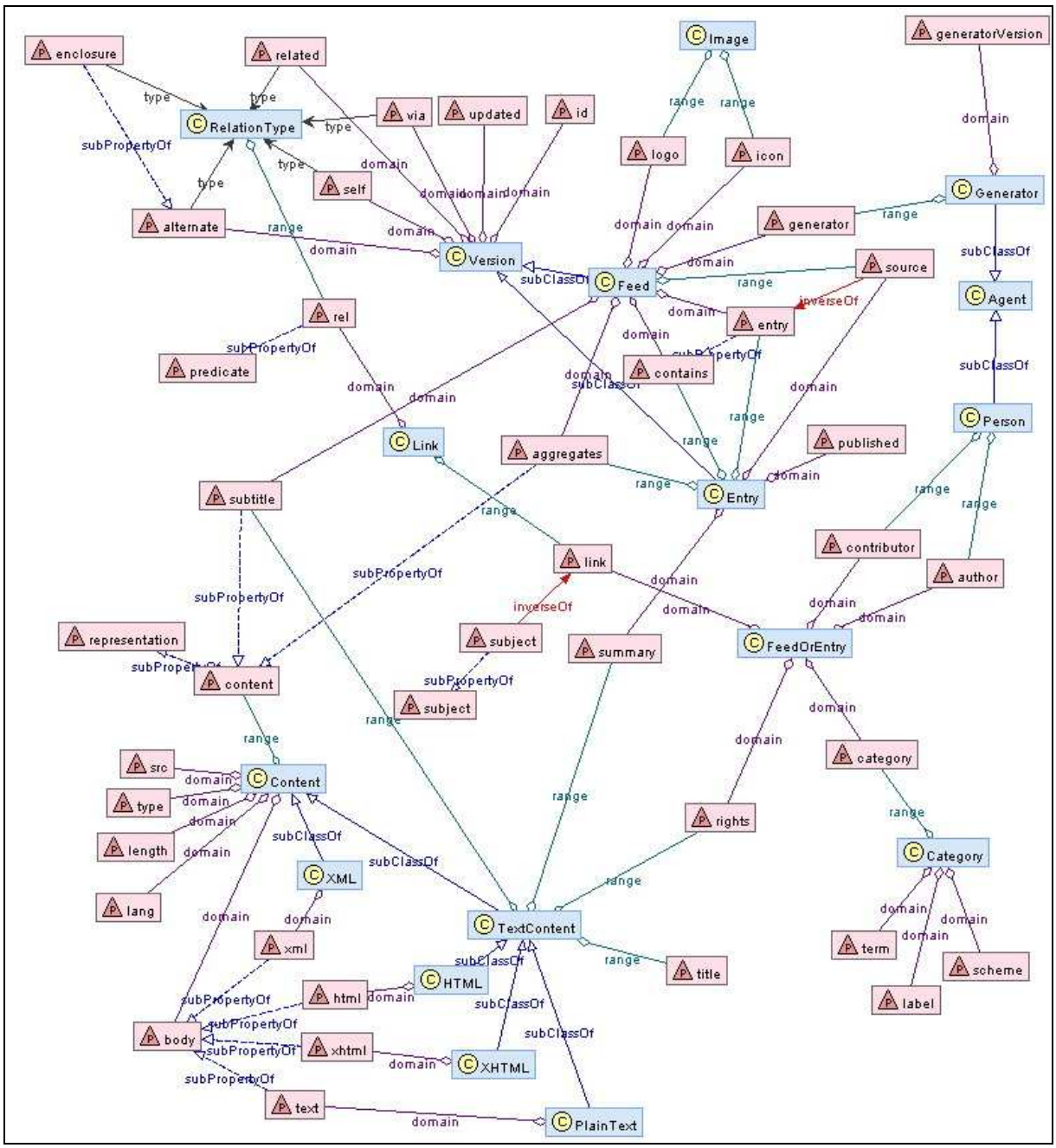
1. Sukurti tuščią ontologijos modelį, jei nebuvo sukurtas (šis žingsnis atliekamas tik pirmą kartą).
2. Ištrinti ontologijos modelį ir sukurti tuščią, jei norima išvalyti duomenis (galima praleisti, jei nereikia ištrinti buvusių duomenų)
3. Agreguoti duomenis iš įvestų URL identifikuojančių duomenų šaltinių.
4. Surašyti duomenis į ontologijos modelį ją papildant. Duomenys susirašys į duomenų bazę automatiškai per persistencijos mechanizmą.
5. Atspausdinti modifikuotą modelį į XML/RDF formato failą.
6. Peržiūrėti suformuotą XML/RDF failą su rezultatais panaudojant RDF Gravity grafinį interpretatorių ar visi duomenys iš duomenų failo buvo parsiųsti ir teisingai surašyti į ontologiją.

4.3. Eksperimentas

Eksperimento metu naudojamas sukurtas Atom 1.0 duomenų failas pateikiamas Priede A. Toks failas patalpinamas Internete ir toliau eksperimentas atliekamas testavime (žiūr. 4.2 skyrių) sudaryta tvarka.

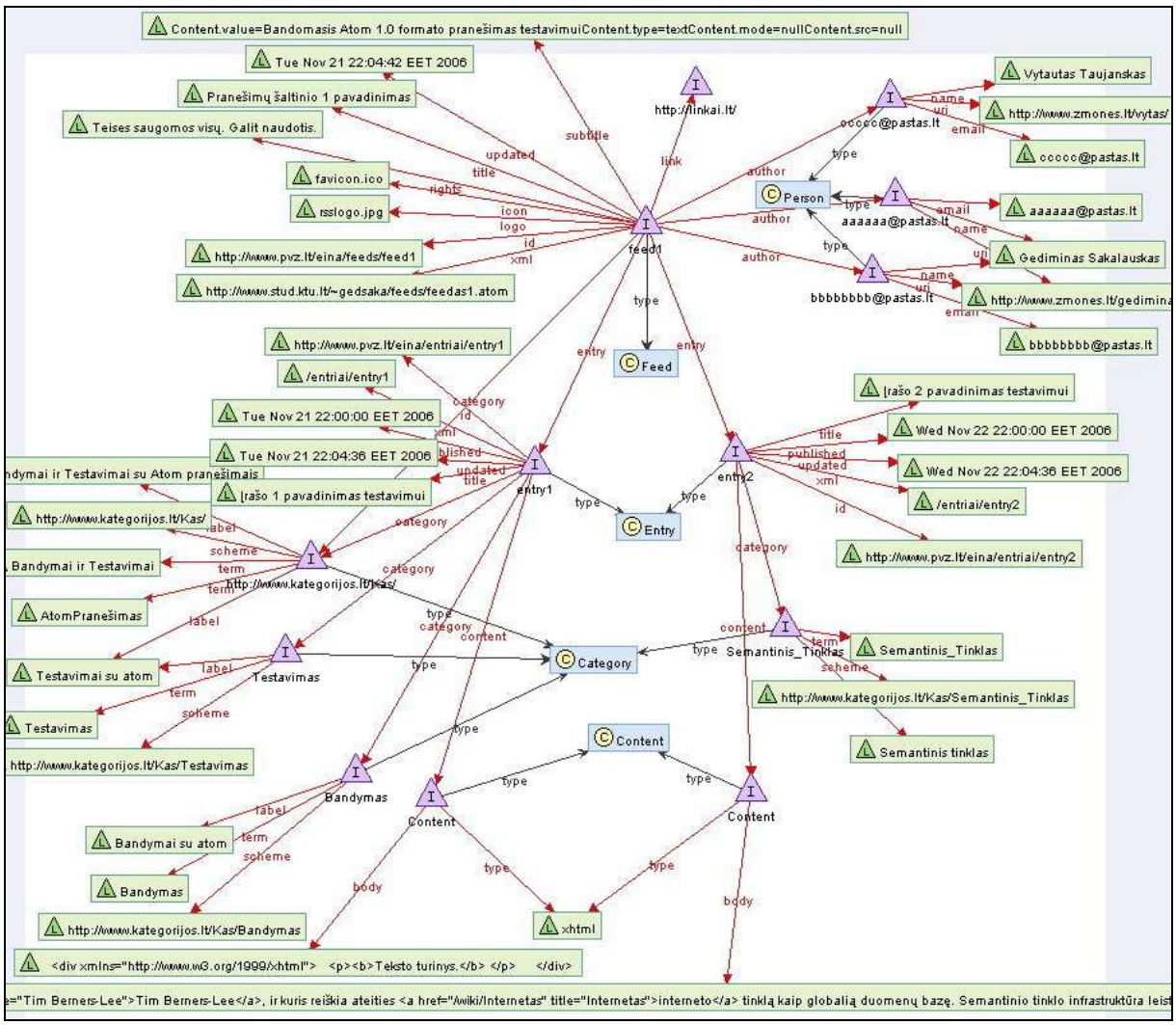
Prieš agreguojant duomenis sukuriamas tuščias ontologijos modelis duomenų bazėje, kaip terpė saugoti duomenims. Tuščias ontologijos modelis atitinkantis Atom 1.0 formatą atvaizduotas grafiniame interpretatoriuje pateikiamas 26 paveiksle, kuriame paliktos tik klasės su savybėmis. Notacija ir ontologijų aprašas toks pat kaip 2.4.1.2 skyriuje.

