Fakultet Elektrotehnike i Računarstva
ZEMRIS

SEMANTIČKI WEB

Gospodnetić Luka
0036375942

Sadržaj

1 Uvod

World Wide Web (WWW) sadrži ogromne količine informacija stvorenih od različitih organizacija, zajednica te pojedinaca iz raznih razloga. Korisnici Web-a mogu vrlo lako pristupiti tim informacijama specificirajući URI adresu, pretraživanjem te slijeđenjem linkova pristupati ostalim povezanim resursima. Jednostavnost uporabe je glavna značajka zašto je Web tako popularan, što više život je postao nezamisliv bez njega.

Jednostavnost uporabe trenutnog Web-a ima svoju negativnu stranu, lagano se izgubiti ili otkrivati nevažne i nepovezane informacije. Time se naša pretraživanja svode na dug i mukotrpan posao, jer ne trošimo vrijeme samo tražeći informacije koje nas zanimaju već «gubimo» vrijeme selektirajući korisne informacije od nepovezanih i nevažnih. Kako bi ilustrirali ovaj problem, pretpostavimo da tražimo nešto vrlo jednostavno, tipa «John Doe». Kao rezultat pretraživanja dobit ćemo vrlo mnogo informacija, počevši od telefonskih imenika koji sadrže ime «John» te «Doe», vrsta dionica (bezvrijednih), pa do informacijama o nestalim osobama u Americi.

Vrlo je česta pojava prilikom pretraživanja, ukoliko naše pretraživanje ima više značenja, dobiti mnogo različitih informacija. Ta višeznačnost se može odnositi i na nacionalne jezike a i na pojedini jezik. Npr., ako se gleda na razini nacionalnih jezika, riječ «pile», koja u hrvatskom jeziku označava životinju, u engleskom označava riječ «hrpa». Ako se gleda značenje unutar jedno nacionalnog jezika, npr. hrvatskog, riječ «led» se može odnositi na smrznutu tekućinu ali također i na Led diodu.

Sa ubrzanim razvojem World Wide Web-a, količina dostupnih informacija na mreži se povečava eksponencijalno. Nedostatak standardizacije te zajedničkog rječnika i dalje stvara heterogenost, koja sprečava razmjenu informacija i komunikacija.

Cilj Semantičkog Web-a je stvaranje standarda i tehnologija koji podržavaju razvoj, dizajniranih da pomognu strojevima da razumiju više informacija o Web-u, tako da mogu rezultirati sa bogatijim rezultatima pretraživanja, podatkovnom integracijom, navigacijom te automatizacijom zadataka. Sa Semantičkim Web-om ne dobivamo samo točnije rezultate kad koristimo pretraživače, već također znamo kad možemo povezati informacije sa različitih izvora, znamo koje informacije se mogu uspoređivati te možemo pružiti razne automatizirane usluge u različitim domenama, počevši od budućeg doma, digitalnih knjižnica pa sve do elektronskih poslova i zdravstvenih servisa.

Sa Semantičkim Web-om možemo asocirati bilo koji resurs sa semantički bogatom opisujućom informacijom, tj. svaki resurs će se opisati s informacijama. Npr., dodavajući meta podatke (podaci o podacima) o stvaranju nekog dokumenta, možemo pretraživati dokumente koji imaju meta podatke koji npr. specificira John Doea kao pisca nekog dokumenta. Sa malo više meta podataka možemo također pretraživati samo dokumente koji spadaju u kategoriju «znanstveni seminari». U dosadašnjem Web-u mi smo imali samo URI dokumenta i ništa više, a na Semantičkom Web-u nama su pruženi koncepti dokumenata, ljudi i veze između njih i dokumenta. Npr., davajući unificirane identifikatore osobi, uloga «pisac» te koncept «znanstveni seminar», vrlo jasno prikazujemo da znamo ko je ta osoba, te da znamo pripadajuće veze između te osobe i odgovarajućeg dokumenta. Što više, specificirajući o kojoj osobi govorimo, možemo razlikovati preobilje dobivenih «John Doeova». Također možemo kombinirati informacije sa različitih mjesta i saznati više o toj osobi u različitim kontekstima, kao pisac, menađer itd.

Semantički Web je vizija, to je ideja da se informacije na Web-u definiraju i povezuju na način da mogu biti korištene, sa strane strojeva, ne samo za prikazivanje, već i za automatizaciju, integraciju te ponovnu upotrebu informacija duž kojekakvih aplikacija.

Ontologije obuhvaćaju semantiku informacije te se dohvaćaju s raznih mjesta. One daju uniformni, deklarativni i jasni opis pa su dobile bitnu ulogu zbog zahtjeva akademija i industrija. Kako se broj različitih ontologija povećava, zadatak održavanja i reorganiziranja ontologija sa svrhom da se omogući ponovna uporaba znanja je zahtjevna. Prodor u ontološkoj tehnologije zahtjeva metodološku pomoć i alate koju omogućuju efikasnu i efektivnu izradu (development). Ključni aspekt, u ostvarivanju toga, je uspješna ponovna uporaba ontologija. Ontološki knjižnični sistemi (ontology library systems) su važni alati u grupiranju i i reorganiziranju ontologija za ponovnu uporabu, integraciju, održavanje, mapiranje i verziranje.

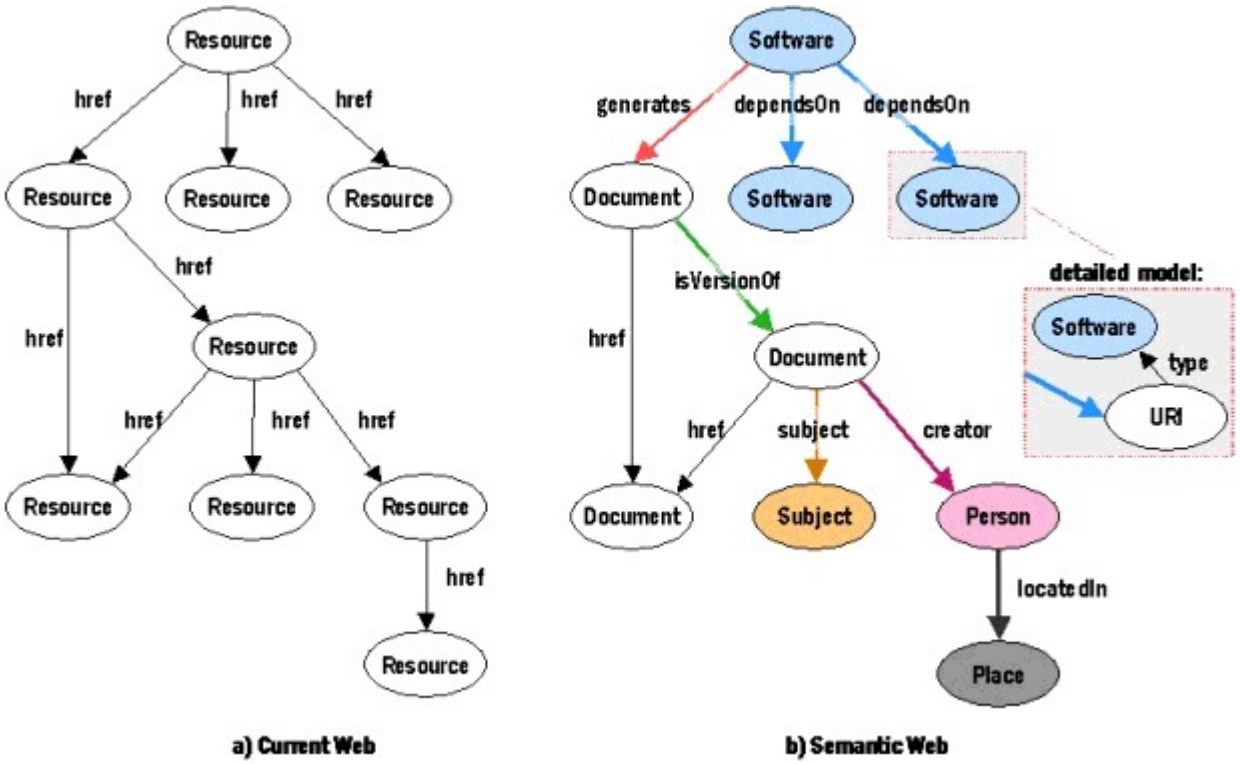
2 Glavni principi Semantičkog Web-a

2.1 Sve može biti identificirano URI-ovim

Ljudi, mjesta i stvari u fizičkom svijetu mogu biti u Semantičkom Web-u referirana koristeći niz identifikatora. Onaj tko ima kontrolu na dijelu prostora imena na Web-u (Web namespace) može kreirati URI i reći da on identificira nešto u fizičkom svijetu. Neki ljudi su zahtjevali da samo mali dio Web URI prostora imena bude dostupan za tu namjenu. Semantički Web ne potražuje, ni ne podržava takva ograničenja. Možemo se također fizičkim entitetima obraćati indirektno. Npr., osobi s imenom «John Doe» možemo se obraćati, poznavajući URI njegovog e-mail sandučića, u rečenici tipa: «Osoba čija je e-mail adresa glasi «mailto:john_doe©tenkre.com» i čije je ime John Doe.». Možemo nastaviti s specificiranjem puno više stvari o toj osobi bez da ikad dodamo novi identifikator za tu osobu.

2.2 Resursi i linkovi mogu imati tipove

Trenutačni Web sastoji se od resursa i linkova (slika 1 lijevo). Resursi su Web dokumenti namijenjeni za ljudsku uporabu i obično ne sadrže meta podatke koji opisuje čemu služe i koje su njihove relacije prema ostalim Web dokumentima. Dok čovjek može shvatiti lako da je jedan resurs pismo, drugi novela a treći znanstveni rad, za strojeve je ova informacija često nedostupna. Slično korisnik može nagađati koje veze ima resurs čitajući ga, dok je stroju puno teže doći do istih rezultata. Više informativne veze bi bile npr. «ovisi o», «je verzija od», «ima temu», «autor».



Slika 1, Razlike trenutačnog i Semantičkog Web-a

Semantički Web se također sastoji od resursa i linkova (slika 1 desno). Međutim, sada resursi i linkovi mogu imati tipove koji definiraju koncept koji daje više informacija stroju. Npr., neki likovi mogu reći da je resurs verzija nekog drugog resura, da je napisan od resursa koji opisuje osobu, ili da resurs sadrži programsku podršku koji ovisi na nekim drugim programskim podrškama.

2.3 Tolerira se parcijalna informacija

Trenutačni Web je neograničen tj. žrtvuje integritet linka za mogućnost brzog i nenajavljenog rasta. Autori mogu lako povezati vlastiti resurs sa drugim, bez brige o povezivanju prema vlastitom resursu. Bez načina da se informira «linkera» kad se resurs premjestio (pomaknuo) mi prihvaćamo mogućnost da dobijemo 404 link koji nas informira da link više ne vodi na neki Web resurs.

Slično je i Semantički Web neograničen, bilo tko može reći bilo što o bilo čemu i stvarati različite tipove ili linkove između resursa. Uvijek će postojati nešto više za otkrivanje. Neki od povezanih resursa mogu prestati postojati ili adrese mogu biti ponovno iskorištene za nešto drugo. Alati na Semantičkom Web-u trebaju tolerirati taj raspad podataka (data decay) i biti u mogućnosti funkcionirati unatoč tom raspadu.

2.4 Nema potrebe za apsolutnom istinom

Sve što je pronađeno na Web-u nije nužno istinito i Semantički Web ne mijenja to na bilo koji način. Vjerodostojnost ili istina se evaluira sa u svakoj aplikaciji koja procesira informaciju na Web-u. Te aplikacije odlučuju u što da vjeruju koristeći kontekst izjave (tko je rekao što, kad te koje su kredencije imali za to izreći)

2.5 Podržana je evolucija

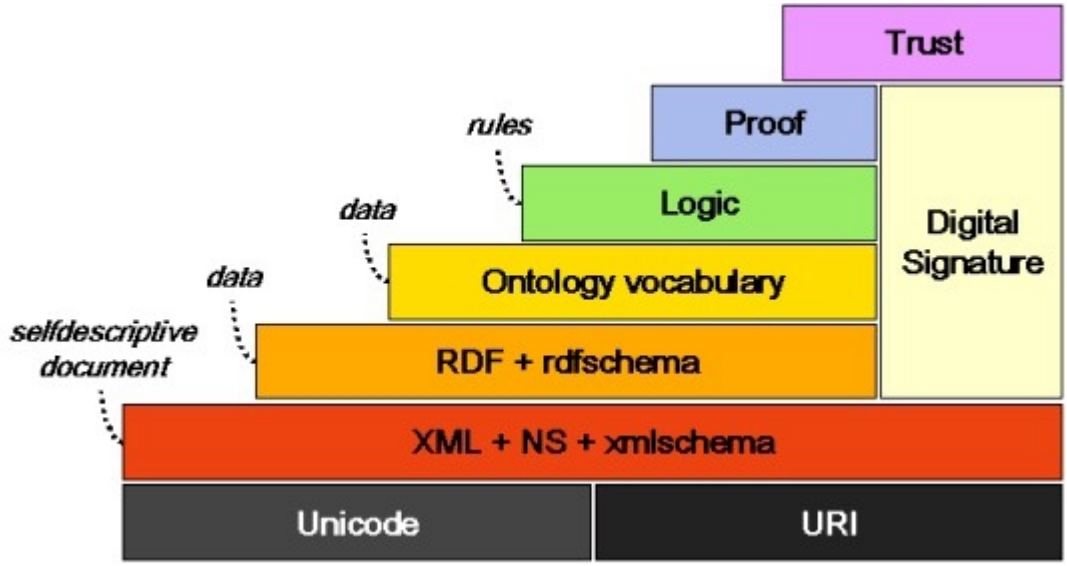
Normalno je da su koncepti slične tematike često definirani različitim grupama ljudi na različitim mjestima na mreži ili istom grupom ljudi ali u različita vremena. Često bi bilo korisno kombinirati dostupnost podataka na Web-u koji koristi te koncepte. Semantički Web koristi opisne konvencije koje se mogu širiti kao što se i ljudsko razumijevanje ekspandira. Štoviše, konvencije dopuštaju efektivne kombinacije nezavisnih poslova raznolikih zajednica čak i kad koriste različite rječnike.

