Predstavljanje znanja u informacijskim sustavima

Seminar

Semantički web

Krešimir Pavić

Sadržaj:

1. Uvod

Semantički web predstavlja sljedeću evolucijsku stepenicu u razvoju World Wide Web-a (WWW ili kraće web). U proteklom razdoblju WWW je prošao ogroman put (od alata za razmjenu i vezivanje dokumenata unutar jednog istraživačkog centra do najpopularnije i najkorištenije usluge na Internetu). Temeljeći se na nizu široko prihvaćenih standarda dostigao je neslućene razmjere. Ti vrlo važni standardi su TCP/IP protokol na koji se nadovezuje HTTP (Hypertext transfer protocol) i jezik za opis stranice HTML (Hypertext markup language). Povijesno gledano, možemo reći kako je web trenutno u svojoj drugoj generacijskoj dobi. Prva generacija okarakterizirana je «ručnim» pisanjem/kodiranjem statičkih HTML stranica te slabim ili nikakvim dinamičkim generiranjem stranica. Sljedeći, očigledni korak je išao prema stvaranju dinamičkih stranica generiranih od strane računala koja su unaprijed iskodirana u nekom prigodnom jeziku (CGI, ASP, JSP/Servlet, PHP ...). Osnovna odlika spomenutih generacija je ta da su namjenjene za rad s ljudima/fizičkim osobama, a ne s drugim strojevima. Osobe ih mogu logički procesuirati - čitati, *browsati*, pretraživati te ispunjavati razne web forme. Treća generacija cilja na novi web koji će omogućiti i strojevima da procesuiraju na višem nivou informacije koje se na njemu nalaze. Ta generacija naziva se semantički web. Semantički podrazumjeva pridjeljivanje značenja informacijama na webu. Na prvi pogled ovo može biti zbunjujuće jer je web prepun informacija, no opet se treba prisjetiti da ih samo osobe mogu razumjeti dok je strojevima, usprkos raznim *meta tagovima*, on i dalje nerazumljiv. Web trenutno ne sadržava nikakvo značenje, on dobiva značenje tek nakon ljudske interpretacije.

Pod patronatom World Wide Web Consortium (W3C) objedinjuje se razvoj današnjeg i budućeg weba. W3C je od kasnih 90tih godina prošlog stoljeća krenuo u promoviranje i razvoj semantičkog weba. Osnovna ideju počeo je promovirati Tim Barnes-Lee, 1998. godine, kao «plan za postizanje povezanih podatkovinih aplikacija na webu u takovoj formi da oblikuju konzistentni logičku mrežu (web) podataka (sematički web). T. Barnes-Lee se smatra ocem web-a i jedan od glavnih osoba odgovornih za novi, semantički web, čiju je viziju predstavio u dva povezana dokumenta koja opisuju semantički web iz «vrlo velikih prespektiva/visina» (20.000 i 50.000 stopa).

1.1 Use case scenario

Smisao i svakodnevnu očekivanu upotrebu semantičkog weba najbolje je prikazati na običnom primjeru. Pretpostavimo da želimo sutra otputovati iz Zagreba u Dubrovnik i natrag u Zagreb za tjedan dana. Koristeći web odabrat ćemo najbrže prijevozno sredstvo. Pretpostavimo da postoji *site* ili više njih koji

sadržava/ju sve informacije o redovima vožnji autobusa, vlakova te redova letenja sa zagrebačke zračne luke. Kao prvi izbor izabiremo zrakoplov, ipak je najbrži, no nakon posjeta web siteu Državnog hidrometeorološkog zavoda primjećujemo da se sutradan očekuje gusta magla iznad aerodroma koja će potrajati cijeli dan. Kako mi moramo obavezno sutra stići u Dubrovnik, nastavljamo dalje s potragom. Autobus predstavlja sljedeći izbor, no opet HAK na svojim stranicama javlja da su već danima ogromne gužve na autocesti zbog radova nastalih kao posljedica prošle oluje. I na kraju odabiremo vlak. Kako on ne ide do Dubrovnika, morat ćemo presjesti u Splitu, te se bacamo na daljnje istraživanje kako doći od Splita do Dubrovnika. Na ovo istraživanje može izgubit poprilično vremena i opet ne biti sasvim sigurni da smo izabrali optimalni put i pripadajuće preijvozno sredstvo. Pitanje koje slijedi jest, zašto to ne bi umjeto nas obavila računala. Mi svojem «agentu» zadamo odredište, vrijeme dolaska te očekujemo da nam «u tren oka» ponudi optimalno riješenje, odnosno da obavi gore navedne zadatke. No, odmah nastaju problemi jer računala ne mogu shvatiti što je to npr. automobili, čak i ako im to «objasnimo» ostaje problem da svi site-ovi ne koriste riječ automobil, nego vozilo itd. Na žalost, trenutno stanje je takvo da računala ne mogu obaviti naš zadataka iz više razloga:

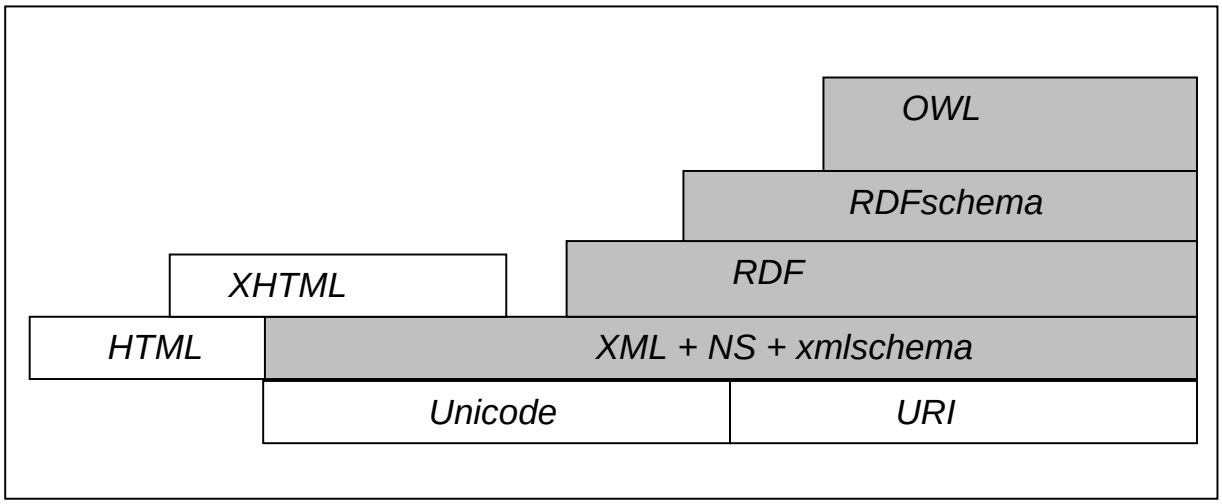
- neki podataci nisu dostupni
- podaci nisu unešeni u strojno čitljivom obliku
- podaci nisu izloženi u obliku da ih strojevi mogu procesuirati
- ne postoji *browser* (agent) u kojem se mogu sročiti navedeni problemi
- ne postoji definiran jezik za takve upite,...

Čak i da svi podaci postoje oni ne bi bili razumljivi. Npr. na početnoom site-u stoji popis svih «prijevoznih sredstava» (auto, vlak, autobus, zrakoplov). Slobodno se možemo zapitati kako će računalo znati da je auto prijevozno sredstvo koje vozi po autocesti te otići na HAK-ov site provjeiti da li su ceste slobodne. Tamo vjerojatno stoji da za određenu skupinu vozila neke ceste su nedostupne i slično. Da li je auto dio skupine vozila? Koje skupine vozila? Ovakve probleme može trenutno razriještit samo ljudska osoba. Jedna od namjena semantičkog weba je da nam olakša ovakve i slične poslove. On želi stvoriti takvu okolinu gdje će softverski agenti lunjati od stranice do stranice te obavljati sofisticirane zadatke za razne korisnike (fizičke osobe ili druge agente).

2 Jezici semantičkog weba

Izgradnja semantičkog weba nije jednostavan proces i on se može postići samo ako su upostavljene nove razine međuoperabilnosti temeljeni na otvorenim standardima. Sandardi moraju pružati ne samo dobru definiciju za sintaktičku formu dokumenata, nego i za njihov sintaksni sadržaj. Ovakva semantička interoperabilnost je glavni zadatak W3C-a koja se oslanja na postojeće

standarde te uvodi neke nove u obliku novih jezika namjenjenih webu. Shematski prikaz tih jezika dan je na sljedećoj slici:



slika 1.
Prikaz i odnos jezika semantičkog weba

Svaki od tih jezika ima određenu i strogo definiranu funkciju. Viši sloj koristi funkcionalnost nižeg sloja.

3 XML

3.1 SGML

XML (Exensible Markup Language) je pojednostavljena forma SGML-a (Standard Generalized Markup Language) standarda razvijenog početkom 80-tih godina prošlog stoljeća. SGML je razvijen u IBM-ovim laboratorijima i bio je ekstremno velik i širok standard za opis svih dokumenata koji je u svom obliku bio prezahtjevan za implementaciju što je dovelo do razvijanja novog standarda koji bi trebao biti temeljen na SGML-u, no jednostavniji za implementirati. Velike zasluge za postavljanje XML-a na noge, pripisuju se Jon Bosaku (Sun) koji je pokrenuo W3C radnu grupu (*working group*) čija je odgovornost bila skalirati SGML-a u jednostavniju formu XML.

3.2 Definicija XML-a

XML je napravljen s namjerom da bude univerzalni markup/opisni-jezik koji opisuje i tvori bilo kakvu strukturu dokumenta neovisno o krajnjem prikazu dokumenta. XML dokument se sastoji od niza ugnježđenih elemenata nazvnih tagovi, unutar jednog izvorišnog taga (*root element*). Svaki od elemenata

(tagova) može imati proizvoljan broj atributa (svojstava). Da bi se neki dokument mogao nazvati XML dokumentom on mora zadovoljavati sljedeće stvari:

- mora koristiti DTD (ili XML schemu) ili se deklarirati kao samostojeći dokument
- sve vrijednosti atributa moraju biti navedeni unutar navodnika
- element (tag) mora imati otvarajući i zatvarajući element osim ako nije prazan element
- ako je elemet samostojan onda mora posjedovati zatvarajući slash (/) prije kraja taga
- svi tagovi se moraju pravilno ugnijezditi
- svaki XML dokument bez DTD-a moraju posjedovati atribute tipa CDATA
- određeni elementi moraju se referenecirati na drugi način kako ne bi razbili strukturu dokumenta (tipa <, >, &, ...)
- postoji jedan izvorišni element

Za bilo koji XML dokument možemo reći kako on predstavlja označeno (labelirano) stablo (*tree*), pri čemu svaki tag je u relaciji s labeliranim čvorom u podatkovnom modelu i svaki ugnježđeni tag predstavlja dijete u stablu. XML dokumentom možemo na više načina predstaviti bilo kakvu strukturu pri čemu treba naznačiti da ne postoji jednoznačan način predstavljanja te strukture odnosno moguće je napraviti velik broj različitih XML dokumenata koji predstavljaju jednu te istu stvar, a opet nisu jednaki dokumenti (sintaksno).