[32] http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html



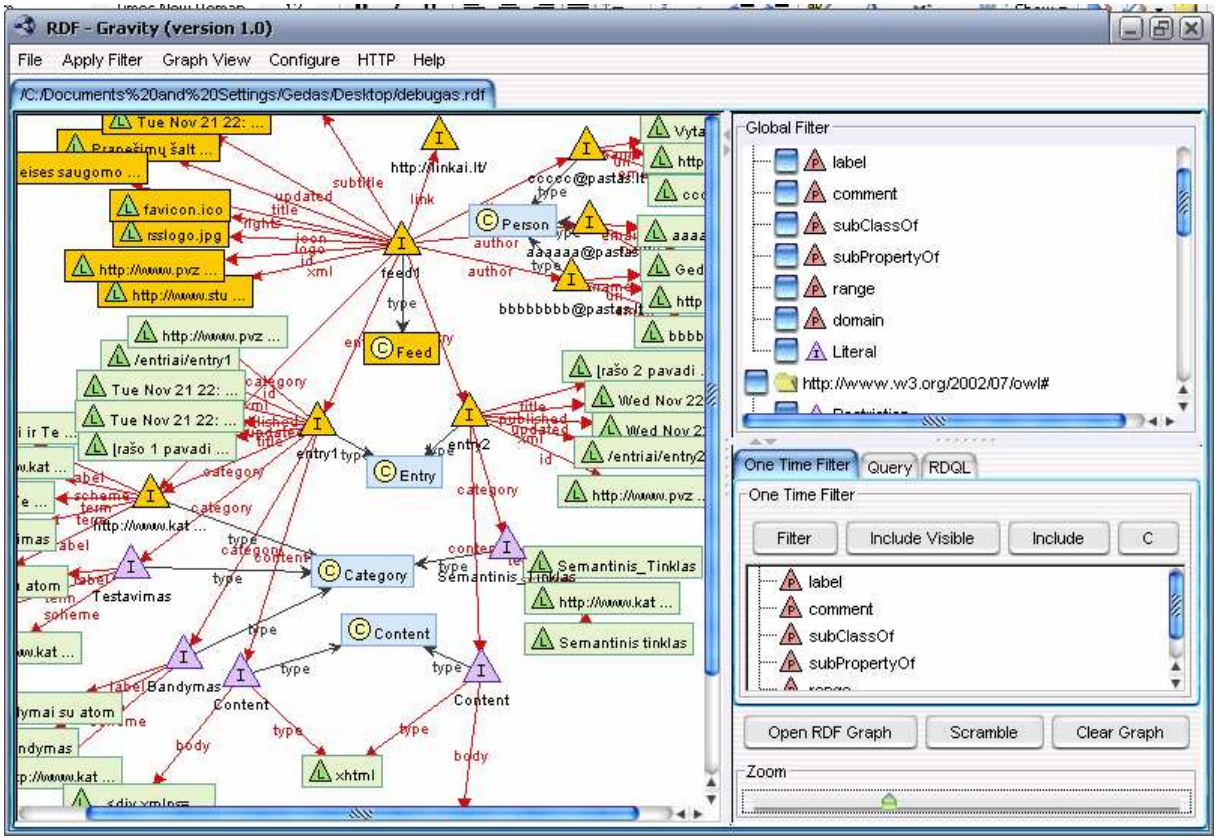
26 pav. Ontologija atvaizduojanti Atom 1.0 formato struktūrą – be egzempliorių

Agreguojamas Atom 1.0 formato pranešimas ir jame esantys duomenys perkeliami į ontologijos modelį. Šis modelis automatiškai išsaugomas duomenų bazėje ir išvedamas į rezultatų failą teisingumo tikrinimui. Užpildytas ontologijos modelis egzemplioriais iš testinių duomenų ir išvestas į rezultatų failą pateikiamas 27 paveiksle – matome, kad šalia klasių atsirado egzemplioriai su galimomis savybėmis ir joms priskirtomis reikšmėmis:



27 pav. Ontologija atvaizduojanti Atom 1.0 formato struktūrą – su egzemplioriais

Testavimo aplikacijos rezultatai išvesti į rezultatų failą panaudojant JENA instrumentinės priemonės išvedimo standartinius metodus. Rezultatų failas atvaizduotas grafiniame interpretatoriuje pateikiamas 28 paveiksle.



28 pav. Grafinis interpretatorius RDF Gravity atvaizduoja testinės aplikacijos rezultatus

4.4. Siūlomo metodo realizacijos ir eksperimento išvados

Atlikus siūlomo duomenų agregavimo metodo realizaciją JAVA programavimo kalba gauti rezultatai kokių ir buvo tikėtasi. Analizuojant Atom 1.0 formato ontologijos aprašymą ir grafą be duomenų pateiktame 26 paveiksle matome, kad ontologijos grafas sudarytas pagal šio formato specifikacijas. Vadinasi specifikacijų perkėlimas į duomenų bazėje saugomas ontologijas panaudojant JENA instrumentinę priemonė įvykdytas sėkmingai.

Lyginant Atom 1.0 pranešimo testinių duomenų failą ir ontologijos grafą su užpildytais duomenimis matome, kad visi duomenys buvo sėkmingai perkelti į ontologijos grafą. Vadinasi agregavimo ir duomenų perkėlimo žingsniai buvo įvykdyti teisingai.