2.6 Minimalistički dizajn

Semantički Web čini jednostavne stvari jednostavnim, a složene stvari čini mogućim. Cilj W3C Activity je da se standardizira samo ono nužno. Ovaj pristup omogućuje implementaciju jednostavnih aplikacija sad kad se baziraju na već standardiziranim tehnologijam.

3 Slojevi Semantičkog Web-a

Principi Semantičkog Web-a su primjenjeni na slojeve Web tehnologija i standarda. Slojevi su prikazani na slici. Unicode i URI slojevi služe da nam omoguće korištenje internacionalnog skupa znakova (character set) i pruže način za identificiranje objekata u Semantičkom Web-u. XML sloj sa prostorom imena i shemom definicija osiguravaju integraciju definicije Semantičkog Web-a sa ostalim standardima baziranim na XML-u tj osigurava serijalizaciju. Sa RDF-om (RDF) te RDF Shemom (RDFS) je moguće napraviti izjave o objektima sa URI-ovima i definirati rječnike kojima se može baratati preko URI-ova. Ovo je sloj u kojem možemo dati tipove resursima i linkovima. Ontološki sloj podržava evoluciju rječnika, također može definirati relacije između različitih koncepata. Navedeni slojevi, zajedno sa slojem za detekciju promjena kod dokumenata (Digital Signature layer), trenutno se standardiziraju u radnoj grupi W3C.



Slika 2, Slojevi Semantičkog Web-a

Slojevi na vrhu: *Logic*, *Proof*, *Trust* se trenutno razvijaju te se grade jednostavni demonstrativni primjeri. Logički sloj omogućava pisanje pravila, dok *Proof* sloj izvršava (executes) ta pravila i evaluira, zajedno sa *Trust* slojem, da li da se vjeruje danom dokazu (proof) ili ne.

4 W3C Semantic Web Activity

W3C Semantic Web Activity grupa je osnovana da bi glumila ulogu vođe u dizajniranju specifikacija te u izradi otvorenih, kolaborativnih tehnologija. Ona je nasljednik W3C Metadata Activity. Sljedeći zadaci su joj dodijeljeni:

1. *Uvođenje standarda*
2. *Edukacija*
3. *Suradnja i koordinacija*
4. *Napredni razvoj*

Cilj W3C Semantic Web Activity je standardiziranje ključnih tehnologija koje omogućuju necentralizirani razvoj a da pri tom paze da svi dijelovi odgovaraju jedan drugom. Glavni fokus im je informacija koja može biti konzumirana i shvaćena od strane strojeva. Trenutačno Semantic Web Activity ima dvije radne podgrupe koje definiraju standarde i tehnologije: RDF Core Working Group (RDFCore) te Web Ontology Working Group (WebOnt).

Trenutačni Resource Description Framework (RDF) standard (1999) pruža obično okruženje (framework) koji reprezentira meta podatke u mnogim aplikacijama. RDF Core Working Group je namijenjena da završi RDF opisni rječnik u RDF Schema Candidate Recommendation-u (RDFS 2000). Da bi postigla svoj cilj RDF Core Working Group izdala je niz dokumenata:

1. *RDF/XML Syntax Specification*
2. *Resource Description Framework Concepts and Abstract Data Model*
3. *RDF Vocabulary Description Language 1.0: RDF Schema*
4. *RDF Primer*
5. *RDF Model Theory*
6. *RDF Test Cases*

Web Ontology Working Group ima zadatak da na temelju RDF Core-a stvori jezik za definiranje strukturiranih ontologija baziranih na Web-u. Taj jezik će omogućiti bogatiju integraciju te interoperabilnost podataka između raznih zajednica. Da bi postigli svoj cilj, radna grupa je izdala je niz dokumenata:

1. *Requirements for a Web Ontology Language*
2. *Feature Synopsis for OWL Lite and OWL*
3. *OWL Web Ontology Language 1.0 Abstract Syntax*
4. *OWL Web Ontology Language 1.0 Reference*
5. *Web Ontology Language (OWL) Test Cases*

5 Ontologija

Filozofska definicija ontologije je učenje o općim, fundamentalnim i konstitutivnim određenjima bitka. Ontologije se često vidi kao osnovni građevni blokovi Semantičkog Web-a. Ontologije sadrže ponovno iskoristive dijelove znanja o specifičnim domenama. Međutim, ti dijelovi informacija nisu često statički, već evoluiraju tijekom vremena. Promjene na domenama, prilagodba različitim poslovima ili promjene u konceptualizaciji zahtijevaju modifikacije ontologija. Iz toga zaključujemo da je potrebna podrška za podržavanje tih promjena. To je jako važno u decentraliziranom i nekontroliranom okružju kao što je Web, gdje se promjene događaju bez obavijesti. Promjene u nekontroliranim okruženjima mogu imati veće posljedice na nepredvidivost rezultata naspram kontroliranim okruženjima. Rastom Semantičkog Web-a, te nekontrolirane promjene, će imati još veći utjecaj, jer će kompjuteri koristiti podatke. Nema više ljudi u lancu koji korištenjem ogromnog iskustva te metodama rješavanja problema induktivnim razmišljanjem, mogu uočiti pogrešne kombinacije koje su posljedice nepredvidivih promjena.

Problem podrške za podržavanjem promjena je jako velik, jer postoje velike zavisnosti između podataka, aplikacija i ontologija. Promjene pri kraju će imati dalekosežne posljedice. U praksi je često vrlo tesško uskladiti promjene na ontologiji sa modifikacijom aplikacije i podataka koji ju koriste.

Ontologije mogu povećati funkcionalnost Web-a na mnogo načina. Mogu biti iskorištene na jednostavan način, poboljšavanje preciznosti u pretraživanju Web-a – pretraživački programi mogu tražiti samo one stranice koje se odnose na određeni koncept umjesto da pronalaze stranice koje koriste jednu od neodređenih ključnih riječi. Naprednije aplikacije će koristiti ontologije za povezivanje informacija na stranicama sa strukturama znanja i pravilima zaključivanja.

5.1 Problem verziranja ontologija

Prije nego počnemo istraživati rješenja za ontološko verziranje, prvo moramo detaljnije pogledati što je to uopće. Općenito gledanom ontološko verziranje samo znači da okolo postoje višestruke varijacije ontologije. U praksi, te varijacije često proizlaze iz promjena na već postojećoj varijaciji ontologije, pa prema tome grade nasljedno (derivirajuće) stablo. Isto je moguće da razne verzije ontologija su nezavisno stvorene i nemaju međusobno nasljednu vezu. U našem pogledu, ontološko verziranje će se blisko povezivati sa promjenama u ontologiji.

Definirat ćemo ontološko verziranje kao *sposobnost obrađivanja promjena u ontologijama tako da stvaramo i upravljamo novom varijacijom te ontologije*. Da bi to uspjeli napraviti, treba nam metodologija sa metodama za razlikovanje i prepoznavanje verzija, te sa procedurama za moderniziranje i promjene u ontologijama. To implicira da trebamo pratiti relacijske veze između verzija.

Možemo reći da *metodologija verziranja* pruža mehanizam koji onemogućuje korisnicima ontoloških varijacija dvosmislene interpretacije koncepata, te omogućuje kompatibilnost varijacija. Stupanj promjene određuje kompatibilnost između varijacija. To implicira da treba proučiti semantičke posljedice promjena. Metodologija verziranja odgovara na pitanje: «Kako ponovno iskoristiti već postojeće ontologije u novim situacijama, bez onemogućavanja besprijekorne trenutne uporabe?»

5.1.1 RAZLOZI ONTOLOŠKIH PROMJENA

Metodologija verziranja za ontologije hvata se u koštac sa promjenama u ontologijama. Da bi proučili razloge promjena, moramo proučiti prirodu ontologija. Prema *Gruber*-u [25] (1993), promjene u ontologijama su izazvane:

1) *Promjenama u domeni*
2) *Promjenama u podijeljenoj konceptualizaciji*
3) *Promjenama u specifikaciji*

Prvi tip promjene se često događa. Taj problem je često znan iz područja verziranja shema baza podataka. *Ventrone* i *Heiler* [22] (1991) su skicirali sedam različitih situacija u kojima promjena u domeni zahtjeva promjenu u modelu baze podataka. Primjer toga je spajanje dva odjela na fakultetu, ta promjena u stvarnom svijetu zahtjeva da se ontologija koja opisuje tu domenu također promjeni.

Promjene u podijeljenoj konceptualizaciji se također često događaju. Važno je znati da podijeljena konceptualizacija domene nije statička specifikacija koja se stvori jednom u povijesti. *Fensel* [23] (2001) opisuje ontologije kao dinamičke mreže sa značenjem, u kojima se koncenzus postiže u socijalnom procesu razmjena informacija i značenja. Ovaj pogled daje dvostruku ulogu ontologija u razmjeni informacija: pružaju koncenzus koji je ujedno i preduvjet za izmjenu informacija i rezultat tog procesa razmjene.

Konceptualizacija se može također promijeniti zbog perspektive na uporabu. Različiti poslovi mogu zahtijevati drugačija gledišta na domenu i kao posljedicu imati različitu konceptualizaciju. Kad se ontologija prilagodi na novi zadatak ili na novu domenu modifikacija reprezentira promjene u konceptualizaciji. Npr., ontologija prijevoznih povezanosti nekog grada, sa konceptima kao ceste, biciklističke staze, kanali, mostovi itd. Kad se ontologija prilagodi sa biciklističke perspektive na perspektivu prijevoza po vodi, konceptualizacija mosta se mijenja sa načina prijelaza preko kanala na vremenski okupacionu zapreku.

Promjene ontologija mogu biti izvedene na razne načine:

- Prethodna verzija ontologije se zamjeni novo verzijom bez bilo kakve (formalne) notifikacije
- Nova verzija ontologije je dostupna a prethodna je zamijenjena novom
- I nova i prethodna verzija ontologije su dostupne
- I nova i prethodna verzija su dostupne, te postoji eksplicitna specifikacija relacije između koncepta nove i prethodne verzije

5.1.2 POSLJEDICE ONTOLOŠKIH PROMJENA

Podrška za verziranje je neophodna jer promjene u ontologijama mogu izazvati nekompatibilnosti koje znače da promijenjena ontologija ne može samo tako biti zamijenjena sa nepromijenjenom verzijom. Postoji mnogo detalja koji ovise o ontologiji. Svaka od tih ovisnosti može uzrokovati drugačiji tip nekompatibilnosti.

1) Postoje podatci koji se prilagođavaju ontologiji. U Semantičkom Web-u to mogu biti Web stranice na kojima je sadržaj komentiran sa terminima iz ontologije. Kad je ontologija promijenjena, ti podaci mogu dobiti drugačiju interpretaciju ili mogu koristiti nepoznate termine

2) Ostale ontologije koje koriste promijenjene ontologije. To mogu biti ontologije koje su stvorene iz izvorne ontologije ili koje unose ontologiju. Promjene u izvornoj ontologiji mogu utjecati na rezultirajuče ontologije.

3) Aplikacije koje koriste ontologije mogu isto biti utjecane promjenama na ontologijama. U idealnom slučaju, konceptualno znanje koje je potrebno za aplikaciju trebalo bi samo biti specificirano u ontologiji, međutim, u praksi aplikacija također koristi interni model. Taj interni model može postati nekompatibilan sa ontologijom.

Mijenjanje ontologija bez ikakvih notifikacija može rezultirati s ispravnom interpretacijom podataka. To je slučaj kad modifikacija ontologije ne utječe na postojeće definicije. Također se vidi da bi bilo korisno imati pristup starijim verzijama ontologija. To dopušta usporedbu definicija i ocjenjivanje ispravnosti definicija korištenih s drugom verzijom podataka.

Sveukupno razmatrano, zaključujemo da je metodologija verziranja neophodna za brigu oko sljedećih relacija:

1) *između sukcesivnih revizija jedne ontologije*
2) *između ontologije i :*
   a. *podatka*
   b. *povezanih ontologija*
   c. *povezanih aplikacija*

Sad kad smo razjasnili probleme oko verziranja ontologija, možemo formulirati zahtjeve i želje za metodologiju verziranja. Prvo ćemo definirati generalni cilj verziranja.