Moguće je za svaki XML dokument definirati strukturu tagova (koji se unalazi unutar kojeg, koliko ih je moguće, kakve atribute smiju posjedovati i sl.). Za tu definiciju služi DTD (Document Type Definition), no DTD ima ograničene mogućnosti za definiranje XML struktura što se posebno očituje u nemogućnosti definiranja tipova podataka unutar tagova ili atributa. Zbog toga je ustanovljena nova specifikacija XML Schema koja je nadomjestila i zamijenila DTD, a sadrži u biti isto što i DTD samo prošireno i formalnije. Tako da se može napiasti kako je XML Schema zapravo gramatika za XML. U praksi XML se koristi za:

- serijalizaciju drugih markup jezika
- semantički *markup* jezik za web stranice (nakon što se takav dokument propusti kroz XSLT dobije se valjana (X)HTML ili WAP stranica)
- jedinstven format za izmjenu podataka (*data-exchange*)

Može se postaviti pitanje zašto ne iskoristiti XML kao osnovni i jedini jezik za izradu semantičkog weba, odnosno zašto uvoditi u priču druge, nove jezike ako nam je XML dostatan. Na to pitanje možemo odgovoriti nakon što postavimo zahtjeve za jezik kojim bi se definirao semantički web, a oni su:

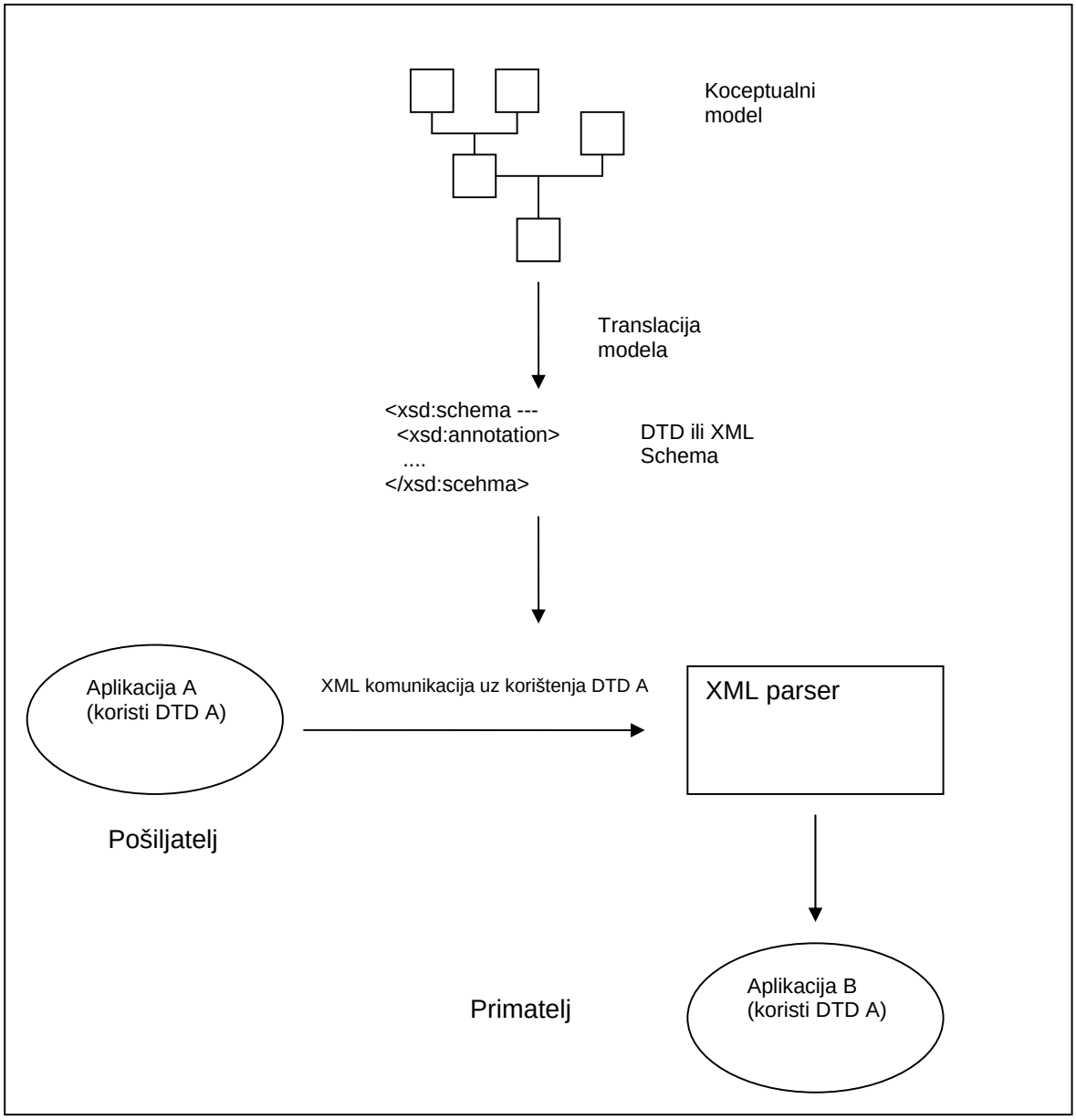
1. univerzalana ekspresivnost – kako nije moguće predvidjeti sve potencijalne upotreba, jezik mora imati dovoljno ekspresivnosti da pokrije bilo koji oblik podataka

2. podrška za sintaksnu interoperabilnost – pod ovim se podrazumjeva lakoća kojom se čitaju podaci te kako jednostavno razne aplikacije koriste te podatke (razni parseri)
3. podrška za semnatičku interoperabilnost – podrazmjevaju se teškoće (ne)razmjevanja podataka (njihov smisao, odnosno maprianje podataka između nepozantih i poznatih pojmova). Ovo je najvažniji zahtjev koji mora biti odlično podržan.

Ako se na XML gleda kao na formalizam za definiranje gramatike, onda je prvi zahtjev u potpunosti ispunjen jer (gotovo) sve se može zapisati u XML dokumentu. Za XML postoji cijeli niz parsera (DOM, SAX, Streaming SAX) pri čemu su ti parseri višestruko iskoristive komponente. S postojanjem parsera ispunjen je i drugi zahtjev. Treći zahtjev (semantička interoperabilnost) je teško u potpunosti ispuniti XML-om i tu dolazi do velikih problema.

XML je namjenjen definiranju strukture dokumenata, no ne i njegovog značenja pri čemu je takvo nepodržavanja bilo kakve interpretacije podataka sadržanih u dokumentu glavni nedostatak XML-a kao univerzalnog jezika za semantički web. Taj nedostatak je najlakše ilustrirati preko primjera.

Pretpostavimo da postoje dvije aplikacije koje pokušavaju komunicirati jedna s drugom (*one-to-one* scenario). Prvo u izgradnji takve komunikacije potrebno je premapirati *domain model* u odgovarajući objektni (najčešće UML-om) te onda taj objektni model premapirati u dobar DTD (ili XML Schema) dokument. Kako je napisano, to premapiranje iz objektnog modela u DTD nije jednoznačno tako da je moguče da svaki od sudinoka komunikacije može imati vlastiti DTD. No možemo pretpostaviti, kako su se sudionici unaprijed dogovorili o obliku DTD-a. U tom slučaju komunikacija je moguća. Pravi problemi nastaju kada se u tu komunikaciju pokušava ugurati dodatan sudionik koji posjeduje svoj *domain model* s vlastitim DTD-om. Tada nije moguće automatski ugurati sudionika, nego je potreban detaljan i opsežan rad na usuglašavanju modela, DTD-ova, njihovo mapiranje. Što je opet potrebno učiniti za svakog novog sudionika. U ovom slučaju, koji je sasvim normalan i uobičajen u svakodnevnici, XML kao univerzalni jezik nije dobar.



slika 2.
Primjer aplikacija koje koriste XML za definiranje i proslijeđivanje podataka.

4 RDF

Kao osnova semantičkog weba uzima se jezik RDF. RDF je skračenica za Resource Description Framework i u svojim počecima predstavljao je jednostavn jezik za prezentiranje informacija o web resursima. Resurs treba promatrati u najširem mogućem obliku. Te informacije su tipično uključivale podatke o nastanku resurasa (web stranice), autoru, *copyrightu* i ostale slične stvari koje su se uobičajno zapisivale u meta tagovima HTML stranica. Ovakva upotreba RDF-a posebno ne pridonosi uspostavi željenog semantičkog weba. Zato se prionulo na generalizaciju pojma «web resurs» koji u novoj iteraciji predstavlja bilo što, što se može identificirati na webu čak i ako se ne može direktno dohvatiti s weba. Primjer za to su pojedinačne on-line narudžbe s web shopova. U tom smislu RDF služi kao *framework* za izražavanje informacija o općenitim resursima koji mogu obrađivati različite aplikacije i nisu primarno namjenje za direktan prikaz ljudskom korisniku.

4.1 Definicija RDF-a

Nova defincija RDF-a glasila bi: RDF je sintaksno neovisan, apstraktni model koji određuje standard o meta podacima (podaci o podacima) koji služe za opis resursa na webu. Kada u ovom obliku govorimo o meta podacima, njih isto tako uzimamo u najširem mogućem opsegu, ne ograničajući se ne prvotnu namjenu (naslov, autor ...). RDF je standard kojeg je uspostavio i dalje razvija W3C te ga «reklamira» kao osnovu semantičkog weba, na temelju kojeg se izgrađuju svi ostali jezici semantičkog weba.

RDF se temelji na prijašnjim radovima koji su rezultirali modelima za prikaz imenovanih svojstava i njihovih vrijednosti. Osnova konstrukcija RDF je tvrdnja *(assertion, statement*) oblika:

```
objekt – atribut – vrijednost.
```

Objekt O posjeduje atribut A s vrijednošću V. Uobičajna su dva načina prikazivanja nevedne tvrdnje. Prvi je

```
A(O,V) ,
```

dok je drugi:

```
'[O] – A -> [V]
```

Primjer za ovaj prvi oblik je:

```
hasPrice('http:// www.bookshop.com/book/Test', '$123'),
```

dok je drugi odmah razumljiv i prirodan je za grafički prikaz u obliku usmjerenog grafa. Iz gore navednog primjećuje se da se RDF data model sastoji od nekoliko članova:

(a) Resurs (*resource*) predstavlja bilo što što se može dobaviti na webu. U biti sve što posjeduje URI (Unifrom Resource Identyfier) je resurs. Što podrazumjeva da je to jedan HTML dokument, jedan XML dokument ili kolekcija bilo kojih od navedenih dokumenata, cijeli web site ili jednu osbobu... URI specifikacija pruža dovoljno proširivosti da možemo reći kako bilo što može posjedovati jedinstveni identifikator.

(b) Atributi, ili još poznati pod drugim imnom kao svojstva (*properties*), predstavljaju specifičan aspekt nekog resursa. Svaki atribut poseduje vlastito značenje koje dopušta određeni opseg vrijednosti ili može biti povezan samo s određenim resursima. Ovakve ograde nisu dio RDF specifikacije, nego su određene u RDF Schema standardu koji je sljedeća logična nadogradnja na RDF (otprilike kao DTD, odnosno XML Schema na XML standard).

(c) Vrijednosti koja može biti ili literal ili neki drugi resurs ili bilo koji drugi primitiv definiran XML-om

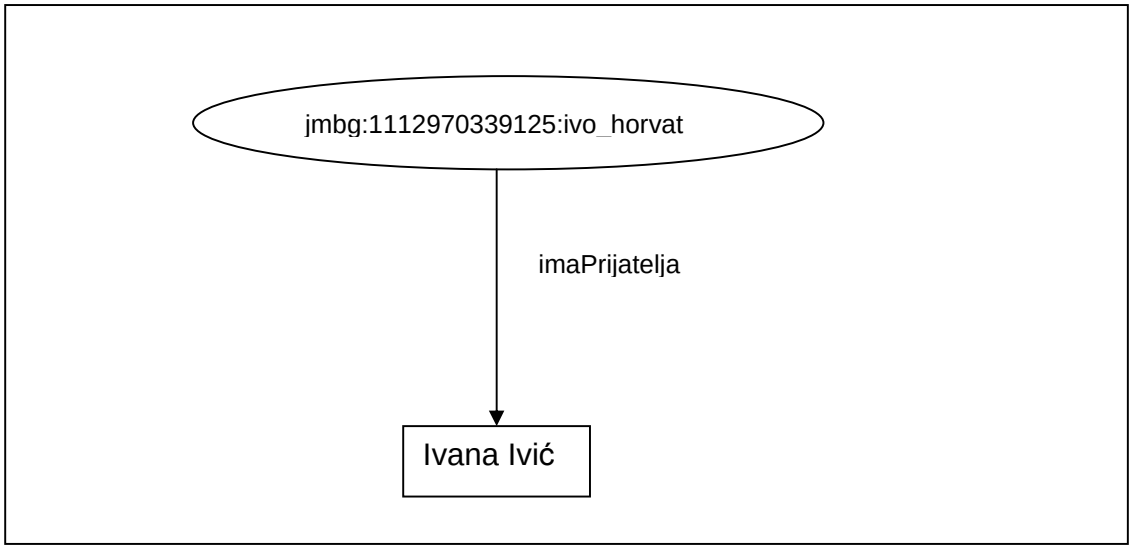
(d) Sama tvrdnja (*statement*) je spomenuta trojka koja se sastoji od trojke: subjekta, predikata i objekta.