Eksperimentas buvo atliktas tik su Atom 1.0 formato failais ir šio formato ontologijos modeliu. Eksperimentą pavyko atlikti sėkmingai ir gauti rezultatai buvo tokie, kokių tikėtasi. Kadangi ROME gali agreguoti skirtingus formatus, galima rasti kitų formatų ontologijų modelius ir juos panaudoti klasių generavimui, daroma prielaida, kad pateiktas metodas analogiškai veiks kitiems formatams: RSS 1.0, RSS 2.0 ir kt.

5. Išvados

Darbe išanalizuotos semantinį pasaulinį tinklą realizuojančios technologijos ir pasiūlytas metodas jas panaudojantis duomenų agregavimui išsaugant ontologijose.

Išnagrinėti ontologijų valdymo uždaviniai ir šia tema perskaitytas pranešimas „*Wordnet uždaviniai ontologijos išgavime iš laisvojo teksto*“, 11-oje tarpuniversitetinėje doktorantų ir magistrantų konferencijoje „Informacinės technologijos 2006”. Straipsnis atspausdintas konferencijos pranešimų medžiagoje: I dalis, 166 psl. (literatūros sąraše [27], priedas F). Ontologijų aprašai šiame darbe panaudoti pateikiant metodą realizuojančios programos tarpinius ir galutinius rezultatus.

Detalizuoti su semantinio pasaulinio tinklo vizija susiję uždaviniai, ontologijų aprašymo kalbos (RDF, OWL), semantinių užklausų kalba (SPARQL), įrankiai ir instrumentinės priemonės naudojami darbui su ontologijomis. Išanalizuotas šiuo metu pasauliniame tinkle naudojamas duomenų agregavimo-sindikavimo metodas. Duomenų sindikavimo formatai RSS, Atom, bei jų specifikacijas aprašanti ontologija. Ištirtas įrankis atliekantis agregavimo, sindikavimo uždavinius. Pasiūlyta kategorizavimo uždavinį spręsti panaudojant atvirojo katalogo sistemą (DMOZ.org).

Pagal siūlomą metodą suprojektuota sistema: pateikti panaudojimo atvejai, sekų, būsenų diagramos, architektūros modelis, detalizuoti komponentai ir ryšiai tarp jų.

Sistemos projektas realizuotas JAVA programavimo kalba Eclipse aplinkoje, panaudojant duomenų publikavimo ir duomenų agregavimo įrankį ROME, egzistuojančią instrumentinę priemonė JENA darbui su ontologijomis bei kitus įrankius, ir panaudotas eksperimentui atlikti.

Atlikus eksperimentą su sudarytais testiniais duomenimis gauti rezultatai, kokių ir buvo tikėtasi, o tai reiškia, kad metode siūlomą funkcionalumą galima pasiekti esamais įrankiais.

Siūlomo metodo rezultatai gali būti panaudoti kaip posistemė semantinio portalo pasiūlyto [1] straipsnyje sukuriant naują semantinio pasaulinio tinklo paslaugą. Vadinasi darbe pasiūlytas pasauliniame tinkle transliuojamų duomenų agregavimo metodas panaudojant ontologijas gali būti pilnai įgyvendinamas egzistuojančiomis semantinio pasaulinio tinklo technologjomis.

Literatūra

[1] Vytautas Taujanskas, Rimantas Butleris, *"Categories extraction for reuse in semantic applications and profile based recommendation service"*, Information Systems Development Conference Proceedings, Springer Verlag 2006

[2] Tim Berners-Lee, James Hendler, Ora Lassila, *"The Semantic Web"*, Scientific American, May 2001

[3] T. R. Gruber, *"A translation approach to portable ontologies. Knowledge Acquisition"*, 5(2):199-220, 1993

[4] J.A. Hendler, *"Frequently Asked Questions on W3C's Web Ontology Language (OWL)"*, W3C, 2004; www.w3.org/2003/08/owlfaq.

[5] Aimilia Magkanaraki, Grigoris Karvounarakis, Ta Tuan Anh, Vassilis Christophides, Dimitris Plexousakis; *"Ontology Storage and Quering"*; Technical Report No 308;FORTH Institute of Computer Science, 2002.

[6] Saulius Maskeliūnas, *"Ontologijų naudojimas interneto technologijomis grindžiamoms paslaugoms intelektualizuoti"* Informcijos Mokslai, 2003 Nr. 26, ISBN 1392-0561

[7] Lina Tutkutė, Vytautas Taujanskas, *„Internetinio socialinio tinklo formavimas semantiniame žiniatinklyje įverčių skaičiavimų pagrindu."* Informacinės technologijos: XI tarpuniversitetinė magistrantų ir doktorantų konferencija: konferencijos pranešimų medžiaga. Kaunas, 2006, ISBN 9986-1-877-1, p. 178-183.

[8] Ben Hammersley, *"Content Syndication with RSS"*, O'Reilly, ISBN: 0-596-00383-8, March 2003

[9] Dave Johnson, *"RSS and Atom in Action"*, United States of America, Manning Publications Co., 2006, ISBN 1932394494

[10] John Davies, Dieter Fensel, Frank van Harmelen, "*Towards the semantic web : ontology-driven knowledge management*", Willey, England, ISBN 0-470-84867-7, 2002, 328 p

[11] Soumen Chakrabarti, "*Mining the Web*", Morgan Kaufman Publishers, USA, ISBN: 1-55860-754-4, 2003

[12] John Davies, Rudi Studer, Paul Warren*; "Semantic Web Technologies: Trends and Research in Ontology based Systems"*, Wiley, Jul 2006, 327 p.

[13] Giorgos Stamou, Stefanos Kollias (Eds.); *"Multimedia Content and the Semantic Web: Standards, Methods and Tools"*. Wiley, May 2005, 414 p.

[14] Ubbo Visser; *"Intelligent Information Integration for the Semantic Web"*; Springer (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence), 2005, 150 p.

[15] Tim Berners-Lee, Eric Miller, "*The Semantic Web lifts off*", W3C, ERCIM News No. 51, 2002.

[16] Tim Berners-Lee, Dan Connolly, Ralph R. Swick, "*Web Architecture: Describing and Exchanging Data*", W3C Note 7, 1999

[17] Nigel Shadbolt, Tim Berners-Lee and Wendy Hall, "*The Semantic Web Revisited*", IEEE Intelligent Systems 21(3) pp. 96-101, 2006

[18] Michael C. Daconta, Leo J. Obrst and Kevin T. Smith, "*The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*", John Wiley & Sons, 2003, ISBN:0471432571

[19] Shelley Powers, "*Practical RDF*", O'Reilly, ISBN: 0-596-00263-7, 2003

[20] Ben Hammersley, "*Developing Feeds with RSS and Atom*", O'Reilly, ISBN: 0-596-00881-3, 2005

[21] David William Williamson; *"A Lightweight Data Integration Architecture";* A thesis submitted for the degree of Master of Science at the University of Otago, Dunedin,New Zealand. 12 June 2006, 101 p. http://eprints.otago.ac.nz/484/01/Williamson_thesis.pdf

[22] Adrian Listyo; *"Integration of Heterogeneous Data Stream Types in JOpera"*; Master thesis, Information and Communication Systems Research Group, Institute for Pervasive Computing, Department of Computer Science, ETH Zurich, February 24, 2006, 51 p. http://www.iks.inf.ethz.ch/iks/publications/files/Adrian_Listyo_MT.pdf

[23] Reuven M. Lerner; *"At the Forge: Aggregating Syndication Feeds"*; Linux Journal, Vol. 2004 , Issue 128 (December 2004) http://www.linuxjournal.com/article/7775

[24] David Williamson, Nigel Stanger; *"A Lightweight Data Integration Architecture using Atom";* The Information Science Discussion Paper Series, No. 2005/04, March 2005, 7 p.

http://www.business.otago.ac.nz/infosci/pubs/papers/papers/dp2005-04.pdf

[25] Shelley Powers; *"What Are Syndication Feeds"*, ISBN: 0-596-52697-0, 2005

[26] http://www.w3.org/TR/owl-ref/, *"OWL Web Ontology Language Reference"*; W3C Recommendation 10 February 2004.