Metodologija verziranja trebala bi pružiti mehanizme i tehnike baratanja promjena na ontologijama, pazeći pri tom na postizanje maksimalne interoperabilnosti sa postojećim podacima i aplikacijama. To znači da bi trebalo zadržati što više moguće informacija i znanja, bez izvođenja pogrešnih informacija.

Nameću se sljedeći zahtjevi na verziranje, povećavajući pri tom nivo težine:

- za svaku uporabu koncepta ili relacije, verziranje bi trebalo pružiti točno određenu referencu na namijenjenu definiciju – identification

- verziranje bi trebalo napraviti eksplicitnu relaciju između raznih verzija koncepata ili relacija – change specification
- trebalo bi, koliko god je to moguće, automatski prevoditi i povezivati verzije i podatke, da bi se omogućio otvoren i jasan pristup – transparent evolution

5.1.3 IDENTIFIKACIJA ONTOLOGIJA

Prvo pitanje na koje treba odgovoriti, kad želimo identificirati verziju ontologije na Web-u je: «Što je identificiranje ontologije?». To nije tako trivijalno kako izgleda. U ovom vrlo maglovitom području mi ćemo zauzeti sljedeće stajalište. Pretpostavljamo da je ontologija predstavljena, reprezentirana datotekom na Web-u. Svaka promjena koja rezultira drugačijom znakovnom reprezentacijom ontologije se smatra revizijom. U slučaju da logičke definicije nisu promijenjene, odgovornost ostaje na autoru revizije da odluči da li je ta revizija semantička promjena pa prema tome formulira novu konceptualizaciju sa svojim novim identitetom ili je samo promjena reprezentacije iste konceptualizacije.

Drugo pitanje je: «Kako identifikacija na Web-u utječe na Web resurse i njihove identitete?». To nas dovodi na jako sklisku debatu o značenju URI-ova, URN-ova i resursa. Glavno pitanje u ovoj diskusiji je: «Što je resurs i kako ćemo ga identificirati?». Međutim, mi ćemo izbjeći ova pitanja i pristupiti problemu iz drugog smjera: «Jesu li entiteti u našoj domeni resursi i možemo li ih identificirati?». Na ova pitanja relativno je jednostavno odgovoriti. Prema definiciji URI-a (Uniform Resource Identifiers), definirano u *Berners-Lee*–u [25] (1998), «resurs može biti sve što ima identitet» a prema *Berners-Lee-u* [24] (1996) «resurs je konceptualno entitet». Pa prema tome ontologija se može bezbrižno zvati i resursom. Prema njima se URI (kompaktni string znakova za identificiranje apstraktnih ili fizičkih resursa) može koristiti za identificiranje resursa. Primijetite da URI pruža mehanizam za generalnu identifikaciju, nasuprot tome URL (Uniform Resource Locators) je ograđen sa lokacijom resursa.

Važan korak u ovoj predloženoj metodi je kompletna separacija identiteta ontologije od identiteta datoteka na Web-u koji specificiraju ontologiju. U drugim riječima klasa ontoloških resursa treba se razlikovati od klasa resursa. Svaka revizija je nova resursna datoteka i dobiva novi identifikator, ali to ne znači da automatski se dobiva i novi ontološki identifikator.

Kad uzmemo u obzir sva ova razmatranja, predlažemo metodu identifikacije koja se temelji na sljedećim točkama:

- razlika između tri klase resursa:
    1. datoteke
    2. ontologije
    3. linije unazad kompatibilnih ontologija
- promjena u datoteci rezultira novim identifikatorom datoteke
- korištenje URL-a za identifikaciju datoteka
- samo promjena konceptualizacije rezultira novim identifikatorom ontologija
- novi tip URI-a za identifikaciju ontologije sa dvorazinskom numeracijskom shemom:
    1. manji brojevi za unazad kompatibilne modifikacije (ontološki-URI koji završava sa manjim brojem identificira specifičnu ontologiju)
    2. veći brojevi za nekompatibilne promjene (ontološki-URI koji završava sa većim brojem identificira liniju unazad kompatibilne ontologije)

- individualni koncepti ili relacije, čiji se identifikatori razlikuju samo u manjem broju, se smatraju jednakim

Ideja ovih navedenih točaka je sljedeća. Koristeći novu vrstu URI-a. moguće je zapisati sve informacije koje su potrebne za korištenje, i također onemogućiti konfuziju sa URL-ovima koji specificiraju lokaciju.

Činjenica da se individualni koncepti sa identifikatorima, koji se samo razlikuju u manjem broju smatraju ekvivalentnim, je potrebno da bi se omogućilo kompatibilnost unazad. (backward compatibility). Po početnim pretpostavkama, svi resursi na Web-u sa različitim identifikatorima se smatraju različitim. Ova izjava omogućuje kreiranja samostojećih revizija ontologija, koje imaju koncepte koji su jednaki onim u prethodnim verzijama.

U ovom dijelu smo istražili problem verziranja u kontekstu Web-a. Diskutirali smo prirodu ontoloških promjena i promatrali posljedice. Naš glavni cilj je postizanje maksimalne iskoristivosti već stečenog znanja. To implicira da nije dovoljno saznati da li je specifična interepretacija ontologije na podatku ispravna, već pokušavamo izderivirati, izvući, što više korektnih informacija.

Materijal opisan u ovom seminaru je još uvijek u procesu rasta. Još je puno stvari koje još nisu završene. Smatra se da je najvažnija mana nedostatak detaljne analize efekta specifične promjene na interpretaciju podatka..

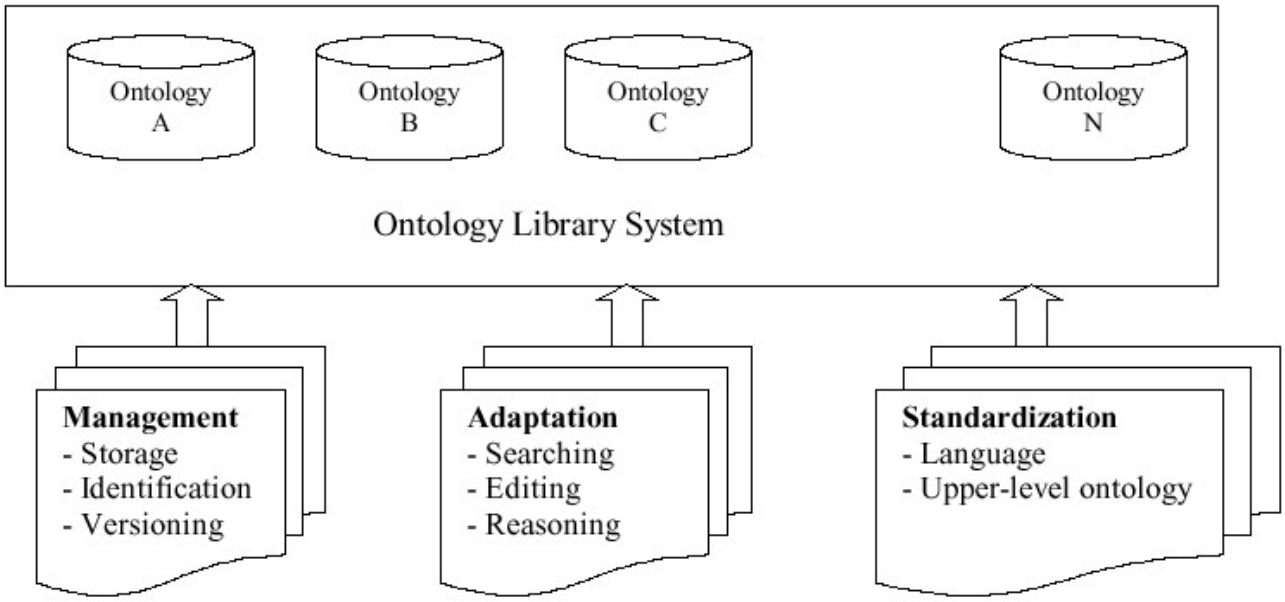
5.2 Ontology Library System

Ontološki knjižnični sistem je knjižnični sistem koji omogućuje razne funkcije za upravljanje, prilagođavanje i standardizaciju grupa ontologija. Ovaj sistem bi trebao ispuniti potrebe za ponovnu uporabu ontologija. U tom smislu, ontološki knjižnični sistem bi trebao biti lako pristupačan i pružati efikasnu potporu za ponovnu uporabu već postojećih bitnih ontologija i standardizirat ih po «upper-level» ontologijama i ontološki reprezentativnim jezicima.

«Upper-level» ontologija obuhvaća i modelira osnovne koncepte i znanja koja bi mogla biti ponovno iskorištena u kreiranju novih ontologija i u organiziranju ontoloških knjižnica. Prisutna je u većini ontoloških knjižnica. Npr., u ontološkoj knjižnici DAML te u Ontološkom serveru (Vrije Universiteit, Brussels), se NE nalazi, a u nekima (WebOnto, SHOE) imaju svoje bazične ontologije. Moglo bi se reći da je «upper-level» ontologija u većini slučajeva roditeljska ontologija. Većina se ontologija poziva (extends) na «upper-level» ontologiju. One su jako bitne za bolju organizaciju ontoloških knjižničnih sistema.

Ovdje ćemo identificirati glavni kriterij za evaluaciju funkcionalnosti ontološkog knjižničnog sustava. Također ćemo pažljivo promotriti postojeće prijedloge u skladu s tim kriterijima. S namjerom da omogućimo ponovnu upotrebu ontologija, knjižnični sistem mora barem zadovoljiti sljedeće kriterije (slika 3):

- *ponovnu uporabu ontologije sa identifikacijom i verizacijom*
- *ponovnu uporabu ontologije omogućavajući besprijekoran pristup postojećim ontologijama i pružajući naprednu podršku u adaptiranju ontologija na određenu domenu (umjesto da zahtijevaju da se takva ontologija napiše od početka (from scratch)*
- *ponovnu uporabu ontologije tako da se iskoriste prednosti standardizacije*



Slika 3, Općeniti pregled strukture

5.2.1 UPRAVLJANJE

Najvažnija funkcija ontoloških knjižničnih sistema je omogućavanje ponovnog korištenja znanja (ontologija). Važan aspekt funkcionalnosti ponovne uporabe ontološkog knjižničnog sistema su otvoreno skladištenje (*open storage*), identifikacija i podrška verziranja.

- *Skladištenje* (kako uskladištiti ontologiju):
  1. Da li je ontologija pristupačna udaljenom pristupom i uređivanju?
  2. Da li su ontologije klasificirane po nekom postojećem ili «homemade» kategorijom
  3. Da li su ontologije pohranjene u modulima?

- *Identifikacija* (kako jedinstveno identificirati ontologiju):
  1. Svaka ontologija mora imati jedinstveni identifikator u ontološkom knjižničnom sistemu

- *Verziranje* (kako održavati evidenciju promjena na ontologijama):
  1. Verizacija je jako kritična u osiguravanju konzistentnosti među različitim verzijama ontologija

5.2.2 ADAPTACIJA

Ontološki knjižnični sistem bi trebao omogućiti povećanje i moderniziranje ontologija. Trebao bi omogućiti *pristupaćno* okruženje za traženje i editiranje ontologija. Važan aspekt u ontološkom knjižničnom sistemu je objedinjavanje podrške u nalaženju i modificiranju postojećih ontologija.

- *Pretraživanje* (kako pretraživati ontološki knjižnični sustav?)

• *Editiranje* (kako dodavati, brisati i uređivati specifične ontologije u ontološkom knjižničnom sistemu?)

1. Jedan od najvažnijih mogućnosti koje ima ontološki knjižnični sistem bi trebala biti ona koja modificira spremljene ontologije ili dodaje nove ontologije.
2. Kako sistem podržava funkcije editiranja?
3. Da li podržava *remote* editiranje?

• *Rasuđivanje* (Reasoning) (Kako izvesti posljedice iz ontologije?)

1. Kako sistem podržava evaluaciju i verifikaciju ontologija?
2. Da li podržava izvođenje bilo kojeg pitanje-odgovor zadatka?

5.2.3 STANDARDIZACIJA

Ontološki knjižnični sistem bi trebao slijediti postojeće i dostupne standarde, kao što su ontološki reprezentacijski jezici, standardizacija taksionomija ili strukture ontologija.

• *Jezik* (vrsta jezika korištena u ontološkom knjižničnom sistemu npr. RDF, XML, DAML+OIL)

1) Da li sistem samo podržava jedan standardni jezik ili druge razne jezike?

• *Upper-level ontology* (Da li je ontološki knjižnični sistem «prizemljen» u bilo kojoj postojećoj *upper-level* ontologiji?)