4.2 RDF svojstva

Na nizu primjera pokazat ćemo sve osobine RDF-a. Za jednostavnu izjavu: Osoba Ivo Horvat ima prijateljicu Ivanu Ivić, lagano određujemo subjekt (Ivo Horvat), predikata (ima prijateljicu) i objekt (Ivanu Ivić) tako lagano dobivamo grafički prikaz izjave. No, prije samog grafa trebamo jedinstveno odrediti Ivu Horvata što je moguće napraviti jednostavnim definiranjem njegovog jedinstvenog URI-a koji za ove potrebe može biti u obliku:

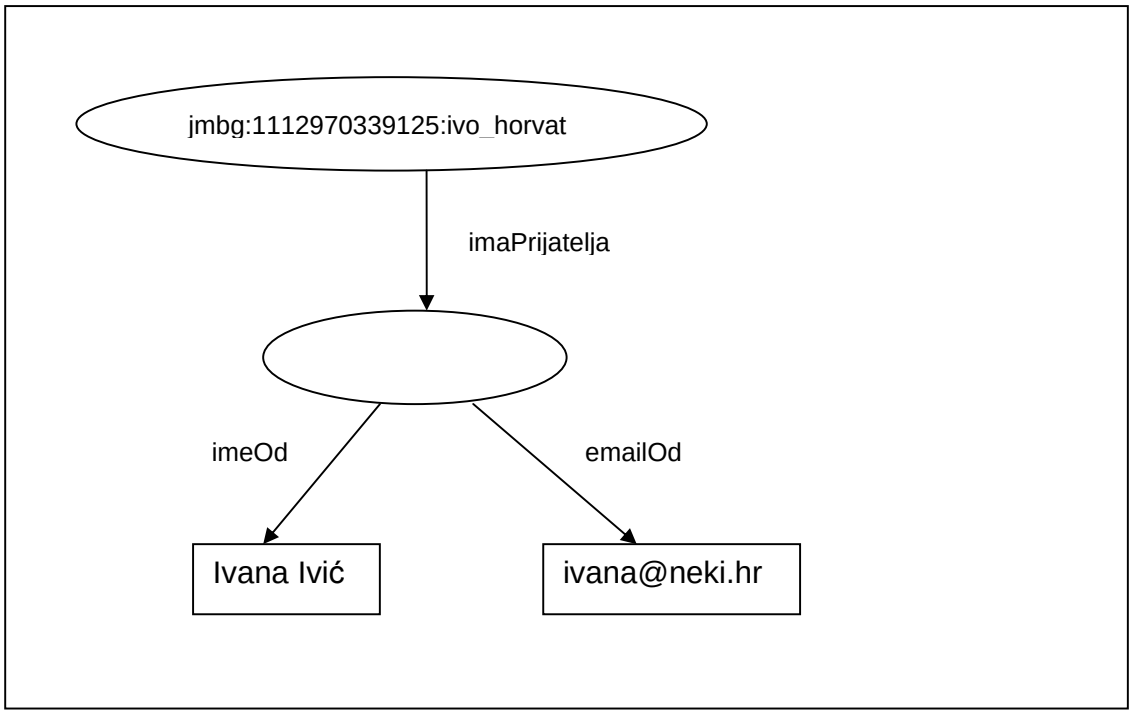
```
jmbg:1112970339125:ivo_horvat
```

što onda daje sljedeći graf.



slika 3.
Jedna RDF izjava

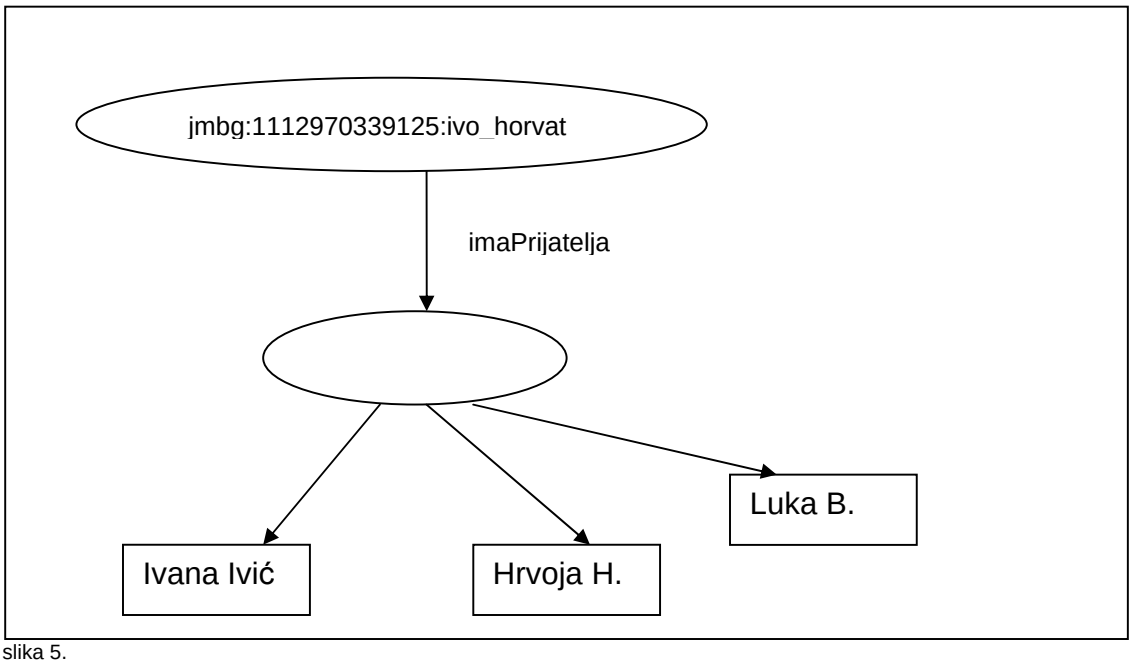
Konstruiranje grafa je jednostavno. U ovalu se prikazuje URI tj. subjekt. Predikat je predstavljen strelicom dok je u pravokutniku objekt koji je u našem slučaju literal. Naravno, objekt može bit i resurs tj. neki drugi URI.



slika 4
Složenija RDF izjava uz upotrebu anonimnog resursa

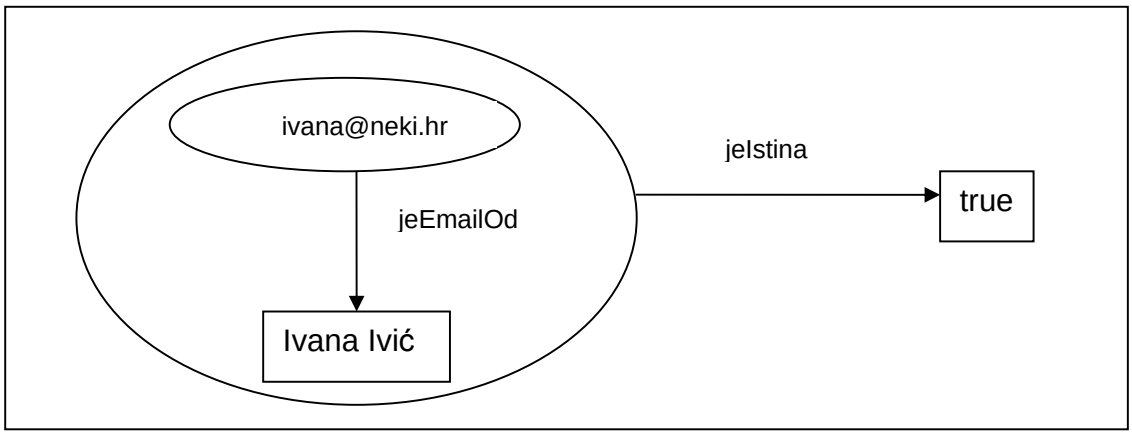
U gornjem prijmeru objekt je anonimni resurs koji je opisan s dvije izjave `imeOd` i `emailOd`. U RDF modelu postoji struktura koja se naziva kontejner i ona omogućava ponavljanje elemenata. Postoje tri tipa kontejnera:

1. «bag» neuređena lista vrijednosti
2. «sequence» (sekvenca) je uređeni slijed
3. «alternativa» određuje jednu od mogućih vrijednosti



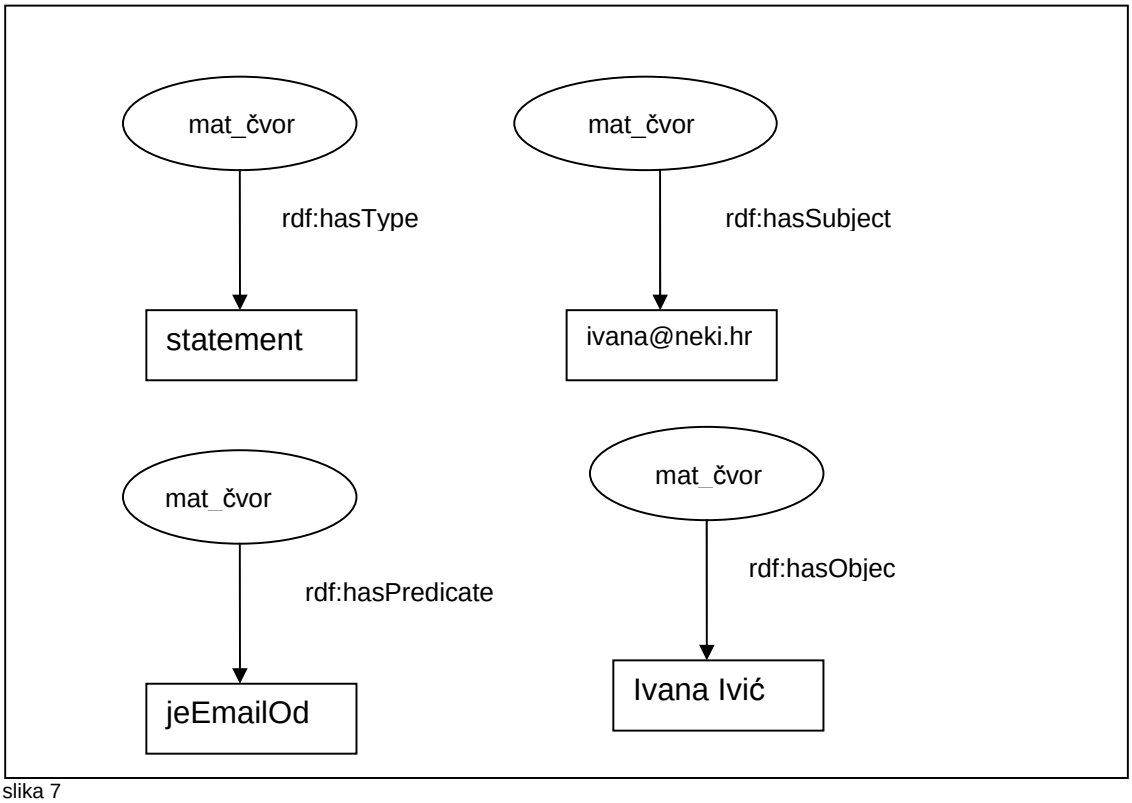
slika 5.
Primjer kontejnera u RDF izjavi

Ovim razmišljanjem možemo se zapitati da li je moguće da URI postane druga izjava (statement).



slika 6.
Primjer materijalizacije u RDF-u.