[27] Gediminas Sakalauskas, Vytautas Taujanskas; "*Wordnet uždaviniai ontologijos išgavime iš laisvojo teksto*", Tarpuniversitetinė doktorantų magistrantų konferencija "Informacinės Technologijos 2006", Kaunas, 2006.

PRIEDAI:

Priedas A: Atom 1.0 formato pranešimo pavyzdys

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:thr="http://purl.org/syndication/thread/1.0"
      xml:base="http://www.stud.ktu.lt/~gedsaka/feeds"
      xml:lang="en-us" >
 <title>Pranešimų šaltinio 1 pavadinimas</title>
 <id>http://www.pvz.lt/eina/feeds/feed1</id>
 <link rel="self" href="" xml:base="http://www.stud.ktu.lt/~gedsaka/feeds/feedas1.atom" />
 <link rel="http://www.tbray.org/ongoing/ongoing.atom" href="./" />
 <logo>rsslogo.jpg</logo>
 <icon>favicon.ico</icon>
 <updated>2006-11-21T12:04:42-08:00</updated>
 <category term="Bandymas" scheme="http://www.kategorijos.lt/Kas/" label="Bandymai su atom"></category>
 <category term="Testavimas" scheme="http://www.kategorijos.lt/Kas/" label="Testavimai su atom"></category>
 <author>
  <name>Gediminas Sakalauskas</name>
  <email>aaaaaa@pastas.lt</email>
  <uri>http://www.zmones.lt/gediminas/</uri>
 </author>
 <author>
  <name>Vytautas Taujanskas</name>
  <email>ccccc@pastas.lt</email>
  <uri>http://www.zmones.lt/vytas/</uri>
 </author>
  <author>
  <name>Gediminas Sakalauskas</name>
  <email>bbbbbbbb@pastas.lt</email>
  <uri>http://www.zmones.lt/gediminas/</uri>
 </author>
 <subtitle>Bandomasis Atom 1.0 formato pranešimas testavimui</subtitle>
 <rights>Teises saugomos visų. Galit naudotis.</rights>
 <generator uri="http://www.generatoriai.lt/">Generatoriaus versija</generator>

<entry xml:base="http://www.pvz.lt/eina/entriai/entry1">
 <title>Įrašo 1 pavadinimas testavimui</title>
 <link rel="http://www.stud.ktu.lt/~gedsaka/feeds/" href="Entriai">
  <subject>Subjektas įrašo 1</subject>
  <title>Nuorodos pavadinimas įrašo 1</title>
 </link>
 <id>http://www.pvz.lt/eina/entriai/entry1</id>
 <published>2006-11-21T12:00:00-08:00</published>
 <updated>2006-11-21T12:04:36-08:00</updated>
  <category term="Bandymas" scheme="http://www.kategorijos.lt/Kas/" label="Bandymai su atom"></category>
  <category term="Testavimas" scheme="http://www.kategorijos.lt/Kas/" label="Testavimai su atom"></category>
 <content type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
   <p><b>Teksto turinys.</b> </p>
  </div>
 </content>
</entry>

<entry xml:base="http://www.pvz.lt/eina/entriai/entry2">
 <title>Įrašo 2 pavadinimas testavimui</title>
 <link rel="http://www.stud.ktu.lt/~gedsaka/feeds/" href="Entriai">
  <subject>Subjektas įrašo 2</subject>
  <title>Nuorodos pavadinimas įrašo 1</title>
 </link>
 <id>http://www.pvz.lt/eina/entriai/entry2</id>
 <published>2006-11-22T12:00:00-08:00</published>
 <updated>2006-11-22T12:04:36-08:00</updated>
```

```
  <category term="SemantinisTinklas" scheme="http://www.kategorijos.lt/Kas/" label="Semantinis
tinklas"></category>
 <content type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
   <p><b>Semantinis tinklas</b> yra terminas, kurį sugalvojo <a href="/w/index.php?title=Tim_Berners-
Lee&amp;amp;action=edit" class="new" title="Tim Berners-Lee">Tim Berners-Lee</a>, ir kuris reiškia ateities <a
href="/wiki/Internetas" title="Internetas">interneto</a> tinklą kaip globalią duomenų bazę. Semantinio tinklo
infrastruktūra leistų tiek žmonėms, tiek mašinoms daryti sprendimus kaip kategorizuoti informaciją ir ją panaudoti. Į
architektūrinius elementus įeina semantika (elementų pavadinimai), struktūra (elementų hierarchija) ir sintaksė
(bendravimas)</p>
  </div>
 </content>
</entry>

</feed>
```

Priedas B: Atom 1.0 struktūra Relax NG notacijoje

```
# -*- rnc -*-
# RELAX NG Compact Syntax Grammar for the
# Atom Format Specification Version 11

namespace atom = "http://www.w3.org/2005/Atom"
namespace xhtml = "http://www.w3.org/1999/xhtml"
namespace s = "http://www.ascc.net/xml/schematron"
namespace local = ""

start = atomFeed | atomEntry

# Common attributes

atomCommonAttributes =
   attribute xml:base { atomUri }?,
   attribute xml:lang { atomLanguageTag }?,
   undefinedAttribute*

# Text Constructs

atomPlainTextConstruct =
   atomCommonAttributes,
   attribute type { "text" | "html" }?,
   text

atomXHTMLTextConstruct =
   atomCommonAttributes,
   attribute type { "xhtml" },
   xhtmlDiv


atomTextConstruct = atomPlainTextConstruct | atomXHTMLTextConstruct

# Person Construct

atomPersonConstruct =
   atomCommonAttributes,
   (element atom:name { text }
    & element atom:uri { atomUri }?
    & element atom:email { atomEmailAddress }?
    & extensionElement*)

# Date Construct


atomDateConstruct=
   atomCommonAttributes,
   xsd:dateTime

# atom:feed

atomFeed =
   [
      s:rule [
         context = "atom:feed"
         s:assert [
            test = "atom:author or not(atom:entry[not(atom:author)])"
            "An atom:feed must have an atom:author unless all "
            ~ "of its atom:entry children have an atom:author."
         ]
      ]
   ]
   element atom:feed {
      atomCommonAttributes,
      (atomAuthor*
       & atomCategory*
       & atomContributor*
       & atomGenerator?
       & atomIcon?
       & atomId