1) *Upper-level ontology* obuhvaća i modelira temeljni koncept i znanje koje bi moglo biti ponovno iskorišteno u stvaranju novih ontologija i u organizaciji ontoloških knjižnica

5.2.4 NAJPOPULARNIJI ONTOLOŠKI KNJIŽNIČNI SISTEMI

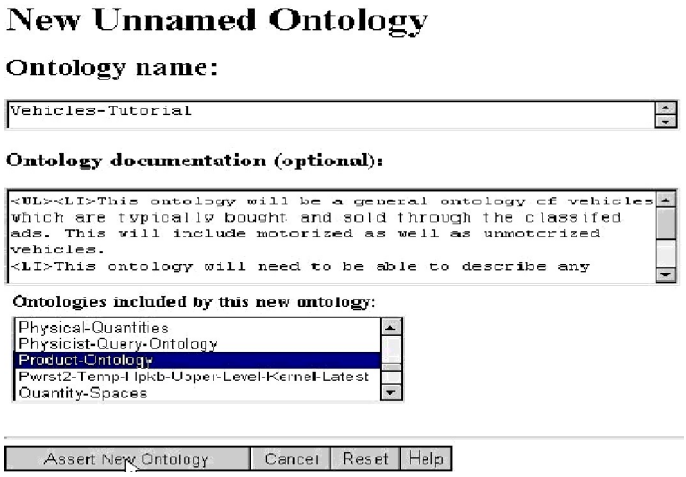
Ova poglavlje daje pregled trenutno važnih ontoloških knjižničnih sistema. To uključuje *WebOnto* (Knowledge Media Institute, Open Univeristy, UK), *Ontolingua* (Knowledge Systems Laboratory, Stanford University, USA), *DAML ontološki knjižnični sistem* (DAML, DAPAR, USA), SHOE (Univeristy of Maryland, USA), *Ontology Server* (Vrije Universiteit, Brussels, Belgium), IEEE Standard Upper Ontology (IEEE), *OntoServer* (AIFB, University of Karlshruhe, Germany) i *ONIONS* (Biomedical Technologies Institute (ITBM) of the Italian National Research Council (CNR), Italy). *ONIONS* je metodologija za ontološku integraciju i bio je vrlo uspješno implementiran u nekoliko medicinskih ontoloških knjižničnih sistema. Striktno govoreći, to nije ontološki knjižnični sistem. Međutim, kako on definira razne kriterije za razvoj takvih sistema, mi ćemo ga na kratko promotriti ovdje. Postoji jako puno ontoloških knjižničnih sistema. Tu su navedena samo ona o kojima su javno dostupna dovoljno dobra i detaljna informacija da bi mogli proučavati i uspoređivati njihove aktualne funkcionalnosti.

1 *WebOnto*

WebOnto je ontološki knjižnični sistem napravljen na Knowledge Media Institute Otvorenog Sveučilišta u UK. Dizajniran je da podržava kolaborativno stvaranje, pretraživanje i editiranje ontologija. Pruža direktnu manipulaciju okruženja prikaza ontoloških izraza i također pruža ontološki alat za diskutiranje *Tadzebao*, koji podržava ujedno i asinkrono i sinkrono diskutiranje o ontolgijama.

2 *Ontolingua*

Otolingua je napravljen u ranim devedesetim na Knowledge Systems Laboratory of Stanford University. Sastoji se od poslužitelja i reprezentativnog jezika. Server pruža repozitorij ontologija koji pomaže korisnicima u generiranju, stvaranju novih ontologija i kolaborativno poboljšavanje postojećih ontologija. Ontologija pohranjena na serveru se može prebaciti u različite formate.



Slika 4, Prikaz dijela za stvaranje ontologija u Ontolingua sistemu

3 *DAML Ontology library system*

DAML ontology libray system je dio DARPA Agent Markup Language (DAML) programa, koji je počeo u Kolovozu 2000. Cilj truda DAML-a je razvitak jezika i alata koji bi omogućili stvaranje koncepta «Semantički Web». Ontološki knjižnični sistem sadrži katalog ontologija razvijenih DAML-om. Taj katalog DAML ontologija je dostupan u XML, HTML i DAML formatu. Ljudi mogu poslati nove ontologije putem javnog DAML ontološkog knjižničnog sistema.

```
<ontology uri="http://www.davincinetbook.com:8080/daml/rdf/personal-info.daml" id="2">
  <description>DAML ontology for homework 1</description>
  <poc name="Mark Neighbors" organization="Booz-Allen &amp;amp; Hamilton" email="neighbors_mark@bah.com"
  <submitter name="Mark Neighbors" organization="Booz-Allen &amp;amp; Hamilton"
      email="neighbors_mark@bah.com" date="2000-10-31"/>
  <keyword>personal information</keyword>
  <dmoz>http://dmoz.org/dmoz5</dmoz>
  <dmoz>http://dmoz.org/dmoz6</dmoz>
  <funder>DARPA DAML Program</funder>
  <class>AnnotatedBulletList</class>
    <class>Bullet</class>
  <class>BulletList</class>
  <class>Company</class> ……
……
    <property>bulletList</property>
  <property>companies</property>
  <property>currentEmployerIDs</property>
  <property>currentProjectIDs</property>
  <property>description</property>
……
<namespace>http://156.80.108.115/2000/10/daml-ont.daml</namespace>
  <namespace>http://www.w3.org/1999/02/22-rdf-syntax-ns</namespace>
  <namespace>http://www.w3.org/2000/01/rdf-schema</namespace>
 </ontology>
```

Slika 5, Primjer ontologije u DAML ontološkoj knjižnici

4. SHOE

SHOE (Simple HTML Ontology Extension) je razvijen na Univeristy of Maryland (USA). SHOE je također prvi Web-semantički jezik stvoren kao *markup*, i upotrebljavan je za razne aplikacije, uključujući i za sigurnost hrane za US Food and Drug Administration te za sistem za vojna logistička planiranja.

5. Ontology Server

Ontology Server, koji je napravljen na Vrije Universiteit u Bruxellesu, povezuje ontološko inženjerstvo sa semantičkim bazama podataka. On izdaje programske podrške na bazama podataka za upravljanje i shvaćanje ontologija. Sistem upravljanja bazom podataka (Database management system (DMBS)) opremljen sa raznim sintaktičnim konstruktorima omogućuje izradu dijagrama baze podataka za prezentiranje objekata, ograničenja integriteta, pravila deriviranja itd.

6. IEEE Standard Upper Ontology

IEEE Standard Upper Ontology (SUO) Working Group je investirala ogroman trud radeći s velikom količinom ontologista da bi stvorili standardnu vrhunsku ontologiju koja bi omogućila razne aplikacije, kao što su podatkovna inteoperabilnost, pretraživanje i dohvat informacija te procesiranje prirodnih jezika. Njihov ontološki knjižnični sistem je poprilično jednostavan te je dostupan u svojom preliminarnoj formi na njihovoj Web stranici [21]. Jedan trud je vrijedan spomena ovdje, a to je po prvi put napravljen *spajaoc* (merger) određenih SUO izvornih kodova u jednu koherentnu ontologiju koja je dostupna na njihovoj Web stranici. Taj *spajaoc* je uspješno napravljen spajanjem strukturalne ontologije Davida Whittena,

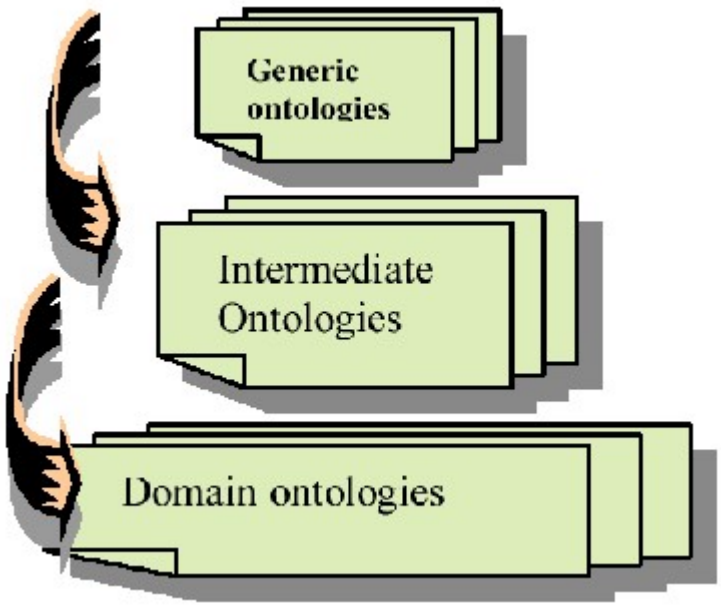
upper ontologije Johna Sowa, *temporal* logika Allensa, Casati i Varzijeva formalna teorija rupa, Smithova formalna teorija ograničavanja i dr.

7 OntoServer

OntoServer, koji je trenutno u fazi izgradnje, je ontološki server koji podržava izgradnju, održavanje i korištenje ontologija. Ima klijent/server baziranju arhitekturu, koja integrira razne tipove softvera i alata da bi formiralo alatno baziranu podršku za ontološke okoline.

8 ONIONS

ONIONS (Ontological Intergration Of Naive Sources) je metodologija za integraciju ontologija. Stvoren je u ranim devedesetim da bi riješio problem konceptualne heterogenosti. ONIONS kreira strateški dizajn ontološkog knjižničnog sistema. On sadrži bogatu dokumentaciju, formalizirane generičke ontologije i kognitivno providni gornji sloj. Štoviše, *intermediate* modul sadrži koncept domene, baziran na generičkim ontologijama i gornjem sloju.



Slika 6, Strateški dizajn ontološkog knjižničnog sistema (ONIONS)

ONIONS je opredijeljen razvoju visoko razinskog ontološkog knjižničnog sistema za medicinsku terminologiju. ONIONS je većinom metodologija integracije ontologija, koji je implementiran s puno projekata. Kreira strateški ontološki knjižnični sistem uključujući generičku ontologiju, *intermediate* ontologiju te domensku ontologiju.

Tablica 1, Usporedba ontoloških knjižničnih sistema

		WebOnto	Ontolingua	DAML library	SHOE	Ontology Server
Management	Storage	- client/server-based - no classification - modularity storage	- client/server-based - no classification - modular structured library (lattice of ontologies, naming policy)	- client/server-based - classification of ontology - no modularity storage	- web accessible - classification of ontology - tree structure of ontology dependency	- database access - no classification - modularity storage
	Identification	- unique name - unique unit name	- unique name	- unique URI and Identifier	- unique name	- unique name
	Versioning	- indirect: inherited from ancestor ontology	No versioning	No versioning	- versioning support for ontology revision	- no versioning
Adaptation	Searching	- graphical display - simple browsing	- simple browsing - idiom-based retrieval facility for simple query answering - wild-card searching - context sensitive searching - reference ontology as the index	- simple browsing	- simple browsing	- database API - DBMS - add, modify, retrieve - ontology manager - ontology browser
	Editing	TaDzeBao: - asynchronous and synchronous discussions and editing on ontologies,	- simple interfaces - collaborative ontology construction - vocabulary translation - undo/redo - hyperlinked environment	No specific editing functions	- no editing	- add, modify, retrieve
	Reasoning	- rule-based reasoning	- use situation to determine the expected properties. - ontology testing	- no reasoning	- limited reasoning support for ontology revision	- no reasoning
Standardization	Language	OCML	KIF - ontology language translation	RDF, RDFs, DAML+OIL	SHOE	XML
	Upper-level Ontology	- no standard upper-level ontology - a more fine-grained structure: based ontology, simple-time, common concepts	- public version of CYC upper-level ontology (HPKB-UPPER-LEVEL)	No standard upper-level ontology	- Base Ontology	- no standard upper-level ontology

5.3 Ontology Builder

Ontology Builder je više kor isnički kolaborativni stvaraoc ontologija i sredstvo održavanja, stvoren da ujedini najbolje dijelove već postojećih ontoloških alata s ciljem da pruže jednostavan, snažan i iskoristiv alat. Napisan je potpuno u Javi pa prema tome Ontology Builder se može pokretati na mnogim platformama. Baziran je na J2EE (Java 2 Enterprise Edition) platformi, koja je standard za implementiranje i lansiranje enteprise aplikacija. Termin «enteprise» označava: rastuću, stalno dostupnu, visoko pouzdanu te sigurnu aplikaciju.

Ontology Builder podržava koncept ponovne uporabe i uključivanje ontologija kroz relaciju «uses» (koristi). Relacija «uses» omogućuje da sve klase, instance, svi *slotovi* i *facet*-i iz uključene ontologije budu vidljivi i korišteni u ontologiji. Relacija «uses» se može lagano dodavati i izbacivati iz ontologije. Kad se relacija «uses» izbaci, može doći do nekonzistentnosti u trenutno aktivnoj ontologiji jer koncepti definirani u izbačenoj «uses» ontologiji su još uvijek referencirani iako se ta ontologija više ne koristi. Promjene na ontologiji su propagirane u stvarnom vremenu na sve ontologije koje koriste tu promijenjenu ontologiju. Relacija «uses « je tranzitivna (a «uses» b, b «uses»c povlači a «uses» c).

Ontology Builder također nam pruža:

- *«import and export» baziran na XOL-u (Ontology Exchange Language baziran na XML-u).*
- *verifikaciju osmišljenu da održava konzistentnost terminologije u jeziku*
- *Sigurnosni model za sigurnost podataka i pristupa ontologijama*

Verifikacija u stvarnom vremenu je kompleksan posao i potražuje korištenje neke vrste TMS-a (truth maintenance system) radi performansi. Ako se TMS ne koristi, temeljita provjeravanja će se morati obavljati na svim elementima u ontologiji, što nije prihvatljivo zbog performansi. Ontology Builder provodi neke verifikacije tijekom uređivanja i kreiranja (Npr., provjere za greške kod tipa vrijednosti), ali za cijelu provjeru konzistentnosti, cijeli verifikacijski postupak mora biti eksplicitno pozvan sa strane korisnika.