U biti sada govorimo o izjavama o izjavi (*statement about statement*). U RDF svijetu to se naziva meterijalizacija (*reification*). Ipak, interno RDF izjavu o izjavi interpretira kao četri izjave. Dekompozicija izjave o izjave provodi tako da prvo «materijaliziranu» izjavu ounačimo s mat_čvorom te tom čvoru pridoijelimo četri nove. Dodavanje novih izjva ne smije utjecati smisao i značenje osnovnih izjava. U tom slučaju moramo definirati poseban riječnik (vokabular) da bi izrazili kocepte kao što su subjekt, predikat i objekt. Kako je RDF sintaksno neovisan jezik, osim grafički, on se može prikazati koristeći bilo koju notaciju, no gotovo uvijek se zapisuje u XML notaciji. Takva kombinacija jezika i zapisa se naziva RDF/XML. Kao željeni vokabula upotrebljava rdf: *namespace*.



slika 7
Interni prikaz izjave o izjavi

4.3 XML notacija

XML notacija RDF-a uvijek započinje tagom rdf:RDF. Svaka RDF izjava (trojka) mora se nalaziti unutar tog taga i ona se prikazuje kroz rdf:Description element koji sadrži jedan ili više predikata i objekata. Identifikator (URI) RDF izjave zapisuje se kao rdf:about atribut unutar rdf:Description elementa. Tako spomenuti prikaz s početaka poglavlja možemo zapisati i u sljedećem oblik:

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
   xmlns:f="http://friends.org/schema/1.0/">
```

```
    <rdf:Description about="jmbg:1112970339125:ivo_horvat ">
       <f:Friend>Ivana Ivić</f:Friend>
    </rdf:Description>

</rdf:RDF>
```

Napomena f: namespace je izmišljen radi jednostavnijeg primjera. Odmah je moguće zamjetiti kakvu ulogu imaju rječnici za RDF. Oni daju konteksno značenje RDF izjavama te njihova standardizacija i njihovo bogatsvo (količinski) omogučit će široku primjenu RDF-a i semantičkog weba. Za računala je XML prikaz RDF-a jako značajan jer omogućuje jednostavno kodiranje, prijenos te parsiranje.

Kao što je spomenuto, RDF podržava kontejnere. Element rdf:Container je osnovna klasa za njgov XML zapis, no u praksi se zapravo koriste izvedne klase iz `rdf:Conteiner`-a i one su:

(1) `rdf:Bag` - za neuređen skup
(2) `rdf:Seq` - za slijedni skup (sequence)
(3) `rdf:Alt` - označava alternative unutar skupa

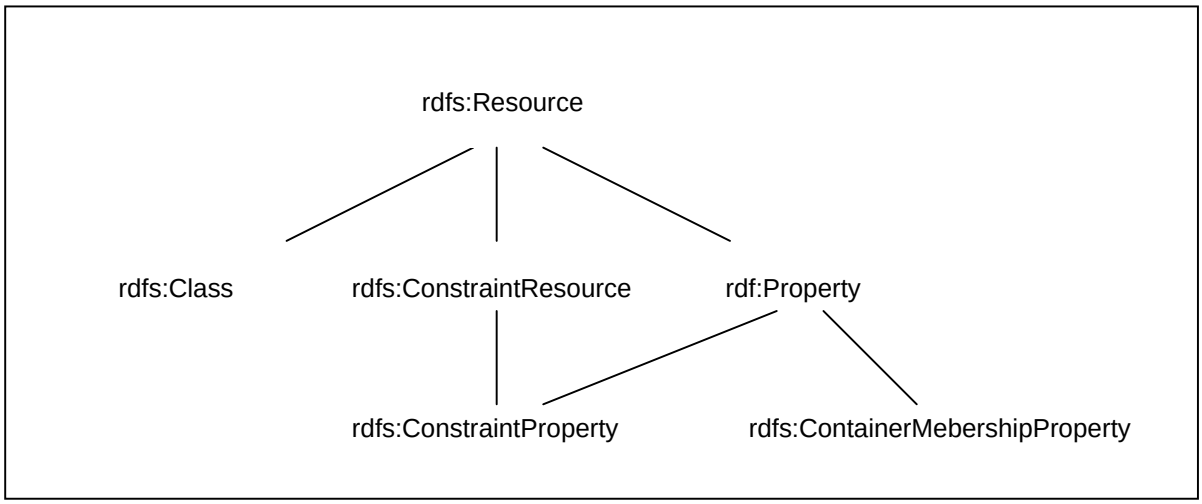
4.4 RDF Schema

Jedan od problema u radu s RDF-om je taj da nije moguće specificrati koji se tipovi podataka pojavljuju u binarnim predikatima, niti kako oni se odnose jedni prema drugima. RDF Schema pokušava nadomjestiti taj nedostatak dopuštajući definiranje tipova slično kao što su oni definirani u objektnim modelima (oo design). Osim definiranja samih tipova određuju se i odnosi između njih. Na RDF Schemu se može gledati kao na nadopuni RDF-a u sličnim odnosu kao što i XML Schema (ili DTD) nadopunjuje XML specifikaciju.U XML notaciji RDF Schema uobičajno koristi namespace `rdfs:`.

Tip se definira korištenjem `rdfs:Class` elementa i dodatno se specijalizira korištenjem `rdfs:subClassOf`. Za razliku od objektno orijentiranog dizajna, atributi nisu definirani u klasi, nego su dinamčki dodijeljeni klasi tako što se koriste dodatne izjave (*statement*) gdje su klase subjekti dok su atributi objekti tih izjava.

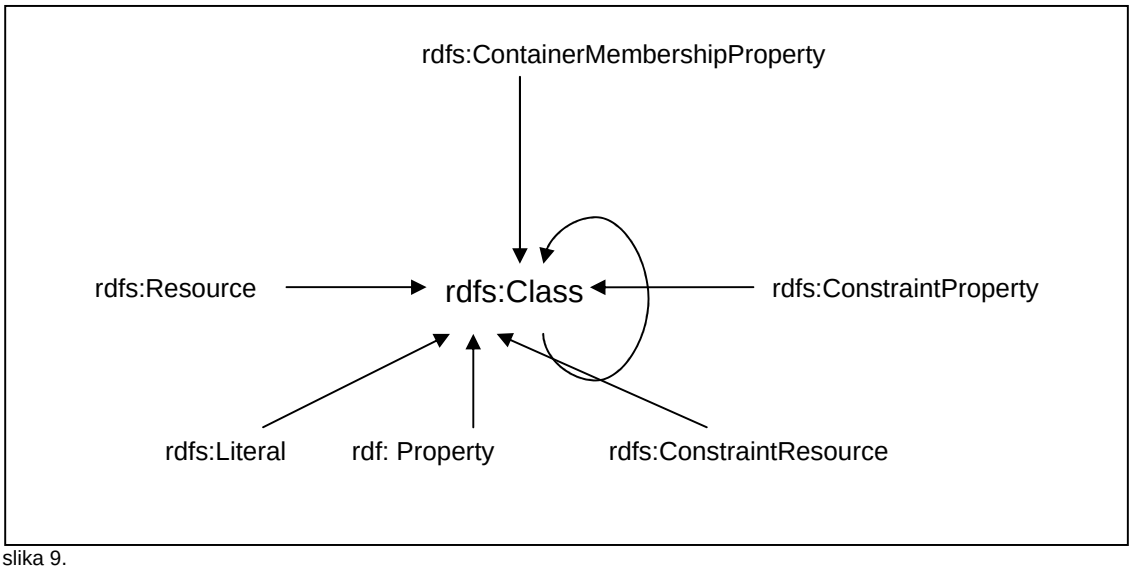
Osnovne klase su `rdfs:Resource`, `rdf:Property` i `rdfs:Class`. Sve što se definira RDF izjavama kreće od instance `rdf:Resource` tako da i nju nasljeđuje osnovna `rdf:Class`. `rdf:Property` je osnovna klasa za sva svojstva. Osnovna svojstva klasa su `rdf:type` (tip), `rdfs:subClassOf` i `rdfs:subPropertyOf`. `rdf:type` određuje relaciju *instance-of* koja je preuzeta iz objektnih modela. Dozvoljeno je

višestruko naslijeđivanje tako da jedna klasa može biti podklasa više drugih klasa.



slika 8.
RDF Schema primitivi i njihov odnos

Osnovna ograničenja ili *constraints* definirana RDF Schema specifikacijom su `rdfs:ConstraintResource`, `rdfs:ConstraintProperty`, `rdfs:range`, i `rdfs:domain`. `rdfs:ConstraintResource` definira osnovnu klasu svih ograničenja (slika 8.). `rdfs:ConstraintProperty` pokriva sva svojstva koja se koriste za definiranje ograničenja. Trenutno postoje dvije instance `rdfs:range` i `rdfs:domain` koja se koriste za restrikciju dosega i domene svojstva. Nije dopušteno izražavanje dva ili više opsega na svojstvu. Za domene ne postoji takvo ograničenje.



slika 9.
RDF Schema - instance-of odnos primitiva

5 Rječnici

5.1 Dublin Core

Dublin Core predstavlja jedan od najstarijih rječnika koji svoje početke ne veže uz semantički web, nego uz potrebu za standardizacijom elektroničkih resursa na «starom» webu. Kasnije se idealno uklopio u svijet semantičkog weba. Rad na Dublin Coru započeo je 1995. inicijalnom radionicom (*workshop*) u kojoj su sudjelovali razni knjižničari, specijalsiti za text-markup i istraživači digitalnih knjižnica. Osnovni problem koji su pokužali razrješiti je kako najbolje katagolizirati sadržaj na webu (bilo kao digitalna knjižnica, bilo kao obična web stranica). Nakon niza provednih radionica 1996. godine napravljen je skup od 15 standardnih elemenata (termina) poznatijih pod nazivom Dublin Core (DC). Svi elementi su opcionalni te se mogu proizvoljno ponavljati u bilo kojem redoslijedu. Nadogradivi su (proširivi) te podržavaju internacionalizaciju (upotrebu raznih jezika). Moguće je da redoslijed elementata može imati ulogu (npr. prvo pojavljivanje jednog elementa nosi veću težinu od ostalih), no ono se ne garantirana da je jednako u svim sustavima. Prvenstveno DC je namjenjen otkrivanju dokumenta, a tek onda kategorizaciji i indeksiranju.

Osnovnih 15 elementa su grupirani u tri kategorije:

(1) sadržaj
(2) intelektualna svojina
(3) instanca

Sjedeća tablica ukratko opisuje spomenute osnovne elemente:

naziv	opis	kategorija
title	ime dano resursu (najčešće od strane creatora ili publishera)	(1)
creator	(ili autor) označava osobu ili organizaciju koja je primarno odgovorna za nastanak resursa	(2)
subject	tema resursa; tipično subject se opisuje riječima ili frazama koje opisuju resurs	(1)
description	kratak opis sadržaja resursa	(1)
publisher	osoba ili organizacija koja je odgovorna da se reusrs pojavio u trenutnoj formi	(2)
contributor	osoba koja je uz creatora odgovorna za nastanak resursa, a ipak posjeduje sekundarnu ulogu (npr. artist, fotograf i sl.)	(2)
date	povezano je s datumom kada je resurs postao dohvatljiv - format datuma	(3)

	temelji se na ISO 8601 standardu	
type	određuje tip resursa (*homepage*, knjiga, priča, pjesma i sl.)	(1)
format	obično opisuje format resursa (knjige i sl.)	(3)
identifier	niz znakova ili broj koji jedinstveno određuje resurs (URN, URI ili ISBN ili bilo koji treći standard)	(3)
source	identifikator drugog izvora na temlju kojeg je nastao trenutni resurs	(1)
language	jezik kojim je napisan resurs	(3)
relation	idntifikator veze s drugim resursima koji su povezani s dotičnim	(1)
covreage	vremenski prostor u kojem je valjan resurs	(1)
rigts	izjava o autorskim i vlasničkim pravima na resurs (može biti i identifikator servisa koji to pruža)	(2)

Tokom vremena rasla je upotreba DC-a tako da se proširivao s dodatnim kvantifikatorima odnosno elementima koji su profinjavalji neke osnovne elemente. Trenutno u upotrebi postoje tri *namespacea* koji se nalaze iza sljedećih adresa.

- `http://purl.org/dc/elements/1.1/` - osnovnih 15 DC elemenata
- `http://purl.org/dc/terms/` - svi dodatni kvantifikatori
- `http://purl.org/dc/dcmitype/` - vokabular tipova (datumi i sl.)

DC predstavlja najuspješniji i najrašireniji rječnik. Koliko je DC ušao u svakodnevnu upotrebu najbolje svjedoči i to da je opisan u RFC dokumentu 2413.

5.2 FOAF

Izgradnja semantičkog weba zahtjeva cijeli niz različitih rječnika koji će pokrivati razne apsekte koji se mogu pojavljivati u takvom webu. Iako je semantički web prvenstveno namjenjen strojnom procesuiranju (i razumjevanju) on nikako ne isključuje ljude jer ipak oni će ga koristiti (sve će u konačnici biti orijentirano na *people centric* aplikacije) i podaci o samim osobama će se morati procesuirati kao i bilo koji drugi podatak u takvom webu. Osobe kreiraju određene podatke, povezani su s njima tako da je za očekivati da postoji jedan rječnik koji opisuje takve slučajeve. Dosadašnju prazninu u tom području ispunjava *community driven* projekt FOAF.
FOAF je akronim za «Friend Of A Friend» i zajednički napor grupe ljudi u izgradnji vokabulara/rječnika za izražavanje metapodataka o ljudima, njihovim interesima, odnosima i aktivnostima u koje su uključeni. FOAF kao otvoreni projekt pokrenut je od strane Dan Brickleya i Libby Millera. FOAF je izgrađen na

temelju RDF-a i kao takav obuhvaća sve dobre strane koje RDF donosi a prvenstveno se odnosi na izražajni jezik i mogućnost kombiniranja s drugim rječnicima u cilju postizanja veće izražajnosti.

FOAF je zapisan pomoću XML-a i kao takav posjeduje odgovarajuću schemu i specifikaciju zajedno s pripadajućim *namespaceom* (imenikom) koji se nalazi iza URI-a `http://xmlns.com/foaf/0.1`. Uobičajno je da *namespace* počinje s `foaf:` prefiksom. FOAF schema sadržava 12 klasa i cijeli niz njihovih *property-a*. Osnovna klasa koju nasljeđuju sve ostale klase, bilo one koje opisuju stvarne stvari ili one koje opisuju softverske artifakte, je `foaf:Agent`, no sve navedne klase nisu nabacane odjednom na gomilu nego se grupiraju u jednu od pet kategorija. Kategorije su:

1. FOAF Basic – opisuje osnovne stvari o osobi (ime, *email*, *homepage* i sl.)
2. Personal Info – dodatni opisi koji su vremenom dodavani određenoj osobi (weblog, dnaChecksum ...)
3. Online Accounts/IM – online *accounti* i *instant messaging* id-ovi (za najpopularnije IM servise: Jabber, MSNChat, ICQ, AIM ...)
4. Projects and Groups – određuje projekte, organizaciju i grupe u kojoj se osoba nalazi
5. Documents and Images – dokumenti i slike povezane s osobom

Najzanimljivije su dvije klase `foaf:Person` i `foaf:Image` na čijim ćemo primjerima upotrebe prikazati sve specifičnosti i značajke FOAF rječnika. Osnovna ideja koja je pokrenula FOAF projekt je kako napraviti rječnik koji će igrati ulogu koja je na početku razvoja weba imala početna stranica osobe (*homepage*). Na svom *homepageu* ljudi su upisivali uobičajno svoje ime, prezime, email, područej interesa te objavljivali jednostavnu skeniranu sliku. Sada je umjesto cijele HTML stranice dovoljeno malo se poigrati s klasom `foaf:Person` i napraviti jednostavan XML dokument.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">

<foaf:Person>
   <foaf:name>Peter Parker</foaf:name>
   <foaf:gender>Male</foaf:gender>
   <foaf:title>Mr</foaf:title>
   <foaf:givenname>Peter</foaf:givenname>
   <foaf:family_name>Parker</foaf:family_name>
   <foaf:mbox rdf:resource="mailto:peter.parker@dailybugle.com"/>
   <foaf:homepage rdf:resource="http://www.peterparker.com"/>
   <foaf:weblog rdf:resource="http://www.peterparker.com/blog/"/>
 </foaf:Person>
```