       & atomLink*
       & atomLogo?
       & atomRights?
       & atomSubtitle?
```

```
       & atomTitle
       & atomUpdated

       & extensionElement*),
      atomEntry*
   }

# atom:entry

atomEntry =
   [
      s:rule [
         context = "atom:entry"
         s:assert [
            test = "atom:link[@rel='alternate'] "
            ~ "or atom:link[not(@rel)] "
            ~ "or atom:content"
            "An atom:entry must have at least one atom:link element "
            ~ "with a rel attribute of 'alternate' "
            ~ "or an atom:content."
         ]
      ]
      s:rule [
         context = "atom:entry"
         s:assert [
            test = "atom:author or "
            ~ "../atom:author or atom:source/atom:author"
            "An atom:entry must have an atom:author "
            ~ "if its feed does not."
         ]
      ]
   ]
   element atom:entry {
      atomCommonAttributes,
      (atomAuthor*
       & atomCategory*
       & atomContent?
       & atomContributor*
       & atomId

       & atomLink*
       & atomPublished?
       & atomRights?
       & atomSource?
       & atomSummary?
       & atomTitle

       & atomUpdated
       & extensionElement*)
   }

# atom:content

atomInlineTextContent =
   element atom:content {
      atomCommonAttributes,
      attribute type { "text" | "html" }?,
      (text)*
   }


atomInlineXHTMLContent =
   element atom:content {
      atomCommonAttributes,
      attribute type { "xhtml" },
      xhtmlDiv
   }

atomInlineOtherContent =
   element atom:content {
      atomCommonAttributes,
      attribute type { atomMediaType }?,
      (text|anyElement)*
   }

atomOutOfLineContent =
   element atom:content {
      atomCommonAttributes,
      attribute type { atomMediaType }?,
      attribute src { atomUri },
```

```
      empty
   }


atomContent = atomInlineTextContent
 | atomInlineXHTMLContent
 | atomInlineOtherContent
 | atomOutOfLineContent

# atom:author

atomAuthor = element atom:author { atomPersonConstruct }

# atom:category

atomCategory =
   element atom:category {
      atomCommonAttributes,
      attribute term { text },
      attribute scheme { atomUri }?,
      attribute label { text }?,
      undefinedContent
   }

# atom:contributor

atomContributor = element atom:contributor { atomPersonConstruct }

# atom:generator

atomGenerator = element atom:generator {
   atomCommonAttributes,
   attribute uri { atomUri }?,
   attribute version { text }?,
   text
}

# atom:icon

atomIcon = element atom:icon {
   atomCommonAttributes,
   (atomUri)
}

# atom:id

atomId = element atom:id {
   atomCommonAttributes,
   (atomUri)
}

# atom:logo

atomLogo = element atom:logo {
   atomCommonAttributes,
   (atomUri)
}

# atom:link

atomLink =
   element atom:link {
      atomCommonAttributes,
      attribute href { atomUri },
      attribute rel { atomNCName | atomUri }?,
      attribute type { atomMediaType }?,
      attribute hreflang { atomLanguageTag }?,
      attribute title { text }?,
      attribute length { text }?,
      undefinedContent
   }

# atom:published

atomPublished = element atom:published { atomDateConstruct}

# atom:rights

atomRights = element atom:rights { atomTextConstruct }
```

```
# atom:source

atomSource =
   element atom:source {
      atomCommonAttributes,
      (atomAuthor*
       & atomCategory*
       & atomContributor*
       & atomGenerator?
       & atomIcon?
       & atomId?
       & atomLink*
       & atomLogo?
       & atomRights?
       & atomSubtitle?
       & atomTitle?
       & atomUpdated?
       & extensionElement*)
   }

# atom:subtitle

atomSubtitle = element atom:subtitle { atomTextConstruct }

# atom:summary

atomSummary = element atom:summary { atomTextConstruct }

# atom:title

atomTitle = element atom:title { atomTextConstruct }

# atom:updated

atomUpdated = element atom:updated { atomDateConstruct}

# Low-level simple types

atomNCName = xsd:string { minLength = "1" pattern = "[^:]*" }

# Whatever a media type is, it contains at least one slash
atomMediaType = xsd:string { pattern = ".+/.+" }

# As defined in RFC 3066
atomLanguageTag = xsd:string {
   pattern = "[A-Za-z]{1,8}(-[A-Za-z0-9]{1,8})*"
}

# Unconstrained; it's not entirely clear how IRI fit into
# xsd:anyURI so let's not try to constrain it here
atomUri = text

# Whatever an email address is, it contains at least one @
atomEmailAddress = xsd:string { pattern = ".+@.+" }

# Simple Extension


simpleExtensionElement =
   element * - atom:* {
      text
   }

# Structured Extension

structuredExtensionElement =
   element * - atom:* {
      (attribute * { text }+,
         (text|anyElement)*)
    | (attribute * { text }*,
       (text?, anyElement+, (text|anyElement)*))
   }

# Other Extensibility

extensionElement =
   simpleExtensionElement | structuredExtensionElement

undefinedAttribute =
  attribute * - (xml:base | xml:lang | local:*) { text }
```

```
undefinedContent = (text|anyForeignElement)*

anyElement =
   element * {
      (attribute * { text }
       | text
       | anyElement)*
   }

anyForeignElement =
   element * - atom:* {
      (attribute * { text }
       | text
       | anyElement)*
   }

# XHTML

anyXHTML = element xhtml:* {
   (attribute * { text }
    | text
    | anyXHTML)*
}

xhtmlDiv = element xhtml:div {
   (attribute * { text }
    | text
    | anyXHTML)*
}

# EOF
```

Priedas C: Atom.java – klasė atitinkanti Atom 1.0 formatą darbui JENA instrumentinėje priemonėje

```
import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.ontology.*;
/**
 * Vocabulary definitions from AtomOwl.rdf
 * @author Auto-generated by schemagen on 10 Lap 2006 16:12
 */
public abstract class Atom
                {
    /** <p>The ontology model that holds the vocabulary terms</p> */
    private static OntModel m_model = ModelFactory.createOntologyModel( OntModelSpec.OWL_MEM, null );

    /** <p>The namespace of the vocabulary as a string</p> */
    public static final String NS = "http://bblfish.net/work/atom-owl/2006-06-06/";

    /** <p>The namespace of the vocabulary as a string</p>
     *  @see #NS */
    public static String getURI() {return NS;}

    /** <p>The namespace of the vocabulary as a resource</p> */
    public static final Resource NAMESPACE = m_model.createResource( NS );

    /** <p>See §4.2.1 rfc 4287 spec.</p> */
    public static final ObjectProperty author = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#author" );

    /** <p>see 4.2.5 of rfc 4287. An icon associated with the object</p> */
    public static final ObjectProperty icon = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#icon" );

    /** <p>the object of a link. We interpret here a Link to be a re-ified relation from
     *  a :Version to a :Content object. The relation is re-ified in order to add
     *  a :title relation to the link. This can probably usually be safely ignored</p>
     */
    public static final ObjectProperty to = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#to" );

    /** <p>See §4.2.7 of rfc 4287 spec. A link associated with the container. If the
     *  link is unreified, we have a relation from the container to some resource</p>
     */
    public static final ObjectProperty link = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#link" );

    /** <p>The feed contains the given Entry. See §4.1.1 of rfc4287. This makes no statement
     *  about whether that Entry has the :Feed as its source.</p> */
    public static final ObjectProperty contains = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#contains" );

    /** <p>see §4.2.8 of rfc 4287. An icon associated with the object</p> */
    public static final ObjectProperty logo = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#logo" );