Ontology Builder ima trenutno najjednostavniji algoritam za nalaženje razlika između dvije ontologije. Razlike se nalaze za dva selektirana koncepta za usporedbu kao i za njihovu djecu ako imaju ista imena. Ako nema istih imena, proces usporedbe staje. Ontology Builder javlja sljedeće razlike:

- Izostanak djece/roditelja
- Izostanak *slot*-ova
- Vrijednosne, vrijednost-tip, vrijednost-domet, domenske, dokumentacijske razlike između dva koncepta

Ontology Builder pruža fleksibilni sigurnosni model koji dopušta korisnicima pristup «back-end» servisima. Svaki korisnik dobiva korisnički račun (account) na sistemu i pristup «back-end» servisima mu je omogućen samo uz ispravnu identifikaciju. Svaki korisnik ima ulogu, koja mu određuje razinu pristupa za upravljanje ontologijom ali korisnik može imati različite razine pristupa za različite ontologije. Osiguravajući podatke u ontologiji i kontrolirajući pristup «back-end» servisima, sigurnosni model Ontology Builder-a zadovoljava jedan od kritičnih zahtjeva za klase enteprise aplikacije. Ovaj sigurnosni model s ulogama pruža nam sigurnost podataka, čuva integritet podataka, omogućuje identifikaciju te višerazinski pristup korisnika.

Ontology Builder je potpuno internacionaliziran i podržava pregledavanje i uređivanje onotlogije u više lokaliteta. Jedna reprezentacija ontologije podržava sve lokalitete. Imena iz svakog lokaliteta su linkane u tu jednu reprezentaciju, tako da se promjene u strukturi ontologije na jednom lokalitet odmah propagiraju i budu vidljive na svim lokalitetima. Ontology Builder također pruža podršku prevođenja sa jednog lokaliteta na drugi.

Ontologija je sastavljena od klasa, *slot*-ova, *facet*-a koji su implementirani kao *frame*-ovi (okviri). Ontologija je također definirana kao *frame* i sadrži informacije kao npr. autor, datum stvaranja i dokumentaciju.

Klase su po početnim pretpostavkama (default) instance metaklase CLASS, ali to korisnik može mijenjati. One mogu također biti instance višestrukih metaklasa i podklase višestrukih superklasa.

Slot-ovi su definirani nezavisno od bilo koje klase, a instance su metaklase SLOT po početnim pretpostavkama, ali i to korisnik može mijenjati. Kao i klase, *slot*-ovi također podržavaju hijerarhiju višestrukog nasljeđivanja.

Facet-ovi specificiraju specifične vrijednosti za *slot*-klase ili *slot-slot* asocijacije. Ontology Builder podržava stvaranje i korištenje *user-defined facet*-a. *User-defined facet*-i se mogu kreirati i spajati na *slot* ako i samo ako je *slot* spojen na *frame*.

Ontology Builder ne koristi pohranjivanje (caching) za dohvaćanje ontoloških podataka, već koristi «lijeno» dohvaćanje informacija. Svaki komad informacije se dohvaća iz baze podataka svaki put kad dođe zahtjev. Prosječno vrijeme odgovora za dohvaćanje jednostavnog *frame*-a sa

roditeljima, djecom, metaklasama i *slot*-ovima (bez vrijednosti *slot*-a i bez *frame-slot-facet*-a) je oko 50 milisekundi (PIII 800MHz 512 RAM) što je otprilike oko 20 transakcija čitanja po sekundi. U praksi ovaj nivo performanse za Ontology Builder se pokazao zadovoljavajućim, jer stvaranje i održavanje ontologije nije performansno zahtjevan proces.

Ontology Builder klijent pruža korisniku pristupaćno sučelje (*easy-to-use interface*) za ontologiste, eksperte pojedinih domena te poslovne analitičare. Iako nisu napravljene formalne studije korisnosti, mnogi eksperti na pojedinim domenama i analitičari su uspjeli koristiti ovaj alat produktivno i s minimalnom količinom vremena potrošenog na treniranje i uvježbavanje. Međutim, vjeruje se da još postoji mjesta za poboljšanje dizajna i iskoristivosti korisničkog sučelja.

5.4 Ontology Server

Ontološki server je server s velikom performansom i velikom mogućnosti rasta. On je kritična komponenta komercijalne aplikacije koja zahtjeva ontologiju da pokreče njihovu uslugu. Ontološki server koristi potpuno istu arhitekturu i reprezentaciju kao i Ontology Builder i pruža XML i Java RMI sučelja za pristup podacima u ontologijama. Optimiziran je za pristup samo čitanja (read-only) pa je za operacije čitanja podataka iz ontologija puno brži od Ontology Builder-a. Ontološki server definira svoje vlastito sučelje, koje je jednostavnije i prikladnije za komercijalnu aplikaciju negoli općenito OKBC (Open Knowledge Base Connectivity Working Group) sučelje.

Zanemarujući umrežavanje, serijalizaciju i vrijeme traženja, vrijeme procesiranja Ontološkog servera je samo 1-3 milisekunde (PIII 800MHz 512 RAM) i kad se jednom *frame* učita u bazu podataka to vrijeme ne varira previše bez obzira na broj klijenata. Inicijalno vrijeme učitavanja svakog *frame*-a je oko 20-250 milisekunde, ovisno o broju *slot*-ova, *facet*-a, klasa, roditelja, djece i relacija metaklasa koji se trebaju dohvatiti. Kad su dohvaćeni, server aplikacija pohranjuje (cache) *frame*-ove i sljedovi zahtjeva koji su potrebni za dohvatiti taj *frame* traju samo 1-3 milisekunde bez obzira na broj klijenata koji potražuju taj *frame*. Broj pohranjenih (cached) *frame*-ova može biti specificirano kao parametar. *Frame*-ovi koji nisu bili dohvaćeni neko vrijeme će biti izbačeni i zamijenjeni sa novijim *frame*-ovima čim se limit dostigne (caching limit)

Ovdje su bila prezentirana dva Verical Net proizvoda: Ontology Builder i Ontology Server. Vjeruje se da ti proizvodi zajedno najbolje iskorištavaju ontologije te su najbolje enterprise arhitekture koje pružaju industrijski snažna rješenja za kreiranje i održavanje ontologija. Gledano prema osnovnim zahtjevima Ontology Builder i Ontology Server zadovoljavaju većinu zahtjeva. Donja tablica (tablica 2) nam prikazuje usporedbu Ontology Builder-a sa ostalim ontološkim okruženjima.

Tablica 2, Usporedba OnologyBuilder-a s ostalim ontološkim okruženjima

	Scalable Available Reliable	Ease of Use	Knowledge Representation	Multi User Collaboration	Security	Diff & Merge	Internationalization	Versioning
Ontolingua/ Chimaera	-	-	+	0	-	+	-	-
Protégé/ PROMPT	-	0	+	-	-	+	-	-
OntoWeb Tadzebao	-	0	+	+	-	-	-	-
OntoSaurus/ Loom	-	-	+	0	-	-	-	-
Ontology Builder	+	0	0	0	0	0	+	-

Iako se iz slike lako zaključi da sad već imamo razumno rješenje koje ispunjava većinu zahtjeva, vjeruje se da ima još puno mjesta za poboljšanje i puno planova za povećanjem funkcionalnosti u tim pojedinim područjima. Ovdje je predstavljeno robusno rješenje koje ispunjava većinu kritičnih zahtjeva: mogućnost rasta, dostupnost, pouzdanost te performansu.

Stvaranje ontologija je skup i dugotrajan posao. Zbog toga je korisno graditi ponovno iskoristive modularne ontologije, tako da se nove ontologije mogu napraviti brzo samo miješajući postojeće ispravne ontologije. I «Ontolingua» i «Protege» imaju sposobnost uključivanja ontologija za ponovnu uporabu. «Protege» dopušta da se uključi projekt, ali se uključeni projekt ne može lagano maknuti i ne mogu se koristiti duplicirana imena tijekom projekta tj. imena moraju biti jedinstvena. Jedinstvenost imena u programu «Protege» je ograničenje jer je dupliciranje imena vrlo česta pojava u praksi. «Ontolingua» pruža fleksibilno kombiniranje aksioma i definicija višestrukih ontologija. «Ontolingua» eliminira sukobljavanje simbola između ontologija na svoj interni način stvarajući lokalni nazivni prostor za svaki simbol definiran u svakoj ontologiji.

6 UML i Semantički Web

Vizija Semantičkog Web-a je da pustimo računalnim programima da nas oslobode teškog bremena lociranja resursa na Web-u koji su nam relevantni, te odvajanje, integriranje i indeksiranje informacija sadržanih unutar resursa. Da bi to omogućili, resursi na Web-u moraju biti kodirani ili notirani sa strukturiranim, strojevima čitljivim, opisima njihovog konteksta i to terminima i strukturama koji su eksplicitno definiran u domenskoj ontologiji.

Trenutno se jako puno istraživanja troši na razvoj jezika koji bi reprezentirao ontologiju te koji bi bio kompatibilan sa World Wide Web standardom, osobito u *Ontology Inference* sloju te DARPA Agent Markup Language projektima.

Ovaj poglavlje razmatra tehnologiju Semantičkog Web-a baziranu na alternativnim paradigmama koje također podržavaju modeliranja koncepata unutar domene (ontologije) te prikazivanje informacija u terminima tog koncepta. Postoji ekspresivan i standardiziran jezik za modeliranje, tzv. Unified Modeling Language (UML) koji sadrži grafički te XML bazirani format, veliku zajednicu korisnika, visoki nivo potpore s komercijalnim alatima. Iako je stvoren kao podrška analizi i dizajnu u softverskom inženjerstvu, UML se počinje koristiti za neke druge probleme modeliranja kao npr. Meta Data Coaliton koji ga koristi za reprezentatiranje meta podatakovnih shema za enterprise podatke.

Prema OMG (Object Management Group) specifikaciji: Unified Modeling Language je grafički jezik za vizualizaciju, specifikaciju, konstrukciju i dokumentaciju rukotvorina u softverskom sistemu. UML nudi standardni način za zapisivanje sistemskih nacrta, uključujući konceptualne stvari kao što su poslovni procesi, sistemske funkcije te konkretne stvari poput izjava programskih jezika, shema baza podataka i ponovno iskorisivih softverskih komponenata.

Unified Modeling Language je brzo postao glavni standard za izgradnju objektno orijentiranog softvera. UML je upotrebljiv za rješavanje objektno orijentiranih problema. Bilo tko, tko je zainteresiran za učenje UML-a mora biti upoznat s bitnim pravilom rješavanja objektno orijentiranih problema – sve počinje s konstrukcijom modela. Domena je stvarni svijet iz kojeg dolazimo do problema.

Model se sastoji od objekata koji komuniciraju šaljući si međusobno poruke. Objekti imaju stvari koje znaju (attributes) i stvari koje mogu raditi (behaviors ili operations). Vrijednost atributa objekta određuje njegovo stanje (state).

Klase su zapisi za objekte. Klasa obuhvaća atribute (podatke) i ponašanja (metode ili funkcije) u jedan određeni entitet. Objekti su instance klasa.

Međutim, postoji jedna negativna strana UML-a, a ta je da mu nedostaje formalna definicija. Semantika UML-a je definirana sa metamodelom, neka dodatna ograničenja su izražena u polu-formalnom jeziku (Object Constraint Language, OCL) a opisi raznih elemenata jezika opisani su na engleskom. Razvoj formalne semantike za UML je aktivno područje za istraživanja. Kako je UML jako veliki jezik sa malo redundancije, započela su istraživanja identificiranja jezgre (core) UML konstruktora modela iz kojeg bi se mogli derivirati ostali jezični elementi. Formalna definicija te jezgre će tada indirektno pružiti semantiku za cijeli jezik.

UML definira notacije i semantiku za sljedeće domene:

- *The User Interaction or Use Case Model* - opisuje ograničenja te komunikaciju između sistema i korisnika
- The Interaction or Collaboration Model - opisuje kako će objekti u sistemu komunicirati međusobno da bi obavili posao
- The Dynamic Model
- *The Logical or Class Model* - opisuje klase i objekte koji će sačinjavati sistem
- *The Physical Component Model* - opisuje softver koji sačinjava sistem
- *The Physical Deployment Model* - opisuje fizičku arhitekturu i raspored komponenata na hardverskoj arhitekturi

7 Tehnologije za razvoj Semantičkog Web-a

Dvije vrlo važne tehnologije za razvoj Semantičkog Web-a su već aktivne: *eXtensible Markup Language* (XML) i *Resource Description Framework* (RDF). XML dopušta svima da kreiraju svoje elemente (tag). Skripte ili programi mogu iskoristiti te elemente na sofistificirane načine, ali pisac skripte mora znati za što koristi koji element pisac stranice. Ukratko, XML dopušta korisnicima dodavanje proizvoljne strukture njihovim dokumentima ali ne kaže ništa o tome što strukture znače. Značenje se izražava sa RDF-om, koji ih kodira u trojke, pri čemu je svaka trojka zapravo *tema*, *riječ* te *objekt* elementarne rečenice. Te trojke se mogu zapisati pomoću XML elemenata. I tema i objekt su identificirani za URI-om (Universal Resource Identifier) isto kao da ih se koristi kao link na Web stranicama (URLs, Uniform Resource Locatora, su najčešći tipovi URI-a). Riječi su također identificirane sa URI-ovima, koji omogućuju svima da definiraju novi koncept, novu riječ samo definirajući URI za njega/nju negdje na Web-u. Kako RDF koristi URI za kodiranje informacija o dokumentu (trojke), URI-ovi osiguravaju da koncepti nisu samo riječi u dokumentu već su povezane s jedinstvenom definicijom koju svi mogu pronaći na Web-u.