Običnim rječnikom rečeno, u gornjem dokumentu stoji: «Postoji osoba koja se zove Peter Parker. Ta osoba je muško, njegovo ime je Peter, prezime je Parker, on ima e-mail adresu peter.parker@dailybugle.com, *homepage* te osobe je

http://www.peterparker.com i on posjeduje svoj blog na internet web adresi http://www.peterparker.com/blog/.

FOAF za jedinstveno određivanje nekog resursa, u ovom slučaju osobe, koristi koncept «*inverse functional property*» (IFT) posuđenog iz OWL-a. Po tom principu resurs je jedinstveno određen (identificran) jednim svojim svojstvom (*property*). Tako možemo reći da je osoba jedinstveno određena jednim od sljedećih svojstava `foaf:mbox, foaf:mbox_sha1sum` i/ili `foaf:homepage`. Može se postaviti pitanje zašto ne koristiti npr. `rdf:about` svojstvo unutar `foaf:Person` taga i tako jedinstveno odrediti osobu. Takav načnin izaziva cijeli niz moralnih i tehničkih problema u smislu tko ili što je sposobno dodijeljivati jedinstvenu osobu svakoj osobi i kako se obraniti da jedna osoba ne dobije isti URI kao neka druga. Korištenjem IFT ovakvi problemi su elegatno izbjegnuti. Naravno, opasno je (*spam*) davati nečiji email samo tako zato se uobičajno ne koristi property foaf:mbox u kojem je u čistom tekstu napisan email, nego se koristi foaf:mbox_sha1sum koji predstavlja SHA1 hash vrijednost email adrese. U slučaju da FOAF aplikacija u svom radu naiđe na dva različita FOAF XML dokumenta koja posjeduju isto inverzno funkcijsko svojstvo (IFT), onda spaja (*merge*) podatke o resursu (osobi) iz oba dokumenta u jedno. Primjer:

```
<foaf:Person>
   <foaf:name>Peter Parker</foaf:name>
   <foaf:mbox_sha1sum>03f63fa7f20bd82</foaf:mbox_sha1sum>

 </foaf:Person>

 <foaf:Person>
   <foaf:name>Spiderman</foaf:name>
   <foaf:mbox_sha1sum>03f63fa7f20bd82</foaf:mbox_sha1sum>
 </foaf:Person>
```

FOAF aplikacija iz prethodnih dokumenta vidi da postoji osoba koja se zove Petar Parker i ima ime Sipderman. Ovaj proces naziva se «*smushing*». Naravno, u gornjem slučaju moramo pretpostaviti da `foaf:mbox` odnsono e-mail adresa jedinstveno identificira tu osobu tj. samo je njoj pridružena i nikome više.

Moguće je označiti odnose jednog FOAF identiteta (foaf:Person) s nekim drugim identitetom. Za tu je namjenu kreiran *property* `foaf:knows` kojim se izjavljuje da jedna osoba «poznaje» drugu. Na žalost, nije specifirano kakva je taj odnos poznavanja i da li je recipročan tj. da li poznavana osoba poznaje originatora.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

 <foaf:Person rdf:nodeID="harry">
   <foaf:name>Harry Osborn</foaf:name>
   <rdfs:seeAlso rdf:resource="http://www.osborn.com/harry.rdf"/>
 </foaf:Person>
```

```
 <foaf:Person>
   <foaf:name>Peter Parker</foaf:name>

   <foaf:knows rdf:nodeID="harry"/>

   <foaf:knows>
       <foaf:Person>
         <foaf:name>Aunt May</foaf:name>
       </foaf:Person>
   </foaf:knows>
 </foaf:Person>

</rdf:RDF>
```

Da bi se gornji problemi oko `foaf:knows` svojstva razrješili, potrebno je koristiti neke druge rječnike koji detaljnije opisuju prirodu veze tako se onda mogu izgraditi detaljni odnosi u nekim kompleksnim strukturama poput velikih korporacija. Jedna od zanimljivih specifičnosti FOAF specifikacije jest i njegova upotreba digitalnih slika . Tako je moguće izjaviti:

```
<foaf:Person>
   ...
   <foaf:depicts rdf:resource="http://www.ppark.com/peter.jpg"/>
   ...
</foaf:Person>
```

Ovom izjavom kazuje se da spomentu osoba je prikazana na slici određenoj `rdf:resource` URI-ijem, no moguće je i inverzno napraviti koristeći `foaf:Image` klasu i unutar nje `foaf:depicts` svojstvo.

```
<foaf:Image rdf:about="http://www.ppark.com/photos/statue.jpg">
   <dc:title>Battle on the Statue Of Liberty</dc:title>

   <foaf:depicts rdf:resource="#spiderman"/>
   <foaf:depicts rdf:resource="#green-goblin"/>