    /** <p>a source of the representation</p> */
    public static final ObjectProperty src = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#src" );

    /** <p>The feed aggregates this given Entry. See §4.1.1 of rfc4287. This means that
     *  the entry does not have the feed as its source. The entry was taken from a
     *  different feed.</p>    */
    public static final ObjectProperty aggregates = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#aggregates" );

    /** <p>See §4.2.14. Title of a container</p> */
    public static final ObjectProperty title = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#title" );

    /** <p>see §3.2.3 of rfc 4287. A mailbox of the Person</p> */
    public static final ObjectProperty email = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#email" );

    /** <p>see §4.2.7.2.1 of rfc 4287. The object points to an alternate version of the
     *  subject</p>
     */
    public static final ObjectProperty alternate = m_model.createObjectProperty( "http://www.iana.org/assignments/relation/alternate" );

    /** <p>see §4.2.4 of rfc 4287. The generator of the object</p> */
    public static final ObjectProperty generator = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#generator" );

    /** <p>See §4.2.10 of rfc 4287 spec. Rights held over a Version.</p> */
    public static final ObjectProperty rights = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#rights" );

    /** <p>See §4.1.3 of rfc 4287 spec. The content of an Entry, a Link or any of the
     *  relation types that are the object of :rel properties</p>
     */
    public static final ObjectProperty content = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#content" );

    /** <p>See §4.2.11 of rfc 4287 spec. The source feed where the entry was found</p> */
    public static final ObjectProperty source = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#source" );

    /** <p>the inverse of the :link relation, not specified in rfc 4287, but added here
     *  for convenience</p>     */
```

```
    public static final ObjectProperty subject = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#subject" );

    /** <p>see §4.2.12 of rfc 4287. Subtitle of the feed.</p> */
    public static final ObjectProperty subtitle = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#subtitle" );

    /** <p>See §4.2.3 of rfc 4287 spec. Someone who contributed to the Version.</p> */
    public static final ObjectProperty contributor = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#contributor" );

    /** <p>see §4.2.2.2 of rfc 4287. Identifies a categorization scheme.</p> */
    public static final ObjectProperty scheme = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#scheme" );

    /** <p>The feed contains the given Entry and that Entry has the feed as its source.
     *  See §4.1.1 of rfc 4287. This relation is a little bit more specialised than
     *  the rfc 4287 entry element. It relates only :Feed objects to :Entry objects
     *  where the :Feed is a :source of the :Entry. See also the :source and :contains
     *  relations.</p>     */
    public static final ObjectProperty entry = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#entry" );

    /** <p>See §4.2.2. A category with which the conainer is associated.</p> */
    public static final ObjectProperty category = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#category" );

    /** <p>See §4.2.13 of rfc 4287 spec. A summary of the content of the Entry</p> */
    public static final ObjectProperty summary = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#summary" );

    /** <p>see §4.2.7.2 of rfc 4287. The relationship type. The relationship type is
     *  a property that relates a :Version to a :Content</p>     */
    public static final ObjectProperty rel = m_model.createObjectProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#rel" );

    /** <p>see §3.1.1.3 of rfc 4287. The datatype of xhtml text elements.</p> */
    public static final DatatypeProperty xhtml = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#xhtml" );

    /** <p>See §4.2.15 of rfc 4287 spec. Indicates the most recent instant in time when
     *  a resource with the given id was modified in a way the publisher considers
     *  significant. Therefore, not all modifications necessarily result in a changed
     *  atom:updated value.</p>     */
    public static final DatatypeProperty updated = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#updated" );

    /** <p>see §4.2.4 of rfc 4287. Indicates the version of the Generator</p> */
    public static final DatatypeProperty generatorVersion = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-
06/#generatorVersion" );

    /** <p>see §4.2.2.3 of rfc 4287. A Human readable label for display.</p> */
    public static final DatatypeProperty label = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#label" );

    /** <p>see §4.1.3.1 of rfc 4287. Relates an :XML object to its content.</p> */
    public static final DatatypeProperty xml = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#xml" );

    /** <p>see §4.2.4 of rfc 4287. A name for the Generator.see §3.2.1 of rfc 4287. A
     *  human readable name for the Person.</p>     */
    public static final DatatypeProperty name = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#name" );

    /** <p>the body of the content.</p> */
    public static final DatatypeProperty body = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#body" );

    /** <p>see §4.2.2.1 of rfc 4287. Identifies the category</p> */
    public static final DatatypeProperty term = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#term" );

    /** <p>see §3.1.1.2 of rfc 4287. The datatype of html text elements. It is considered
     *  to be a relation to the text.</p>     */
    public static final DatatypeProperty html = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#html" );

    /** <p>see §3.1.1.1 of rfc 4287. The datatype of xhtml text elements</p> */
    public static final DatatypeProperty text = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#text" );

    /** <p>see 4.2.4 of rfc 4287. A uri associated the Generatorsee §3.2.2 of rfc 4287.
     *  A uri associated the Person</p>     */
    public static final DatatypeProperty uri = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#uri" );

    /** <p>see §4.2.7.3 of rfc 4287. The mime type of the representation.</p> */
    public static final DatatypeProperty type = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#type" );

    /** <p>see §4.2.7.4 of rfc 4287. The language of the representation.</p> */
    public static final DatatypeProperty lang = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#lang" );

    /** <p>see §4.2.7.6 of rfc 4287. The length in bytes of the representation.</p> */
    public static final DatatypeProperty length = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#length" );

    /** <p>See §4.2.9 of rfc 4287 spec. A date associated with an event early in the
     *  lifecyle of the subject.</p>     */
    public static final DatatypeProperty published = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#published" );

    /** <p>See §4.2.6 rfc 4287 spec. All Versions with the same id can be considered
     *  to be versions of the resource identified by the id. The id mentions the resource
     *  of which it is a representation.</p>     */
    public static final DatatypeProperty id = m_model.createDatatypeProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#id" );

    /** <p>A comment by Reto Bachmann-Gmuer</p> */
    public static final AnnotationProperty retosNote = m_model.createAnnotationProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#retosNote"
);
```

```
    /** <p>A comment by Elias Torres</p> */
    public static final AnnotationProperty eliasNote = m_model.createAnnotationProperty( "http://bblfish.net/work/atom-owl/2006-06-06/#eliasNote" );

    /** <p>A comment by Danny Ayers</p> */
    public static final AnnotationProperty dannyNote = m_model.createAnnotationProperty( "http://bblfish.net/work/atom-owl/2006-06-
06/#dannyNote" );

    /** <p>Container for feed metadata.</p> */
    public static final OntClass Feed = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Feed" );

    /** <p>see §4.1.3 in rfc 4287 One way to think of this is as a class that represents
     *  what one can get from an HTTP connection. It has a body, and a number of headers,
     *  info such as mime-type, content-lenght, etc... .</p>     */
    public static final OntClass Content = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Content" );

    /** <p>see §3.1.1.3 of rfc 4287. The datatype of xhtml strings</p> */
    public static final OntClass XHTML = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#XHTML" );

    /** <p>see §4.2.7.2 of rfc 4287. Relation types are registered, or can be created
     *  as described in rfc 4287. They are special types of properties that relate
     *  :Versions to :Contents.</p>     */
    public static final OntClass RelationType = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#RelationType" );