7.1 eXtensible Markup Language - XML

XML je osmišljen za opis podataka. Elementi u XML nisu predefinirani već se moraju definirati vlastiti elementi. Ovaj jezik koristi *Document Type Definition* (DTD) ili XML Shemu za opisivanje podataka pa ta XML Shema zajedno s DTD-om čine samoopisivajući jezik. XML je platforma koja je softverski i hardverski nezavisan alat za transmisiju podataka.

XML nije dizajniran da nešto radi, što više on NIŠTA ne radi. Iako to je malo teško pojmljivo to je istina. On je napravljen za strukturiranje, pohranjivanje i slanje informacija. Navest ćemo jedan jednostavan primjer XML-a, i to poruku od John Doea za Nikšu Orlića:

```
<note>
<za>Nikša Orlić</za>
<od>John Doe</od>
<zaglavlje>Podsjetnik</zaglavlje>
<body>Super sam se zabavio sinoć, pusa!</body>
</note>
```

Ova poruka sadrži zaglavlje te tijelo poruke. Također sadrži ime osobe koja je poslala kao i osobe kojoj je upučena poruka. Međutim, ovaj XML dokument i dalje ne radi NIŠTA. To je samo čista informacija zamotana u XML elemente. Neko mora napisati djelić softvera za slanje, primanje i prikazivanje te poruke.

XML elementi nisu predefinirani. Treba izmisliti svoje elemente, dok su u HTML elementi predefinirani pa korisnik može samo koristiti te predefinirane elemente i niti jedan drugi. XML dopušta definiranje elemenata i strukture dokumenta. Elementi u gore navedenom primjeru (kao <za> i <od>) nisu definirani u niti jednom XML standardu. Te elemente je izmislio autor XML dokumenta.

XML će u budućnosti biti svugdje. Fascinantno je gledati kako se brzo XML standardi razvijaju, te kako brzo se povećava broj proizvođača softvera koji su prihvatili taj standard. Vjeruje

se da će XML biti važan za budućnost Web-a kao što je HTML bio za osnivanje Web-a, te da će XML biti svakodnevni alat za manipulaciju i slanje podataka.

Kako imena elementa u XML-u nisu fiksna, vrlo često dolazi do konflikta imena kada dva različita dokumenta koriste isto ime za opisivanje dva različita tipa elemenata. Npr.:

Ovaj XML dokument sadrži informacije u tablici:

```
<table>
        <tr>
                <td>Jabuke</td>
                <td>Banane</td>
        </tr>
</table>
```

dok ovaj XML dokument sadrži informacije o stoliću (table)

```
<table>
        <name>Afrički Coffe Table</name>
        <width>80</width>
        <length>120</length>
</table>
```

Ako bi se ova dva, gore navedena, dokumenta spojili zajedno, došlo bi do konflikta imena elemenata jer oba dokumenta sadrže element <table> koji ima različiti kontekst i definiciju u oba dokumenta.

Ovaj vrlo čest problem se može riješiti dodavanje prefiksa. Prethodni primjer ćemo zapisati dodavajući elementima prefikse:

Ovaj XML dokument sadrži informacije u tablici:

```
<h:table>
        <h:tr>
                <h:td>Jabuke</h:td>
                <h:td>Banane</h:td>
        </h:tr>
</h:table>
```

dok ovaj XML dokument sadrži informacije o stoliću (table)

```
<f:table>
        <f:name>Afrički Coffe Table</f:name>
        <f:width>80</f:width>
        <f:length>120</f:length>
</f:table>
```

Sad je nestao problem konflikta imena elemenata jer ta dva dokumenta koriste različita imena za njih <table> element (<h:table> i <f:table>).

7.1.1 TEHNOLOGIJE BAZIRANE NA XML-u

1) *XHTML* – Extensible HTML, to je preformulacija HTML 4.01 u XML. XHTML 1.0 je zadnja verzija HTML-a.

2) *CSS* – Cascading Style Sheets, CSS Style Sheets se mogu dodavati u XML dokumente da bi dobili informaciju o izgledu.

3) *XSL* – Extensible Style Sheet Language, XSL se sastoji od tri dijela: XML Document Transformation, sintaksa za povezivanje uzoraka i interpretacija formatiranja objekata.

4) *XSLT* – XML Trasnformation, XSLT je mnogo jači od CSS-a. Može se koristiti za pretvorbu XML datoteka u razne izlazne formate.

5) *Xpath* – XML Pattern Matching, Xpath je jezik za adresiranje dijelova XML dokumenta. Xpath je dizajniran da bi bio korišten i sa XSLT-om i Xpointer-om.

6) *Xlink* – XML Linking Language, Xlink omogućuje dodavanje elemenata u XML dokumente sa svrhom stvaranja linkova između XML resursa.

7) *Xpointer* – XML Pointer Language, Xpointer podržava adresiranje internih dijelova struktura XML dokumenata kao što su elementi, atributi i sadržaji.

7.1.2 VEZA IZMEĐU HTML-a I XML-a

XML je stvoren da opisivanje podataka te je usredotočen na što je podatak, dok je HTML (HyperText Markup Language) stvoren za prikaz podataka te je usredotočen na kako podatak izgleda. XML nije zamjena za HTML. Svaki od njih je osmišljen sa drugim ciljem. HTML koristi predefinirane elemente, dok XML nema predodređenih elemenata pa svaki korisnik mora izmisliti svoje nove elemente.

Web baziran na HTML-u nije semantički jer mu nedostaje značenje. Semantički Web zahtjeva sadržaje koji su sa značenjem i lagani za procesiranje. Tu u igru dolazi XML. Očito je da XML kod se može jednako lagano pročitati ljudskim okom i procesirati kompjuterskim programom. Kao rezultat možemo puno lakše iskoristiti sadržaj u druge svrhe, mijenjati prezentacijski format ili izvaditi bitne informacije.

XML pruža puno restriktivniju strukturu nego HTML, koja omogućuje ekstrakciju informacija. Štoviše, XML je puno stroži od HTML-a. Npr., u XML dokumentu element <td> mora završiti sa </td> elementom jer inače se neće prikazati na pretraživaču koji podržava XML. XML je dizajniran, za razliku od HTML-a, da bude lagan za parsiranje. Postoje mnogi standardi i mnogi XML parseri koji jako pomažu prilikom ekstrakcije podataka. Također je veličina datoteke u XML puno manja od one napisane u HTML-u, te ta manja veličina datoteke su uzrok efikasne transmisije i procesiranja.

7.2 SIMPLE OBJECT ACCESS PROTOCOL - SOAP

SOAP je jednostavan protokol za izmjenu informacija u decentraliziranom, distribuiranom okruženju. To je protokol baziran na XML-u, koji se sastoji od tri dijela: omotnice (envelope) koja definira *framework* za opisivanje što je u poruci i kako to procesirati, skupa pravila za

izražavanje instanci aplikacijski definiranih tipova podataka te konvencija za reprezentiranje udaljenih proceduralnih poziva i odgovora. SOAP može biti korišten u kombinaciji sa nizom drugih protokola.

Glavni cilj dizajniranja SOAP-a je jednostavnost i proširivost. To znači da postoji nekoliko dodataka iz tradicionalnog sistema poruka i distribuiranog objektnog sistema koji nisu dio jezgre SOAP specifikacije. Takvi dodaci uključuju:

- *Distributed garbage collection*
- *Boxcarring or batching of message*
- *Objects-by-reference* (zahtjeva distributed garbage collection)
- *Activation* (zahtjeva objects-by-reference)

7.2.1 SOAP Model Razmjene Poruka

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
       <m:GetLastTradePrice xmlns:m="Some-URI">
           <symbol>DIS</symbol>
       </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Slika 7, SOAP poruka uključena u HTTP zahtjev

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
       <m:GetLastTradePriceResponse xmlns:m="Some-URI">
           <Price>34.5</Price>
       </m:GetLastTradePriceResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

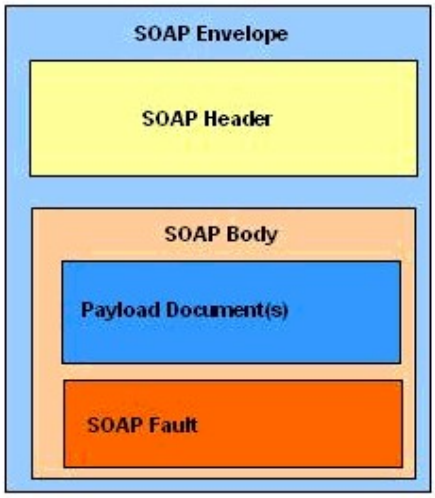
Slika 8, SOAP poruka uključena u HTTP odgovor

SOAP poruke su fundamentalno jednosmjerne transmisije od pošiljatelja do primatelja, ali kako je ilustrirano u gornja dva slučaja, SOAP poruke se često kombiniraju da tvore zahtjeve i potvrde primanja od pošiljatelja i primatelja.

SOAP poruka je XML dokument koji se sastoji od obavezne SOAP omotnice, opcionalnog zaglavlja (header) te obaveznog SOAP tijela (body). Omotnica je vršni element XML dokumenta koji reprezentira poruku. Ako se zaglavlje pojavi ono mora biti prvo dijete koje slijedi element omotnice (envelope). Tijelo (body) poruke mora se nalaziti odmah iza omotnice ili ukoliko postoji zaglavlje, odmah iza zaglavlja. Iako su zaglavlje i tijelo poruke definirani kao zasebni elementi, oni su u stvari povezani.

Prema SOAP specifikaciji postoje dva stila SOPA-a: doslovni (literal) i kodirani (encoded). Doslovni stil SOAP-a u biti znači da se šalje XML dokument kao teret SOAP poruke. Kad se koristi doslovni stil, obično se misli u terminima slanja poruka: Klijent šalje poruku servisu, servis procesira tu poruku te šalje nazad poruku s odgovorom. Treba naznačiti činjenicu da u prethodnoj rečenici nisu bili korišteni termini kao objekt, metoda.

Slika 9, SOAP poruka



The SOAP message. When using literal-style SOAP, the payload contains an XML document that is your data. With RPC or encoded SOAP messages, the payload contains application data (e.g. objects and primitive types) serialized to XML.

Kodirane SOAP poruke koriste standardni način za serijalizaciju objekata i druge strukturirane podatake u XML-u. Cilj ove vrste poruke je da se pozivi udaljenih procedura (Remote Procedure Calls - RPC) transparentno naprave dostupnim klijentu. Npr., klijent misli da poziva metodu na lokalnom objektu kad zapravo se ta metoda prevodi u XML i šalje preko mreže, procesira na drugoj strani te se poruka s odgovorom pažljivo vraća kao povratna vrijednost poziva metode. Klijent uopće nije svjestan da se dogodila razmjena podataka.

SOAP Fault element služi za prenošenje informacije o statusu unutar SOAP poruke. Ako postoji, *SOAP Fault* element MORA se pojaviti kao dio u tijelu a NE SMIJE se pojaviti više od jednom unutar elementa tijelo (body). *SOAP Fault* element definira sljedeće podelemente:

• *faultcode* – faultcode element je namijenjen za uporabu softvera da bi pružilo algoritamski mehanizam za identificiranje greške. On se MORA nalaziti unutar SOAP Fault elementa te vrijednost faultcode-a mora biti kvalificirano ime.

• *faultstring* – faultstring element je namijenjen da pruži ljudima čitko objašnjenje o grešci te nije namijenjen za algoritamsko procesiranje. MORA biti nazočan u *SOAP Fault* element i TREBAO bi isporučiti bar neke informacije objašnjavajući prirodu greške.

• *faultactor* – faultactor element je namijenjen da pruži informaciju o tome tko je izazvao grešku na putanji poruke. Vrijednost faultactor atributa je URI koji identificira uzrok.

• *detail* – element namijenjen za prenošenje specifičnih informacija čvorovima. To su informacije o greškama koje su povezane s elementom tijela (body). Mora biti nazočna ako kontekst, unutar elementa *tijelo* (body), nije uspješno procesirano.

Bez obzira na koji je protokol SOAP vezan, poruke se šalju okolo po tzv. putanjama poruka (message path), koji dopuštaju procesiranje na jednom ili više čvorova, prema krajnjim odredištima.

SOAP aplikacija koja prima SOAP poruku MORA procesirati tu poruku obavljajući sljedeće korake koji su navedeni:

1) Identificirati sve dijelove SOAP poruke koji su namijenjeni za taj čvor.
2) Verificirati da svi identificirani mandatorni dijelovi su podržani u aplikaciji za ovu poruku i odgovarajuće ih procesirati. Ukoliko to nije slučaj, poruka se odbacuje. Procesor MOŽE ignorirati pojedine dijelove identificirane u koraku jedan bez da to utječe na rezultat procesiranja.
3) Ako SOAP aplikacija nije krajnje odredište poruke onda makni sve dijelove identificirane u koraku jedan prije prosljeđivanja poruke.