   <foaf:maker rdf:resource="#peter"/>
 </foaf:Image>
```

FOAF je relativno mlada specifikacija koje vrlo dobro obrađuje područje za koje je namjenjena te se lako proširuje odrugim rječnicima. Ujedno je prisutno podosta gotovih aplikacija koje istu koriste što uključuje i FOAF explorer koji u HTML obliku prikazuje sadržaj FOAF dokumenata. Svi te dokumente nije teško napraviti i bez korištenja posebnih alata (običnim editorom teksta).

6 OWL

6.1 Uvod

Treća osnovna komponenta semantičkog weba je ontologija. Pojam ontologija (grč. *ontos* = ono što jest biće + *logos* = riječ, istina) dobro je poznat filozofima koji ga definiraju kao filozofsku disciplinu o biću kao biću tj. istražuju ono po čemu su bića - bića. U svijetu računala, ontologija je dobila novo modificirano značanje. Ona predstavlja jezike koji se koriste za definiranje vokabulara pojmova između kojih su jednoznačno definirane relacije u tom specifičnom kontekstu. Jezici ontologija služe za zapisivanje istih. Uobičajna je podijela ontologija prema raznim domenama kao što su:

- medicina: UMLS, SNOMED, Galen
- povijesti: AAT, ULAN
- STEP aplikacijski protokol,

i prema općenitosti:

- top-level kategorije
- jedinice i dimenzije

Kao što je vidljivo, prije pojave semantičkog weba, svaka zajednica (organizacija) koja je trebala neku ontologiju, stvarala je vlastitu(e) bez ikakve brige za međuoperabilnosti s drugim zajednicama i njihovim računalnim sustavima. Semantički web je unio potrebu za razvojem ontološkog jezika koji će nadići te probleme interoperabilnosti. Prije samog opisa OWL, možemo postaviti pitanje zašto razvijati i specificirati novi jezik, ako je sve već moguće napraviti s postojećim standardiziranim jezikom jezikom kao što su prije opisani RDF i RDF Schema. Ugrubo, RDF je ograničen na binarne predikatem dok je RDF Schema, isto tako ugrubo, ograničena na hijerarhiju klasa i svojstava s defincijama ranga i domene th svojstva. Na žalost RDF i RDF Scema pate od slabe ekspresivnosti te je moguće vrlo lako i brzo pronaći niz ontoloških *use-case-ova* koje nije moguće zapisati tim jezicima.

Uvidjevši sve te nedostatke, brojne istraživačke grupe u Americi i Europi počele su rad na pronalaženju novog ontološkog jezika. Kako se ne bi bespotrebno duplicirao rad na istoj temi, došlo je do zajedničke udružene inicijative nazvane DAML+OIL (ime je dobiveno iz američkog projekta DAML-ONT i europskog OIL). W3C Web Ontology Gruop, kao krovna organizacija u tim pitanjima, preuzela je zajedničku inicijativu kao početnu točku razvoja OWL-a koji će postati širko prihvaćeni i standardizirani ontološki jezik za semantički web. OWL nije jezik

nastao nidokud, nego je jezik koji se temelji na prijašnjim radovima u stvaranju semantičkog weba, odnosno izgrađen je na temeljima RDF-a i RDF Scheme.

6.2 Zahtjevi

Ontološki jezici trebaju dopustiti korisnicima jednostavno pisanje eksplicitne, formalno konceptualizirane modelne domene. Da bi to uspješno napravili moraju zadovoljavti sljedeće uvjete, koji su istovremeno postavljeni i pred OWL kao jeziku ontologija:

1. moraju posjedovati dobro definiranu sintaksu
2. moraju posjedovati dobro definiranu semantiku
3. moraju posjedovati učinkovitu podršku odlučivanju (*reasoning support*)
4. moraju posjedovati dovoljno ekspresivne snage za opis pojmova

Određene stvari iz zadanog popisa su same po sebi jasne. To se posebno odnosi na zahtjev da jezik posjeduje dobro definiranu sintaksu (*well-defined syntax*). Zapis OWL-a napravljen je u XML-u koji već sam po sebi zadovoljava taj uvjet. Moguće je prigovoriti da XML nije *user friendly* zapis, no kako se uobičajno s XML-om radi pomoću grafičkih alata taj problem otpada. Formalna semantika opisuje precizno značenje znanja. Precizno označava da to znanje nije podložno nikakvoj subjektivnoj procijeni i izmjeni odnosno interpretaciji. Važnost formalne semantike posebno je naglašena u domeni matematičke logike (a i drugdje, iako nije na prvi pogled vidljivo). Upotreba formalne semantike je preduvjet za podršku odlučivanju.

Kao što je napisano, RDF/RDF Schema posjeduju određene onotloške kvalitete, no mnogobrojne mogućnosti im nedostaju. Najvažnije od tih mogućnosti su :

- lokalni doseg svojstava (rdfs:range)
- disjunkciju klasa : katkada želimo imati klase koje su disjunktivne (primjer muško/žensko), no to nije moguće izraziti
- Boolean kombinaciju klasa
- kardinalna restriktivnost
- posebne karakteristike *propertiya* kao što su tranzitivnost, jedinstvenost i inverzija

6.3 OWL jezici

Idealno bi bilo kada bi se OWL mogao izgraditi kao ekstenzija na RDF Schemu (koja je pak ekstenzija na RDF) u smislu da bi se koristila RDF značenja klasa i *propertya* (`rdfs:Class`, `rdfs:subClassOf`). Izrada takve ekstenzije dovela bi do podosta problema u nekontroliranom računanju *propetiya*. Da bi se to izbjeglo,

načinjen je određeni trade-off između ekspresivnosti jezika i učinkovitog zaključivanja. Tako su definirana tri OWL jezika:

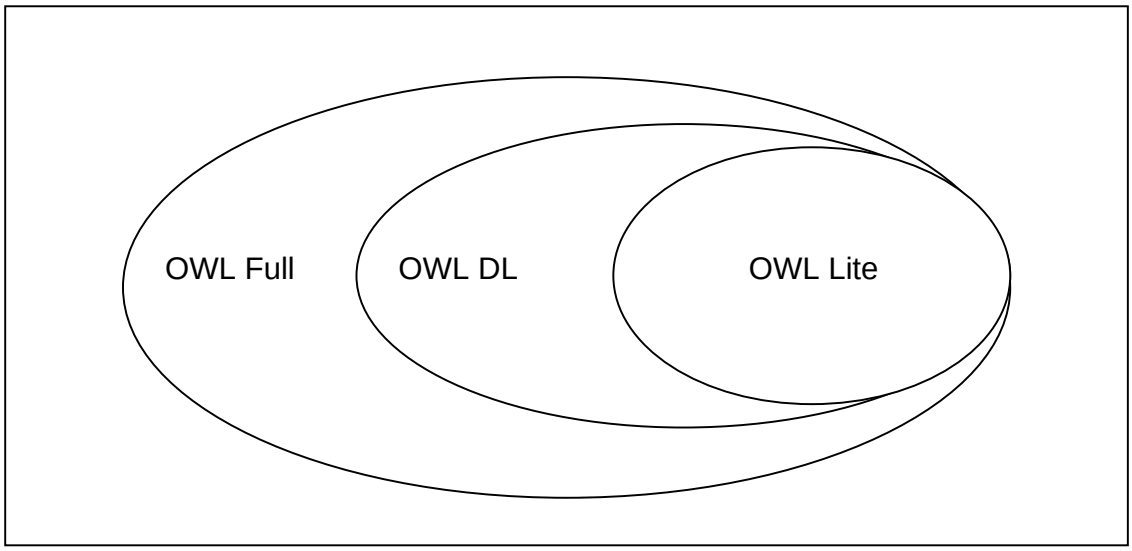
(a) **OWL Full** - je «puni» OWL jezik sa svim mogućnostima. On koristi sve OWL primitive koje je moguće proizvoljno kombinirati s RDF i RDF Schema primitivima. Najveća prednost OWL Full jezika je njegova kompatibilnost s RDF porodicom (svaki OWL Full zaključak je ujedno i RDF zaključak). Najveći nedostatak je u njegovoj kompleksnosti koja katkad može dovesti do nemogućnosti odlučivanja.

(b) **OWL DL** (Description Logic) - predstavlja restrikciju nad OWL Full jezikom jer se je željela postići računalna učinkovitost. Restrikcija je napravljena nad OWL i RDF konstuktorima i njihovoj upotrebi. Najveći nedostatak ovog podjezika je njegova nekompatibilnost s RDF-om.

(c) **OWL Lite** predstavlja daljnu restrikciju nad OWL DL jezikom i to nad jezičnim konsturktorima. Njegova najveća prednost je u tome što je on najlakši za implementiranje i razumjevanje.

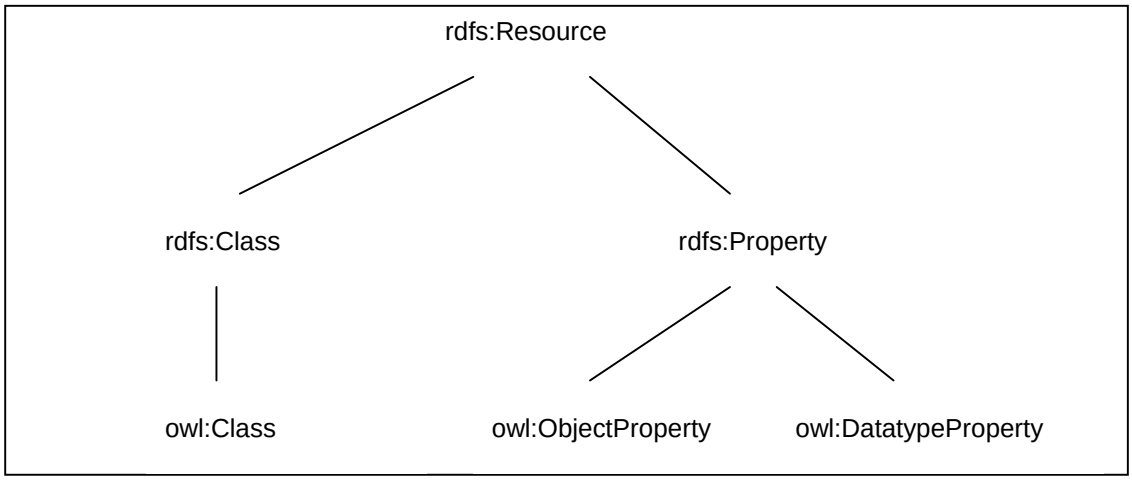
Developeri koji upotrebljavaju OWL trebaju iz tri ponuđena podjezika izabrati onaj koji najbolje odgovara njihovim potrebama. Naravno pri tome postoji kompatibilnost na gore u smislu da je :

- svaka OWL Lite ontologija ujedno i OWL DL ontologija
- svaka OWL DL ontologija ujedno i OWL Full ontologija
- svaki OWL Lite zaključak ujedno i OWL DL zaključak
- svaki OWL DL zaključak ujedno i OWL Full zaključak



slika 10.
Odnos između OWL jezika

Originalna želja autora OWL-a je bila ta da se postigne kompatibilnost s postojećim standardom RDF-om, no zbog spomenutog *trade-off-a* ona je postignuta samo za OWL Full jezik. Odnos RDF Scheme i OWL Full jezika (njihovih primitiva) pirkazan je na donjem grafu:



slika 11.
Odnos primitiva RDF scheme i OWL Full primitiva

OWL je moguće prikazati na više načina:

1. poznatom RDF XML sintaksom za koju se smatra da je primarna sintaksa iako nije pretjerano čitljiva
2. XML baziranom sintaksom koja ne slijedi RDF konvenciju (ova sintaksa je čitljivija od prethodne)
3. apstraktna sintaksa koja se koristi u jezičnoj specifikaciji dokumenata (ona je kompaktnija i čitljivija od prethodnih)
4. grafička sintaksa bazirana na UML-u (Universal Modelling Language). UML točnije njegov Class dijagram predstavlja dobar izbor za sve koji su otprije upoznati s UML te žele koristiti OWL

6.4 OWL zaglavlja

OWL dokumenti se obično nazivaju i OWL ontologije i kako je već spomenuto, oni su i ujedno i RDF dokumenti. Osnovni element/tag u RDF dokumentima je rdf:RDF. On je i početni element OWL dokumenta i također specifira niz imenika (namespaces). OWL dokument započinje i s nizom izjava (*asssertions*) koje uglavnom služe za održavanje dokumenta. Te izjave su grupirane unutar `owl:Ontology` elementa koji sadrži komentare, oznaku verzije i uključivanje drugih ontologija.

```
<owl:Ontology rdf:about=»»>
   <rdfs:comment>primjer OWL dokumenta</rdfs:comment>
   <owl:priorVersion rdf:resource=»http://www.primer.com/prim1»/>
```

```
    <owl:imports rdf:resource=»http://www.dom-primjer.com/onto2»/>
</owl:Ontology>
```

Od svih spomenutih elementa jedino `owl:imports` element ima posljedice na logičko značenje ontologije. On «uključuje» druge ontologije, odnosno za njihov sadržaj se podrazumjeva da je i sadržaj dotične ontologije. `owl:imports` je tranzitivan: ako ontologija A uključuje ontologiju B, i ontologija B uključuje ontologiju C, onda ontologija A uključuje ontologiju C.

Važan dio zaglavlja su i informacija o verziji ontologije odnosno njegovoj kompatibilnosti s prethodnim verzijama ontologije. Za to služe elementi:

- `owl:versionInfo` – općenita informacija o verziji (bilo brojčana, bilo znakovna)
- `owl:backwardCompatibleWith` – ova izjava definira prethodnu verziju s kojom je ontologija kompatibilna
- `owl:incompatibleWith` – označava da nije kompatibilna s nekom prethodnom verzijom

6.5 OWL klase

Svaka klasa se definira korištenjem `owl:Class` elementa. OWL specifikacija podrazumjeva postojanje dvije specijalne klase:

- `owl:Thing` - najopćenitija klasa (sve je stvar) - sve klase su podklase ove klase
- `owl:Nothing` - prazna klasa; svaka klasa je nadklasa (*superclass*) ove klase

Pri definiranju proizvoljne klase moguće je koristiti dodatne elemente kojima se pobliže određuje klasa. Oni su:

- `owl:disjointWith -` definira disjunkciju dviju klasa
- `owl:equivalentClass` - definira ekvivalenciju dviju klase

6.6 Svojstva elemenata

OWL razlikuje dva tipa svojstva:

- svojstvo objekata
- svojstvo tipa podataka (*datatype*)

OWL ne posjeduje predefinirane tipove podataka (cijelobrojni, float, niz znakova, ...), niti pruža specifičan način njihovog definiranja, već dopušta korištenje

predefiniranih tipova iz XML Scheme odnosno njegovih tipova podataka. Na ovakav način postiže se tražena slojevitost arhitektrue gjde «viši» slojevi koriste svojstva «nižih» slojeva. OWL podržava definiranje korisničkih tipova podataka koji se onda uobičajno skupljaju u XML Schemu te se zatim koriste u ontologiji.

Jedna od posebnosti OWL su podržane restrikcije nad svojstima. Tako je pretpostavimo da jedna klasa K je ujedno podklasa klase K'. Sve to podrazmjeva da je svaka instanca klase K ujedno i instanca klase K'. No ako želimo, možemo ograničiti da instance klase K moraju zadovoljiti neke uvjete da bi uopće pripadali toj klasi pri čemu klasu K' označavamo kao anonimnom. To se postiže uvođenjem elemanta `owl:Restriction` koji se obavezno sastoji od elemenata :

- `owl:onProperty` – svojstvo koje se ograničava
- jedan od elemenata koje definiraju ograničavanje.

Elementi koji definiraju ograničavanje mogu biti :

- `owl:hasValue` - posjeduje neku određenu vrijednost
- `owl:allValuesFrom` - posjeduje sve vrijednosti
- `owl:someValuesFrom` - posjeduje neke vrijednosti
- `owl:minCardinality` i/ili `owl:maxCardinality (owl:cardinality)`

Primjer u kojem se ograničava dob osobe na 18 godina:

```
<owl:Class rdf:about=»#nesto»>
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource=»#age» />
        <owl:mainCardinality
            rdf:datatype=»&xsd;nonNegativeInteger»>
          18
        </owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

OWL specifikacija definira neka specijalna svojstva te boolean kombinacije. Posebna svojstva se tiču definiranja tranizitivnosti, simetričnosti funkcijskih svojstava te njihovih inverzija dok boolean kombinacije dopuštaju unije (`owl:unionOf`), njihove presjeke (`owl:intersectionOf`) i komplemente nad svojstima (`owl:complementOf`).