    /** <p>see §4.2.2 of rfc 4287. A Category Type</p> */
    public static final OntClass Category = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Category" );

    /** <p>see §3.1.1.1 of rfc 4287. The datatype of simple text elements. Essentially
     *  this is the class of all plain text literals.</p>     */
    public static final OntClass PlainText = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#PlainText" );

    /** <p>see §4.1.2 of the rfc 4287 spec</p> */
    public static final OntClass Entry = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Entry" );

    /** <p>Union of the Feed and Entry class. Simplifies writing the ontology.</p> */
    public static final OntClass FeedOrEntry = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#FeedOrEntry" );

    /** <p>Metadata about the state of a resource with given :id at an :updated time.</p> */
    public static final OntClass Version = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Version" );

    /** <p>see §3.1 of rfc 4287. The text constructs.</p> */
    public static final OntClass TextContent = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#TextContent" );

    /** <p>see §4.1.3.3 of rfc 4287. The datatype of xml content</p> */
    public static final OntClass XML = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#XML" );

    /** <p>see §4.2.4 of rfc 4287 spec. Generator for the Feed. It has many properties
     *  in common with :Person</p>
     */
    public static final OntClass Generator = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Generator" );

    /** <p>see §3.1.1.2 of rfc 4287. The class of html strings</p> */
    public static final OntClass HTML = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#HTML" );

    /** <p>see §4.2.7 of rfc 4287 spec. Note that the href, hreflang and lengtlength
     *  attributes of rfc 4287 have been moved to the content class and a content
     *  relation from the Link to a Content class has been made. The href is translated
     *  onto the :src relation of the Content class</p>
     */
    public static final OntClass Link = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Link" );

    /** <p>see §3.2 of rfc 4287 spec.</p> */
    public static final OntClass Person = m_model.createClass( "http://bblfish.net/work/atom-owl/2006-06-06/#Person" );

    /** <p>see §4.2.7.2.5 of rfc 4287. The object provided a source of the information
     *  found in the subject.</p>
     */
    public static final Individual via = m_model.createIndividual( "http://www.iana.org/assignments/relation/via", RelationType );

    /** <p>see §4.2.7.2.2 of rfc 4287. The object points to a related version of the
     *  subject. So if the subject is a Entry the object might be a resource with
     *  an html representation of that entry.</p>
     */
    public static final Individual related = m_model.createIndividual( "http://www.iana.org/assignments/relation/related", RelationType );

    /** <p>see §4.2.7.2.3 of rfc 4287. The object is equivalent to the subject.</p> */
    public static final Individual self = m_model.createIndividual( "http://www.iana.org/assignments/relation/self", RelationType );

    /** <p>see §4.2.7.2.4 of rfc 4287. The object is a related resource that is potentially
     *  large in size and requires special handling.</p>
     */
    public static final Individual enclosure = m_model.createIndividual( "http://www.iana.org/assignments/relation/enclosure", RelationType );