Procesiranje poruke ili dijela poruke zahtjeva da SOAP procesor razumije, među ostalim, patent razmjene koji se koristi (one way, request/response, multicast itd.), ulogu primaoca u patentu, zapošljavanje RPC mehanizama, reprezentaciju podataka kao i ostale semantike potrebe za ispravno procesiranje.

7.2.2 Veze s XML-om

Sve SOAP poruke su kodirane korištenjem XML-a. SOAP aplikacija bi TREBALA sadržavati odgovarajući SOAP nazivni prostor za sve elemente i atribute definirane u poruci generiranoj SOAP-om. Mora biti sposobna procesirati nazivne prostore SOAP-a u poruci koju primi. Mora odbaciti poruke koje imaju pogrešan nazivni prostor i MOŽE procesirati SOAP poruku bez SOAP nazivnog prostora kao da imaju ispravan SOAP nazivni prostor. SOAP poruka NE SMIJE sadržavati DTD (Document Type Declaration) i instrukcije procesiranja.

7.2.3 Korištenje SOAP-a za pozive udaljenih procedura

Jedan od dizajnerskih ciljeva SOAP-a je enkapsulacija i razmjena RPC poziva koristeći proširljivost i fleksibilnost XML-a. Korištenje SOAP-a za RPC nije ograničeno na spajanje HTTP protokolom.

Da bi omogućili poziv metode, sljedeće informacije moraju biti dostupne:

- *URI ciljnog objekta*
- *Ime metode*
- *Parametri metode*
- *Opcionalni podaci zaglavlja*

Pozivi i odgovori RPC metoda se odvijaju u SOAP elementu tijelo (body) koristeći sljedeće reprezentacije:

- Poziv metode je modeliran kao struktura.
- Poziv metode se vidi kao jedna struktura koja sadrži pristup za svaki ulazni odnosno ulazno/izlazni parametar. Struktura je imenom i tipom identična imenu metode.
- Svaki ulazni ili ulazno/izlazni parametar se vidi kao pristup, sa imenom koje odgovara imenu parametra, te sa tipom koji odgovara tipu parametra.
- Odgovor metode je modeliran kao struktura.
- Odgovor metode se vidi kao jedna struktura koja sadrži pristup za povratnu vrijednost te za svaki ulazni odnosno ulazno/izlazni parametar.
- Svaki parametarski pristup ima ime koje odgovara ime parametra te tip koji odgovara tipu parametra. Ime pristup povratne vrijednosti nije bitno. Ime strukture također nije bitno, ali običaj je imenovati strukturu po imenu metode s dodatkom «odgovor» (Response).
- Greška u metodi je implementirana koristeći *SOAP Fault* element.

Aplikacija može procesirati zahtjev kojem nedostaju parametri ali također može vratiti grešku u tom slučaju. A kako rezultat indicira uspjeh a greška indicira neuspjeh, pojavljivanje i greške i rezultata kao rezultat metode se smatra greškom.

7.3 Resource Description Framework - RDF

Resource Description Framework je jednostavni resurs-vlasništvo-vrijednost model dizajniran za prikazivanje meta podataka (podatke o podacima) o resursima na Web-u. RDF ima grafičku sintaksu i XML baziranu serijsku sintaksu. RDF Shema je skup predodređenih resursa. RDF je model i sintaksa XML-a za reprezentiranje informacija na način da omogući drugim programima da razumiju značenje. Sagrađen je na konceptu izjava, formi trojki: riječ, tema, objekt (predicate, subject, object). Interpretacija trojke je da <subject> ima svojstvo <predicate> čija je vrijednost <object>. U RDF-u <subject> je uvijek resurs identificiran sa URI-om. <predicate> je vlasništvo resursa a <object> je vrijednost vlasništva resursa.

RDF je temelj za procesiranje meta podataka (podaci o podacima) te pruža interoperabilnost između aplikacija koje izmjenjuju informacija razumljive strojevima na Web-u. RDF se može iskoristiti u raznim područjima aplikacija npr. otkrivanje resursa da bi postigli bolje pretraživačke sposobnosti, inteligentni softverski agenti koji omogućuju dijeljenje i izmjenjivanje podataka i znanja itd. RDF sa digitalnim potpisom (digital signature) će biti ključ u izgradnji web-a povjerenja.

Najprimitivniji cilj RDF-a je definiranje mehanizma za opisivanje resursa koji ne radi pretpostavke o pojedinoj aplikacijskoj domeni niti definira semantiku bilo koje aplikacijske domene. Definicija mehanizma bi trebala biti neutralna s obzirom na domenu, ali bi mehanizam trebao biti prikladan za opisivanje informacija o domeni.

RDF model se može reprezentirati grafički i XML-om. RDF potražuje da drugačiji tipovi semantičkih informacija (subjects, properties, values) budu stavljeni na predodređene lokacije u XML-u. Programi koji čitaju XML zapis RDF-a mogu onda zaključiti da li je određeni element ili atribut se odnosi na *subject*, vlasništvo (property), ili vrijednost vlasništva (value of property).

Pojasnimo malo termine *resources*, *properties* te *statements*.

Resources:

Sve stvari opisane s RDF izrazima se zovu resursima. Resurs može biti cijela Web stranica, može biti dio neke Web stranice ali i kolekcija Web stranica. Resursi su identificirani s URI-ovima.

Properties:

Properties su specifični aspekti, karakteristike, atributi ili relacije korišteni za opis resursa. Svaki *property* ima svoje specifično značenje, definira svoju dopustivu vrijednost, tip resursa koji opisuje te vezu prema ostalim opisima.

Statements:

Specifični resurs zajedno sa imenovanim vlasništvom (property) te vrijednosti tog vlasništva za taj resurs čine RDF *statement* ili RDF izjavu. Ta tri individualna djela izjave zovu se *subject*, *predicate,* i *object*. Objekt izjave može biti drugi resurs, literal itd.

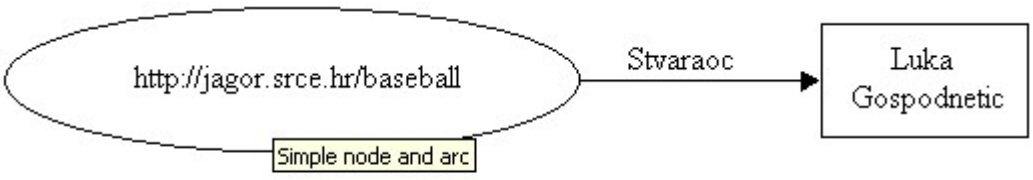
Primjer 1:

Luka Gospodnetić je stvaraoc resursa http://jagor.srce.hr/baseball.

Ova rečenica ima sljedeće dijelove:

Subjekt (resource)	http://jagor.srce.hr/baseball
Predika (property)	Stvaraoc
Objekt (literal)	«Luka Gospodnetić»

Ti dijelovi se mogu zapisati pomoću dijagrama popularno nazvanog "*nodes and arcs diagrams*" pri čemu je jako bitan smjer strelice. Strelica uvijek počinje kod *subjekta* a pokazuje na objekt izjave:

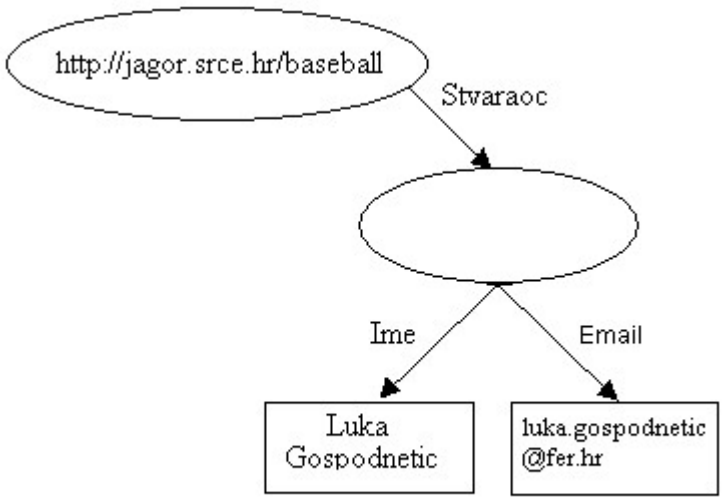


Simple node and arc diagram

Primjer 2.

Individualac čije je ime Gospodnetić Luka, Emil <luka.gospodnetić@fer.hr>, je stvaraoc http://jagor.srce.hr/baseball.

Poanta ovog primjera tj. ove rečenice je vrijednost vlasništva stvaraoca strukturiranim entitetom. U RDF-u se takav entitet reprezentira sa drugim resursom. Gore navedena rečenica nam ne daje ime tog resursa, on je nepoznat, pa ćemo ga u dijagramu niže prikazati kao praznu elipsu:



Property with structured value

Ovaj dijagram bi se također mogao pročitati na slijedeći način:

«*http://jagor.srce.hr/baseball ima stvaraoca* nešto *i* nešto *ima ime Gospodnetić Luka i Email <luka.gospodnetic@fer.hr>.*»

7.3.1 SHEME I NAZIVNI PROSTORI U RDF-u

Kad pišemo rečenice u prirodnom jeziku koristimo riječi koje su namijenjene da daju neko određeno značenje. To značenje je bitno za razumijevanje izjave a u RDF-u je krucijalno za uspostavu ispravnog procesiranja. Krucijalno je da i pisac i čitaoc izjave shvate isto značenje terminologije korištene, kao npr. Stvaraoc, OdobrenoOd, Copyright inače će rezultat biti konfuzija.

Značenje u RDF-u je izraženo preko referenciranih shema. Može se slikovito prikazati da je shema neka vrsta rječnika. Shema definira termine koji će biti korišteni u RDF izjavama i daje im specifično značenje. Razne forme shema se mogu koristiti sa RDF-om, uključujući specifične forme definirane u drugim dokumentima (RDFSchema) koji imaju neke specifične karakteristike koje bi nam pomogle u automatizaciji poslova koristeći RDF.

Shema je mjesto gdje su sve definicije i restrikcije korištenja vlasništva (properties) dokumentirane. Da bi izbjegli konfuziju između nezavisnih, te moguće konfliktnih, definicija istog termina, RDF koristi XML prostor imena. Nazivni prostor je jednostavan način za povezivanje specifično korištenje riječi u kontekstu sa rječnikom (shemom) gdje se

navedena definicija nalazi. U RDF-u svaki korišteni predikat mora biti identificiran sa točno jednim nazivnim prostorom, ili shemom.

7.3.2 KVALIFICIRANE VRIJEDNOSTI VLASNIŠTVA

Često je vrijednost vlasništva nešto što ima dodatne kontekstualne informacije koje se smatraju «dijelom» te vrijednosti. U drugim riječima, dolazimo do potrebe za kvalificiranje vrijednosti vlasništva. Za neke slučajeve prikladno je koristiti vrijednost vlasništva bez kvalifikatora. Npr., u izjavi «Cijena te olovke je 65 Hrv. kn» često je dovoljno reći jednostavno «Cijena te olovke je 65 kn» pri čemu smo izbacili riječ «hrvatskih».

7.3.3 KONTEJNERI

Često je potrebno pozivati se na kolekciju resursa, npr. reći da je na obavljenom poslu radila više nego jedna osoba, lista studenata na kolegiju ili lista softverskih modula u paketu. RDF kontejneri se koriste za pohranjivanje takvih lista resursa ili literala.

RDF definira tri tipa kontejnerski objekata: *Bag, Sequence, Alternative*.

1) Bag: Neporedana lista resursa ili literala. *Bag* se koristi za deklariranje da vlasništvo ima višestruke vrijednosti te da nije bitno u kojem su poretku te vrijednosti. Dopuštena su duplicirane vrijednosti.

2) *Sequence*: Poredana lista resursa ili literala. *Sequence* se koristi za deklariranje da vlasništvo ima višestruke vrijednosti te da je poredak bitan. Može se koristiti za održavanje abecednog poretka vrijednosti. Dopuštene su duplicirane vrijednosti.

3) *Alternative*: Lista resursa i literala koja reprezentira alternative za (jednostruku) vrijednost vlasništva. Može se koristiti za pružanje alternativnih jezičnih prijevoda za tip posla, ili za popis Internet *mirror site*-ova gdje se resursi mogu naći. Aplikacija koja koristi vlasništvo čije je vrijednost *Alternative* svjesne su toga da mogu izabrati bilo koji element u listi po želji i potrebi.