6.7 Enumeracija

Enumeracija u OWL-u je definirana `owl:oneOf` elementom koji se koristi pri deiniranju klase kao niz (listing) svojih elementa.

```
<owl:oneOf rdf:parseType=»Collection»>đ
  <owl:Thing rdf:about=»Ponedjeljak» />
  <owl:Thing rdf:about=»Utorak» />
  ...
  <owl:Thing rdf:about=»Nedjelja» />
</owl:oneOf>
```

7 Upotreba semantičkog weba

Semantički web , već i prije svoje pune zrelosti naišao je na niz primjena u poslovnom svijetu. Tu primjenu je posebno ubrzao ogromni rast World Wide Web-a (WWW). Rast podatka (brojnost novih dokumenata, multimedijalnog sadržaja) u webu donio je nezapamćene probleme oko pronalask informacija, njihovog sortiranja, kategoriziranja i prezentiranja. Iako su ti problemi eksponencijalno manji u zatvorenim WWW okolinama, kao što je intranet ili ekstranet određene kompanije, oni su i dalje vrlo veliki te se sa «starim» metodama i protokolima ne mogu dobro razriješiti. Semantički web djeluje kao vrlo obećavajući projekt u područjima Knowledge management-a (upravljanje znanjem). U sljedećim poglavljima bit će opisane dvije primjene semantičkog weba u *knowledge management* području i to generičku infrastrukturu razvijenu u Ontonknowledge projektu te drugi projekt zvan SWAP koji upotrebljava tehnolofije korištene u Peer-to-peer (P2P) mrežama.

7.1 Upravljanje znanjem i Ontoknowledge projekt

Učinkovito upravljanje znanjem je identificirano kao osnovni element u održanju kompetitivnosti neke tvrtke. Tradicionali *knowledge management* sustvi se trenutno suočavaju s novim problemima koje su potaknuti spomenutom napretkom weba: informacijsko preopterećeje, neučinkovito pretraživanje po ključnim riječima, heterogene informacijkse strukture i geografska dislociranost tvrtkinih podružnica. Ontoknowledge projekt (dalje OTK) je pokušao razriještii navedene probleme tako da svoj trud koncentrira na sadržajno orijentirano upravljanje znanjem kroz rastuće ontologije.

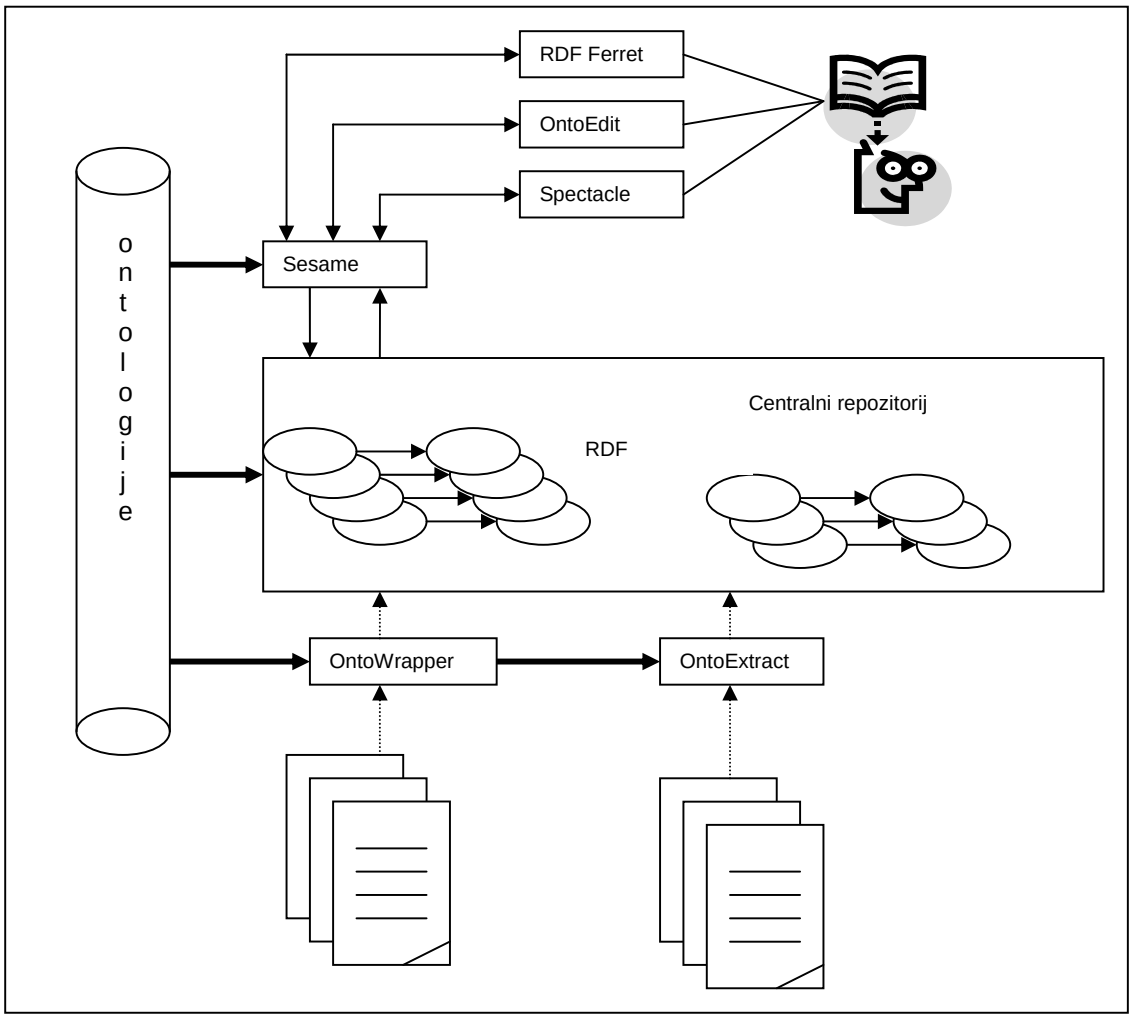
OTK podržava učinkovito i efektivno upravljanje znanjem tako što pruža alate te se fokusira na dohvat, održavanje i pristup informacijskim izvorima koji su slabo strukturirani.

- dohvat (acquiring) podataka: podaci se dohvaćaju iz različitih izvore te se onda iz njih vadi potreban teskt. OTK pruža dva alata za ovu operaciju: OntoExtract i OntoWrapper.

- održavaje (*maintaning*): u održavanju podataka korsite se uobičajni jezici semantičkog weba, a oni su XML, RDF i ontologije (OIL jedan od prethodnika OWL-a). Ti jezici se koriste za opis sintakse i semantike dobivenih podataka. OTK dolazi s alatom za uređivanje ontologije (OntoEdit) te alatom za spremanje i dohvat ontologija (Sesame). Ujedno se održavanje ontologija odvija automatski, a ne samo pri prvom dolasku podatka.

- pristupanje (*accessing*): *push* usluge i tehnologija agenata daje podršku krajnjim korisnicima u pristupu podacima. OTK dolazi s dva alata za tu namjenu. Prvi je navigacijski alat temeljen na ontologijskim informacija koji ujedno omogućuje upite, dok je drugi alat za vizualizaciju (Spectacle).

U biti, u prethodnim opisanim točkama dobiva se opća slike rada OTK-a, gdje se prvo dohvaćaju podaci, zatim se obrađuju i spremaju u repozitoriji dok se na kraju od strane korisnika dohvaćaju preko raznih upita.



slika 12.
Ontoknowledge projekt - sastavni elementi

Sesame predstavlja jedan od najznimljivijih komponenti jer se preko njega šalju upiti i dobivanju podaci iz repozitorija. Njegova funkcionalnost se sastoji od:

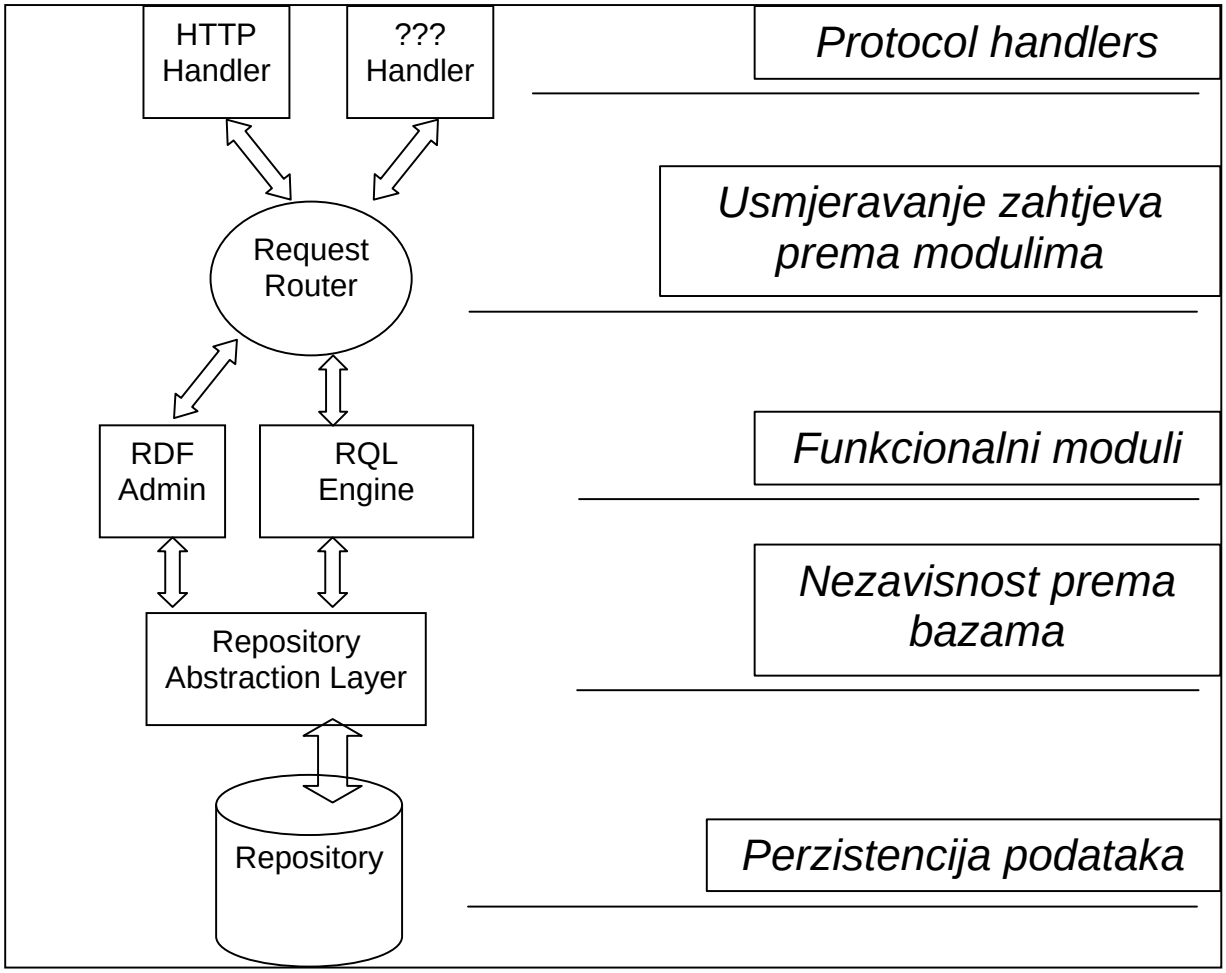
- područja za smještaj RDF podataka (stoarage)
- *query engine* za RDF Query Language (RQL) - upitni jezik za RDF
- *upload* podataka i *download* u RDF obliku
- komunikacija preko raznih protokola (HTTP je podrazmjevani protokol)

Sesame je definiran kao skalabilna komponenta, koji je nazavisan od tipa repozitorija (baza podataka, memorija, datotečnii sustav ...). Sesame posjeduje modularni dizajn tako da je lako dodavati nove module bez narušavanja funkcionalnosti postojećih.

Na **RQL** (upitni jezik) može se gledati kao na SQL, ali on obavlja upite nad RDF izjavama, a ne tablicama u relacijskoj bazi podataka. On je temljen na OQL-u (Object Query Language) koji predstavlja upitni jezik za objektni model i razvijen je unutar OMG-a (Object Management Group). OQL predstavlja standardizirani jezik u svojem području. RQL podržava standardni set osnovnih upita naslijeđen od OQL_a (Class, Property, subClassO ...), razne filtere, boolean izraze te funkcionalnu kompoziciju upita.