            }
```

Priedas D: Pagrindinių terminų žodynas

Sudarytas remiantis leidiniu: Maskeliūnas S. „Žinių technologijų pagrindinių terminų žodynėlis“; MII preprintas Nr. 2006-35, Vilnius, Matematikos ir informatikos institutas, 2006, 10 p. http://www.mii.lt/files/mii_prep_2006_35.doc“.

1. Agregavimas

Sindikavimo formatais publikuojamos informacijos surinkimas (parsiuntimas) iš pasaulinio tinklo.

angl.: *Aggregation*

2. Atom

XML failų formato šeimos sindikavimo formatas, kurio specifikacijas ir kūrimą aprašo RFC 4287 (http://atompub.org/rfc4287.html)

angl.: *Atom*

3. Duomenys

(1) Faktai ar teiginiai, kurie yra ar gali būti surinkti, išsaugoti, apdoroti, tačiau nėra organizuoti ar pateikti kontekste;
(2) Diskretiniai, objektyvūs faktai apie įvykius [A.Čaplinskas];
(3) Faktinės informacijos vienetai, gauti matuojant, tiriant ar kt. (pvz. statistiniai duomenys).

angl.: *data*

4. Duomenų bazė (DB)

Duomenų rinkinys, organizuotas atitinkamai koncepciniam duomenų modeliui, kuris aprašo duomenų charakteristikas ir sąryšius tarp atitinkamų dalykinės srities esybių.

angl.: *database (DB)*

5. Duomenų bazių valdymo sistema (DBVS)

Programinės įrangos rinkinys skirtas organizuoti informacijai duomenų bazėje. Paprastai DBVS turi paprogrames (funkcinius blokus) duomenų įvedimui, patikrinimui, kaupimui, išrinkimui, derinimui ir kt.

angl.: *Database Management System (DBMS)*

6. Duomenų saugykla

Centralizuota duomenų bazė, surenkanti, integruojanti informaciją iš įvairių įmonės verslo procesų dalių. Ši informacija po to gali būti analizuojama naudojant duomenų gavybos, žinių atradimo duomenų bazėse priemones.

angl.: *Data Warehouse (DW)*

7. Folksonomija

Internetine visuomene paremta metodologija informacijai išgauti iš internetinių nuorodų jas bendrai aprašinėjant.

angl.: *Folksonomy*

8. Informacija

(1) Duomenų prasmė, reikšmė, numatyta interpretacija (kurią žmonės suteikia duomenims, dėka žinomų susitarimų, naudojamų juos vaizduojant);
(2) Duomenų organizavimas, susiejimas asociacijomis, apribojimas, sudarantis galimybę juos panaudoti;
(3) Siuntėją ir gavėją turintis pranešimas. Duomenys paverčiami informacija pridedant jiems pridėtinę vertę. Vertė pridedama kuomet duomenys: susiejami su kontekstu, klasifikuojami, apdorojami, patikslinami ir agreguojami [A.Čaplinskas];
(4) Faktų rinkinys, iš kurio galima padaryti išvadas.

angl.: *information*

9. Informacinė sistema (IS)

Automatizuota ar rankinė sistema, apimanti visą infrastruktūrą, organizaciją, personalą ir priemones, komponentus informacijos surinkimui, apdorojimui, saugojimui, persiuntimui, pateikimui, platinimui, panaudojimui.

angl.: *Information System (IS)*

10. Informacinė visuomenė

Visuomenė, kurioje informacijos kūrimas, platinimas, valdymas/manipuliavimas tapo svarbiausia ekonomikos ir kultūros veikla. (Prieš ją buvo industrinė visuomenė, dar anksčiau – agrarinė).

angl.: *Information Society*

11. JENA

Instrumentinė priemonė darbui su ontologijomis.

angl.: *JENA framework*

12. Koncepcinė schema, koncepcinis duomenų modelis

Formalus, griežtas koncepcinio modelio aprašas tam tikroje koncepcinių schemų kalboje.
Taikomosios srities, kuriai kuriama duomenų bazė, pagrindinių sąvokų/konceptų ir jų tarpusavio sąryšių visumos atvaizdas.

angl.: *conceptual schema, conceptual data model*

13. Koncepcinis modelis

(1) Rinkinys prielaidų, kurios supaprastina realią problemą ar realią taikomąją sritį iki patenkinamo požiūrio apie modeliavimo tikslus ir susijusias valdymo problemas;
(2) (Minimalus) rinkinys abstrakčių sąvokų, kurios aprašo tam tikrą taikomąją sritį arba užduočių ar sistemų klasę. Naudojamas kuriant tiek duomenų bazes, tiek sistemas;
(3) Taikomosios srities esybių ar objektų tipų bei sąryšių tarp jų apibrėžimas, nepriklausomas nuo naudojamos DBVS.

angl.: *conceptual model*

14. Modelis

Supaprastintas sudėtingos esybės ar proceso aprašas (pvz. formulių rinkiniu), atvaizdas ar pan.

angl.: *model*

15. Ontologija

Tam tikros srities bendrai naudojamos sąvokų/konceptų, esybių tipų, jų tarpusavio priklausomybių, sąryšių, aksiomų, dėsningumų ir kt. visumos formalus aprašas.

angl.: *ontology*

16. OWL

Formali ontologijų aprašymo kalba panaudojama įgyvendinant semantinį pasaulinį tinklą

angl.: *Web Ontology Language*

17. Pasaulinio tinklo paslauga (PtP)

Pasaulinio tinklo taikomųjų programų standartizuotas integravimo būdas, besiremiantis paslaugomis grindžiama architektūra ir naudojant Interneto atviruosius standartus XML (duomenų žymėjimui), SOAP (duomenų persiuntimui), WSDL (paslaugų aprašymui), UDDI (paslaugų žinynų, viešųjų registrų organizavimui).

angl.: *Web Service (WS)*

18. Pasaulinis tinklas

(1) Interneto svarbiausia sudedamoji dalis, pasauliniu mastu išskirstytas susietų tarpusavyje hipertekstinėmis nuorodomis dokumentų, failų ir paslaugų rinkinys. Tai – milžiniška elektroninė informacijos ir paslaugų biblioteka (saugoma Interneto serveriuose, naudojančiuose HTTP duomenų persiuntimo protokolą) su įvairiomis greitos intuityvios paieškos galimybėmis;
(2) Kompiuterių tinklas, susidedantis iš Interneto serverių, kurie hipertekstų persiuntimo (HTTP) protokolu teikia tekstinius, grafinius, garsinius, video išteklius ir per Internetą valdomas paslaugas.

angl.: *World Wide Web (WWW), the Web*

19. RDF

Resursų aprašymo sistema.

angl.: *Resource Description Framework*

20. RSS

1) yra XML failų formatų šeima internetiniam duomenų rinkimui iš naujientiklių (angl. *news websites*) ir blog'ų (angl. *weblogs*).
2) Išsamus tinklapio aprašymas (angl. *Rich Site Summary*) - (RSS 0.91).
3) tikrai paprastas surinkimas (angl. *Really Simple Syndication*) - (RSS 2.0).
4) sindikavimo formatas.

angl.: *Really Simple Syndication*

21. Saugojimo į RDB mechanizmas

Mechanizmas kai programoje operuojama objektais, o duomenų logikos sluoksnis pasirūpina užklausų generavimu ir bendravimu su DBVS duomenų nuskaitymui, atnaujinimui ir trynimui.

angl.: *Persistency*

22. Semantinio pasaulinio tinklo paslauga (SPtP)

(1) Pasaulinio tinklo paslaugų ir semantinio pasaulinio tinklo technologijų derinys; t.y., išplėstoji pasaulinio tinklo paslauga, pritaikyta panaudojimui semantiniame pasauliniame tinkle;
(2) Pasaulinio tinklo paslauga, besiremianti semantika, kuri vaizduojama kompiuterinių programų interpretuojama (kompiuteriniam apdorojimui tinkama) forma; t.y., SPtP taip aprašo savo savybes ir galimybes, kad kompiuterių programos gali automatiškai nustatyti jos paskirtį.

angl.: *Semantic Web Service (SWS)*

23. Semantinis pasaulinis tinklas (SPt)

Pasaulinio tinklo išplėtimas, kuriame duomenims, informacijai nurodoma formaliai apibrėžta prasmė. SpT realizuojamas XML kalbos pagrindu; semantinius teiginius išreiškiančios deklaratyvios kalbos yra RDF ir RDF Schemos kalbų antstatu. SpT skirtas automatinei kompiuterių tarpusavio sąveikai bendrai priimtos terminijos pagrindu, naudojant ontologijų kalbas, logikos formalizmus ir loginio išvedimo galimybes.

angl.: *Semantic Web*

24. Semantinis tinklas

Vienas iš žinių vaizdavimo būdų; žymėtas kryptingas grafas, kuriame viršūnės vaizduoja konceptus, o žymėtieji lankai – semantinius sąryšius tarp konceptų.

angl.: *Semantic network, Semantic net*

25. Sindikavimas

Duomenų, informacijos kaupimas tam skirtais formatais siekiant standartiškai publikuoti.

angl.: *Syndication*

26. SPARQL

Užklausų kalba resursų aprašymo sistemai.

angl.: *Query Language for RDF*

27. Technologija

(1) Praktinis mokslo panaudojimas pramonėje, prekyboje ir kt.;
(2) Procesas, veiksmų ir poveikių (operacijų) seka norimam produktui gauti [M.Šalkauskas].

angl.: *technology*

28. Tinklas

Esybių (tinko mazgų, viršūnių) ir jų tarpusavio sąryšių visuma.

angl.: *net, network, a web*

29. URL

Vieningas resursų lokatorius.

angl.: *Unifirn Resource Locator*

30. URI

Vieningas resursų identifikatorius.

angl.: *Uniform Resource Identificator*

31. Žiniatinklis

(1) Integruotas išskirstytai saugomų žinių tinklas;
(2) Semantinio pasaulinio tinklo poaibis, sutelkiantis dėmesį į žinias.
(3) Interneto dalis, resursai, kuriuos internete galima pasiekti naudojant URL (Vieningus Resursų Identifikatorius). Dėl savo naudojimo platumo jis dabar yra neretai supainiojamas su visu internetu apskritai, tačiau tai yra tik interneto poaibis.

angl.: *knowledge web, Knowledge Web*

32. Žinios

(1) Psichologinis suvokimo, mokymosi ir samprotavimo rezultatas; informacijos ir patirties, konteksto supratimo, interpretavimo, apsvarstymo derinys, naudojamas sprendimams priimti ir veikti. Žinios būna išreikštos (angl. *explicit*) ir neišreikštos (angl. *tacit*);
(2) Personalizuota informacija [A.Čaplinskas].

angl.: *knowledge*

33. Žinių bazė (ŽB)

Žinių rinkinys, išreikštas naudojant tam tikrą formalią žinių vaizdavimo kalbą; žiniomis grindžiamos sistemos dalis.

angl.: *knowledge base (KB)*

34. Žinių vaizdavimas

Dirbtinio intelekto šaka, nagrinėjanti kaip konstruoti ir naudoti sistemas, skirtas žinių kaupimui.

angl.: *Knowledge Representation (KR)*

35. Žinių vaizdavimo modeliai/kalbos

Produkcinės taisyklės (angl. *production rules*), semantiniai tinklai, freimai (angl. *frames*), pirmos eilės predikatų (angl. *first-order predicate*), tikimybinė (angl. *probabilistic*), neraiški (angl. *fuzzy*) ir nemonotoninė (angl. *non-monotonic*) logikos.

angl.: *knowledge representation models/languages*

Priedas E: Publikacija

(pridedama atšviesta kopija)