Prijmeri RQL upita:

upit	**RQL**
Osnovni upiti	
daj mi sve instance klase Student	`Student`
daj mi sve podklase klase student	`subClassOf(Student)`
Napredni upiti	
daj mi sve studente čije prezime počinje s P	`select R from Student{R}.last_name{N} where N like “P*”`
daj mi sva svojstva koja neki student može imati zajedno s njihovom domenom i rangom	`select @P, domain(@P), range(@P) from {:Researcher} @P {}`



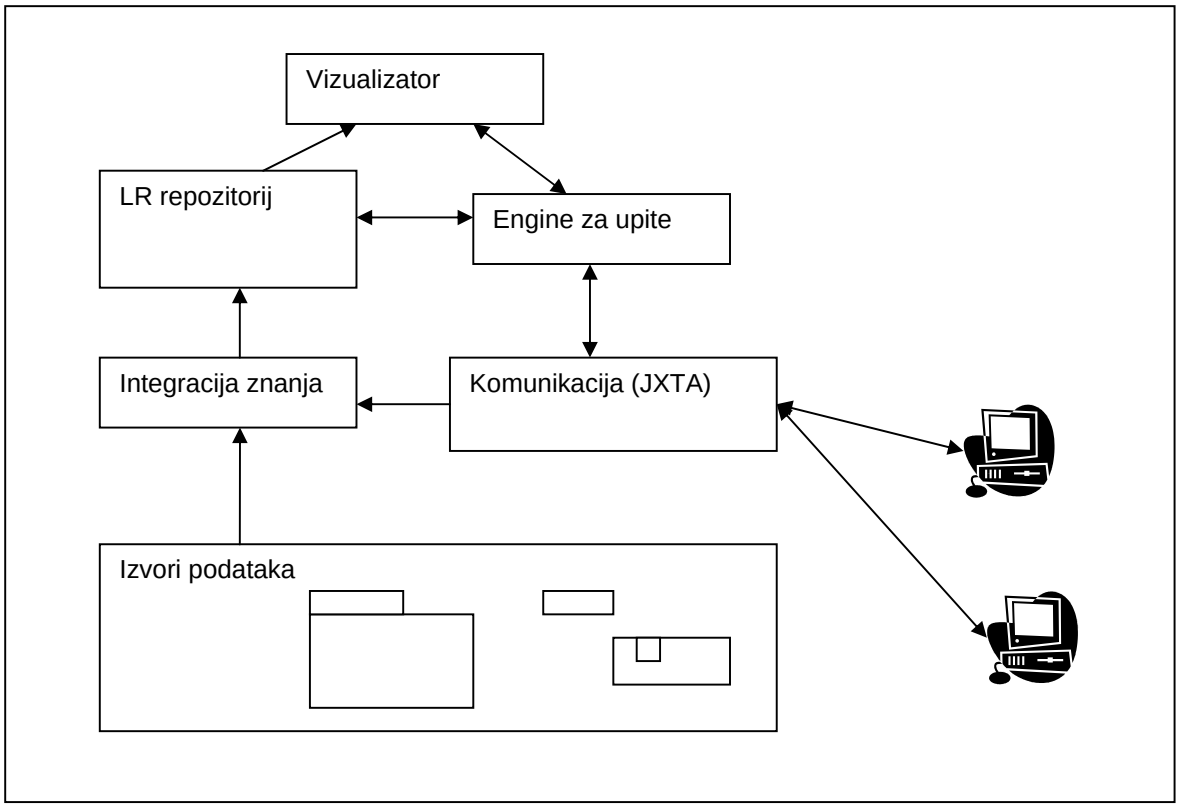
slika 13.
Ontoknowledge projekt - Seasame - sastavni elementi

7.2 P2P knowledge management

Glavni nedostatak centraliziranih *knowledge management* sustava je spori razvoji ontologija. Taj nedostatak moguće je zaobići izradom «laganih» ontologija i primjenom arhitekture temeljene na peer-to-peer (P2P) mrežama. SWAP projekt demonstrira takav način kreiranja i upravljanja znanjem. On je pokrenut od strane Europske unije i predstavlja ogledni primjer u svojoj kategoriji.

SWAP (Semantic Web And P2P) sustav se sastoji od takozvanih SWAP čvorova (*nodes*). Znanje iz pojedinog *peera* je ekstrahirano iz nekoliko izvora znanja (knowledge source) i onda integrirano te spremljeno u LR (*Local Node Repository*). Za pretraživanje/editiranje i davanje upita brine se posebno korisničko sučelje. Pojedni upiti se mogu izvršiti na lokalnom čvoru ili mogu biti propagirani kroz cijli sustav (mrežu). Posebna komponenta se brine o dohvatu takvih rezultata, njihovom prepisivanju i propagiranju na čvorove koji bi mogli dati najbolje odgovore.

Svaki učesnik SWAP sustava sam određuje što će od lokalnih resursa uključiti kao izvor podataka za zajedničko znanje. Nakon tog odabira, lokalni *source* se integrira u LR čiji su unosi napravljeni u RDF-u. Svaki od tih unosa dobiva i vlastiti ID i lokaciju gdje se nalazi (vazano uz ime čvora). LR predstavlja integrirani pogled nad cijelim poznatim znanjem bilo da je iz lokanih izvora ili dobiven iz mreže odnosno od drugih čvorova. Svaka izmjena u izvorištima podataka je propagirana u LR. LR je moguće preko vizualnih alata editirati, no to je preporučeno samo iskusnim osobama. Izmjene u LR se automatski događaju kada se dobiju rezultati od drugih čvorova. Pogledi (*views*) predstavljaju predefinirane prespektive nad znanjem. Mogu se generirati iz selektiranih izvora ili nad cijelim LR-om. Oni su implementirani korištenjem različitih vizualizacijskih tehnika te se mogu lako pregledavati (*browser*). Kao što je napisano, svaki se upit može propagirati nad cijelom mrežom, odnosno nad čvorovima koji su poznati čvoru koji šalje upit. Drugi čvorovi mogu dalje proslijediti upit i vratiti rezultate sve do izvoršnog čvora. Unošenje upita moguće je obaviti vizualno ili manualno kao tekst, a svaki taj upit je translatiran u RDF *query language*. Jedna od karakteristika SWAP-a je ta da za svaki dobiveni odogovor od drugog čvora je moguće zatražiti dokument koji je zapisan u tim odgovorima. Ta opcija podrazumjevano uključuje i sposobnosti *file sharinga*, no to je uglavnom poznato svim P2P mrežama. SWAP je implementiran u JXTA (www.jxta.org) .



slika 14.
Elementi SWAP projekta

8 Zaključak

Trenutni web je došao do svojih granica i potreban mu je podstrek za daljnji razvoj. Ogromna količina sadržaja je nesortirana i cjelokupno znanje koje se u njima nalazi je zapravo sakriveno. Semantički web ima namjeru srediti stanje te dopustiti aplikacijama, a ne samo ljudima, da aktivno sudjeluju u skupljanju, obradi i publiciranju sadržaja.

Odlika semantičkog weba je izgradnja nove generacije (treće) na temelju postignutih tekovina (XML, URI, HTTP) uz korištenje tehnologija koje se nadograđuju na temelje. Nove tehnologije (RDF(S), OWL) su slojevito napravljene tako da svaka od navedenih tehnologija ima specifičnu ulogu te koristi standarde/funkcionalnost komponenti «ispod» sebe. Na taj način se postiže jednostavnost dizajna i implementacije novog weba. Spomenuti standardi su jedan dio implementacije semantičkog weba. Drugi dio predstavlja izgradnja raznih onotologija. Posebno poglavlje poredstavlja nedostatak rječnika. Na to područje lako bi mogla uskočiti *open source* zajednica kao što je učinila s FOAF projektom.

Brzina uvođenja semantičkog weba nije velika jer za njegovu potpunu implementaciju i iskorištavanje potrebno je dosta mjenjati način generiranja sadržaja što nije jednostavan proces. Tako da u prvim koracima se semantički web uvodi i koristi unutar velikih korporacija koje svoj intra/ekstranet obogaćuju semantikom što je posebno vidljivo u korištenju internih tražilica i sustavima za upravljanje znanjem. Jedno od velikih područja gdje je bi upotreba semantičkog weba mogla napraviti revoluciju je e-commerce temeljen na peer-to-peer mrežama.

Semantički web trpi dosta kritika od osoba koji su se puno prije počeli baviti predstavljanjem znanja u informacijskim sustavima. Kritike se tiču osnovnih tehnologija semantičkog weba jer one nemaju punu funkcionalnost rada sa znanjem (izražajnosti, zaključivanja) kao što posjeduju prethodno razvijene tehnologije/metodologije. Te kritike nisu promašene, no da bi se nešto uspjelo napraviti na ovakom webu, mora se krenuti s manjim zahtjevima te onda iterativno dodavati «stara znanja». Rad na semantičkom webu nije dovršen te radne grupe unutar W3C nastavljaju s unaprijeđenjem, poboljšanjem i razvojem (novih) standarda. Na drugoj strani se odvija spora, ali postojana implementacija semantičkog weba u postojeći sustav te izrada pogodnih alata (Protege) i sustava za brži razvoj.

Literatura

[1] Thomas Krichel, «The Semantic Web and an Introduction to RDF_», 8. 8. 2002

[2] Berners-Lee, Tim, «Semantic Web Road map», http://www.w3.org/DesignIssues/Semantic, 1998.

[3] Berners-Lee, Tim, «Web Architecture from 50,000 feet», http://www.w3.org/DesignIssues/Architecture, 1999.

[4] R.Guha, R.McCool1 i R.Fikes, «Contexts for the Semantic Web», Knowledge Systems Lab., Stanford University, USA, 2004.

[5] M. Ehrig, C. Tempich, S. Staab, J. Broekstra, F. V. Harmelen, M. Sabou, R. Siebes i H.Stuckenschmidt, «SWAP: Oontology-based knowledge management with peer-to-peer technology», 2. Konferenz Professionelles Wissensmanagement, Lucern, April 2-4, 2003.

[6] RDF site summary.http://www.purl.org/rss/1.0/.

[7] D. Brickley and R.V.Guha. Rdf schema. http://www.w3.org/TR/rdf-schema/.

[8] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. Van Harmelen, M. Klein, S. Staab, and R. Studer, «OIL: The Ontology Inference Layer, Technical Report», University of Manchester / Vrije Universiteit Amsterdam. http://www.ontoknowledge.org/oil/

[9] Jeen Broekstra, Michel Klein, Stefan Decker, Dieter Fensel i Ian Horrocks4, «Extending RDFS», www.ontoknowledge.org/oil/extending-rdfs.pdf, 4. 9. 2000.

[10] Stefan Decker, Frank van Harmelen, Jeen Broekstra Michael Erdmann, Dieter Fensel, Ian Horrocks, Michel Klein, Sergey Melnik, «The Semantic Web - on the respective Roles of XML and RDF», www.ontoknowledge.org/oil/downl/IEEE00.pdf, 2000.

[11] H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn (eds.), «XML Schema Part 1: Structures W3C Working Draft», http://www.w3.org/TR/2000/WD-xmlschema-1-20000407/, W3C 7. 4. 2000,

[12] Grigoris Antoniou and Frank van Harmelen, «Web Ontology Language: OWL», http://www.cs.vu.nl/~frankh/postscript/OntoHandbook03OWL.pdf, 2003.

[13] D. McGuinness and F van Harmelen, «OWL Web Ontology Language Overview», http://www.w3.org/TR/2003/WD-owl-features-20030331/, W3C 2003.

[14] P. Patel-Schneider, P. Hayes, I. Horrocks, «OWL Web Ontology Language Semantics and Abstract Syntax», http://www.w3.org/TR/2003/WD-owl-semantics-20030331/, W3C 2003.

[15] J. He_in, «Web Ontology Language (OWL) Use Cases and Requirements» http://www.w3.org/TR/2003/WD-webont-req-20030331/, W3C 2003.

[16] D. Fensel, «OIL: An Ontology Infrastructure for the Semantic Web» IEEE Intelligent Systems 16,2 (2001).

[17] Sebastian Schaffert, «RDF and RDF Schema: An Overview», Oberseminar Knowledge Markup, 26.07.2001

[18] Tim Berners-Lee, James Hendler and Ora Lassila, «The Semantic Web», Scientific American: The Semantic Web, 17. 5. 2001.