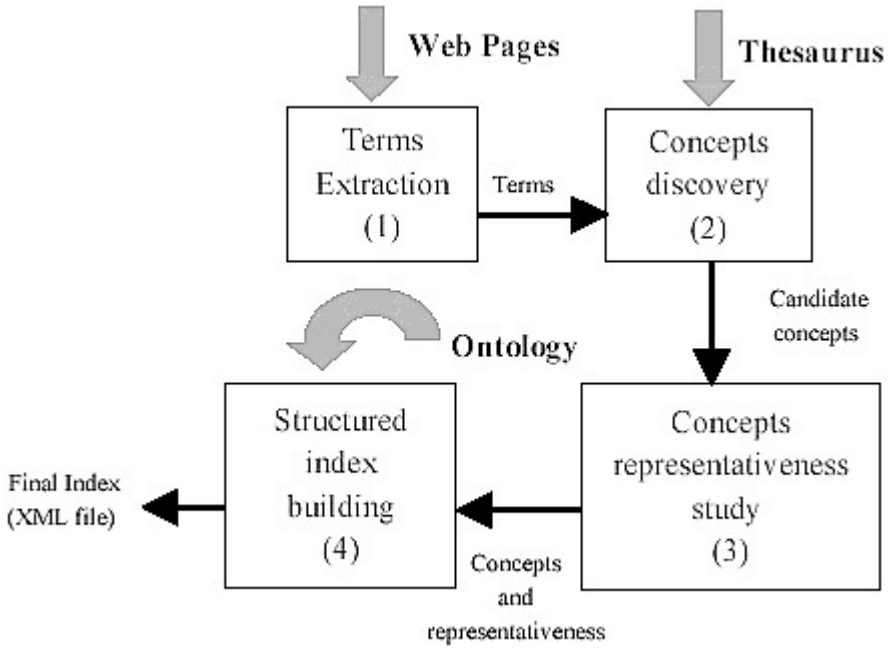
8 Indeksiranje Web Stranica sa terminološki orjentiranom ontologijom

Glavni cilj je izgradnja strukturiranog indeksa Web stranica u skladu s ontologijom. Proces indeksiranja može se podijeliti na četiri dijela:

1. Za svaku stranicu napravi se indeks. Svaki termin ovog indeksa se asocira s njegovom težinskom frekvencijom.
2. Enciklopedija (npr Wordnet Enciklopedija))nam omogućuje generiranje svih kandidirajućih koncepata koji mogu biti označeni sa terminom prethodnog indeksa.
3. Svaki kandidirajući koncept stranice se promatra da bi se odredili reprezentativni koncepti sadržaja stranice. Ova procjena je bazirana na težinskoj frekvenciji i na vezama sa drugim konceptima. To omogućuje biranje najboljeg smisla termina u vezi s kontekstom. Prema tome, što više veza ima između jednog konteksta prema drugim te što su jače te veze, tim je veći značaj tog koncepta na stranici.
4. Između ovih kandidirajućih koncepata, gradi se filtar. Naime, selektirani koncept je kandiditirajući koncept koji pripada ontologiji i ima veliku reprezentativnost sadržaja stanice. I sada se stranice koje sadržavaju takav selektiran koncept dodjeljuju tom konceptu u ontologiji.

Neke mjere su evaluirane da bi se okarakteriziralo indeksirajući proces. One određuju adekvatnost između Web stranice i ontologije. Te mjere uzimaju u obzir broj stranica selektiranih od ontologije, broj koncepata uključenih u stranici itd. Indeks se izgradi kao XML datoteka i neovisna je od Web stranica.

Ovaj proces je poluautomatiziran. Dopušta korisniku da ima globalni pogleda na Web stranicu. Također dopušta nam indeksiranje stranice iako nismo vlasnici te stranice.



Slika 10, Proces indeksiranja

8.1 Građenje indeksa

Postoje dva bitna koraka: (1) vađenje termina iz Web stranice i računanje težinske frekvencije te (2) određivanje kandidirajućeg koncepta i računanje reprezentativnosti koncepta.

8.1.1 VAĐENJE TERMINA

Dobro formirani proces vađenja termina započinje s (1) uklanjanjem HTML elementa s Web stranice, (2) dijeljenje teksta u nezavisne rečenice te (3) lematiziranje riječi na stranici. Nakon toga se stranice notiraju (upotpune s bilješkama) sa govornim elementima koristeći *Brill tagger* pa kao rezultat dobivamo svaku riječ na stranici notiranu sa odgovarajućom gramatičkom kategorijom (imenica, pridjev, ...). Na kraju se površinska struktura rečenica analizira koristeći šablone (patterns) termina (imenica, imenica+imenica, pridjev+imenica, ...) da bi se dobili ispravno formirani termini. Za svaki selektirani termin, izračunavamo težinsku frekvenciju. Težinska frekvencija uzima u obzir frekvenciju pojavljivanja termina te naročito frekvenciju pojavljivanja HTML elemenata koji su vezani uz svako pojavljivanje termina. Može se vidjeti da frekvencija nije glavni kriterij. Utjecaj elementa ovisi o njegovoj ulozi na stranici. Npr., element «title» će imati vrlo veliku važnost na termin (*10) dok će element za *podebljane* znakove <b> imati znatno manji utjecaj (*2). Doljnja tablica demonstrira težine nekih važnijih elemenata. Rezultati u toj tablici su dobiveni eksperimentalnim putem.

Tablica 3, Težine pojedinih HTML elemenata

HTML marker description	HTML marker	Weight
Document title	<TITLE></TITLE>	10
Keyword	<meta name="keywords" ... content=...>	9
Hyper-link	<A HREF=...></A>	8
Font size 7	<FONT SIZE=7></FONT>	5
Font size +4	<FONT SIZE="+4"></FONT>	5
Font size 6	<FONT SIZE=6></FONT>	4
Font size +3	<FONT SIZE="+3"></FONT>	4
Font size +2	<FONT SIZE="+2"></FONT>	3
Font size 5	<FONT SIZE=5></FONT>	3
Heading level 1	<H1></H1>	3
Heading level 2	<H2></H2>	3
Image title	<IMG ... ALT="...">	2
Big marker	<BIG></BIG>	2
Underlined font	<U></U>	2
Italic font	<I></I>	2
Bold font	<B></B>	2
...	...	...

Na Web stranici koja sadrži n različitih termina, za zadani termin T_i (gdje i ide od 1 do n), težinska frekvencija $F(T_i)$ se određuje kao suma od p težina HTML elemenata povezanih sa p pojavljivanja termina. Rezultat se onda normalizira. Gore opisani izračun je prikazan u donje dvije formule gdje $M_{i,j}$ odgovara težini HTML elemenata povezanih sa jotim pojavljivanjem termina T_i.

$$P(T_i) = \frac{P(T_i)}{\max_{k=1..n}(P(T_k))}$$

$$P(T_i) = \sum_{i=1}^{p}(M_{i,j})$$

8.1.2 ODREĐIVANJE KONCEPTA STRANICE

Tijekom procesa izdvajanja termina, zadovoljavajući termini i njihovi frekvencijski koeficijenti su izdvojeni i izračunati. Zadovoljavajući termini su različite forme koje reprezentiraju pojedini koncept (Npr., «chair», «professorship»...) Da bi se odredile grupe koncepata na stranici, a ne samo grupe termina, koristi se enciklopedija (thesaurus). Proces generiranja kadidirajućih koncepata je vrlo jednostavan: iz izdvojenih termina, svi smisleni kandidirajući koncepti se generiraju koristeći enciklopediju. Smisao se reprezentira s listom sinonima (lista je jedinstvena za dani koncept). Onda se za svaki kandidirajući koncept računa reprezentativnost koristeći težinsku funkciju te ukupnu sličnost koncepta prema ostalim konceptima na stranici. Ovaj zadnji korak se bazira na sličnosti dva koncepta ali u to nećemo ulaziti jer to povlači neke matematičke pretpostavke. Također iz istog razloga nećemo ulaziti u detaljno razmatranja izračunavanja koeficijenta reprezentativnosti.

8.2 Iskorištavanje ovog pristupa na pretraživačke alate

Većina pretraživačkih alata koristi jednostavne riječi za indeksiranje Web stranica. Upiti su često napravljeni od liste ključnih riječi povezanih logičkim operatorom («and», «or», ...). U našem kontekstu, mi koristimo terminološki orjentiranu ontologiju te strukturirano indeksiranje da bi poboljšali upit/odgovor procese. Ovaj pristup pruža nekoliko poboljšanja:

1. *korisnikov upit se širi: termini su pretvoreni u koncepte*
2. *logički operatori imaju bogatiju semantiku nego u jednostavnom svijetu ključnih riječi*
3. *odgovori su puno prikladniji korisnikovom upitu*

Širenje upita je prema tome poboljšano koristeći ontologije. Često kad korisnik preda upit koji sadrži termine povezane logičkim operatorima, rezultat je neodređen. U našem pristupu, termini se zamjenjuju njihovim odgovarajućim konceptima. Prvo se izabere kandidirajući koncept u ontologiji. Onda se proučavaju ostali koncepti i logički operatori u upitu. Ako grupa termina iz upita nisu povezana s niti jednim konceptom na kraju procesa, onda se smatraju nevažnima za stranicu.

8.3 Zaključak

Ovaj proces dovodi brojne prednosti starim tradicionalnim metodama indeksiranja:

1. selektirane stranice sadrže tražene koncepte a ne samo ključne riječi
2. ti koncepti su reprezentativci tema tih stranica
3. termini koji su zaslužni za selektiranje stranice nisu nužno oni koji su bili u zahtjevu već mogu biti sinonimi
4. stranice mogu obuhvatiti ne samo traženi koncept već i specifičniji
5. važnost koncepta ne ovisi samo o frekvenciji termina već i o HTML elementima koji ih opisuju

9 Zaključak

Puno se uzbudljivih stvari trenutno događaju u W3C Semantic Web Activity. Radne grupe se stvaraju a tehnologije se standardiziraju na nekim tehnološkim slojevima koji su dobro shvaćeni. Već je moguće implementirati konkretne aplikacije bazirane na ovome. Rad je prijeko potreban za razvijanje alata i stvaranje laganih korisničkih sučelja koji podržavaju korisnike u korištenju meta podataka i dodavanju meta podataka na Web. Ta podrška i automatizacija će biti kritična u razvoju Semantičkog Web-a. Brojčano povećanje te bogatiji meta podaci omogućit će puno veće šanse za razne aplikacije.

Moglo bi se reći da je semantički Web «Potraga za savršenstvom» Nedavni i rastući interes u Semantički Web je bio uzrok probujale i rastuće aktivnosti u standardizacijskim tijelima (W3C) da bi se specificirala semantika korištenjem formalnih jezika i mehanizama sa sposobnosti zaključivanja. Najveći problem je i dalje način povezivanja formalnih semantika sa dubljim značenjem.

Semantički Web će donijeti strukturu smislenom sadržaju Web stranica, kreirajući okruženje gdje softverski agenti, koji «šetaju» od stranice do stranice, mogu brzo i efikasno obavljati sofistificirane poslove za korisnike.

Semantički Web nije poseban Web, već je to nastavak postojećeg Web-a, u kojem se informacijama daje dobro definirano značenje. Kao i Internet, Semantički Web će biti decentraliziran koliko god je to moguće. Decentralizacija zahtjeva kompromise: Web je morao odbaciti totalnu konzistenciju svih svojih interkonekcija, javljajući svima nam poznatu poruku «Error 404: Not Found» ali omogućavajući nesputani eksponencijalni rast.

Semantički Web je vrlo dobra ideja, iako malo preidealizirana. Čini se neizbježna... možda zato jer su počela razna zbivanja i istraživanja. Uglavnom, jednostavno rečeno to je vizija budućnosti Web-a, gdje postoji zajednički jezik koji sve Web stranice razumiju, dopuštajući im da međusobno komuniciraju jedna s drugom.

10 Literatura

[1] INTERNET: Robin Cover: XML and Semantic Transperency, 23.08.1998 *http://xml.coverpages.org/xmlAndSemantics.*html, 05.01.2003

[2] INTERNET Vijesti o Semantičkom Web-u, *http://xml.coverpages.org/xmlAndSemanticWeb.html*, 05.01.2003

[3] INTERNET Resource Description Framework Model and Syntax Specification, *http://www.w3.org/TR/1999/REC-rdf-syntax-19990222*, 05.01.2003

[4] INTERNET Članci o Semantićkom Web-u, *http://www.scientificamerican.com*, 05.01.2003

[5] INTERNET Grupa autora: Simple Object Access Protocol (SOAP) 1.1, *http://www.w3.org/TR/2000/NOTE-SOAP-20000508,* 05.01.2003

[6] INTERNET SOAP: RPC or Messaging?, *http://www.vbws.com/tutors/rpcmsg/rpcmsg.aspx#mytop*, 05.01.2003

[7] INTERNET UML Tutorial, *http://www.sparxsystems.com.au/UML_Tutorial2.htm*, 05.01.2003

[8] INTERNET The First Semantic Web Working Symposium, *http://www.semanticweb.org/SWWS/program/full/SWWSProceedings.pdf*, *05*.01.2003

[9] INTERNET Web-Ontology Working Group, *http://www.w3.org./2001/sw/WebOnt/*, 05.01.2003

[10] INTERNET RDFCore Working Group, *http://www.w3.org/2001/sw/RDFCore/*, 05.01.2003

[11] INTERNET XML Introduction – What is XML, *http://www.w3schools.com/xml/xml_whatis.asp#top*, 05.01.2003

[12] INTERNET XML Introduction – XML Namespaces, *http://www.w3schools.com/xml/xml_namespaces.asp#top*, 05.01.2003

[13] INTERNET XML Introduction – XML Technologies, *http://www.w3schools.com/xml/xml_technologies.asp#top*, 05.01.2003

[14] INTERNET Practical UML : A Hands-on Introduction for Developers, *http://www.togethersoft.com/services/practical_guides/umlonlinecourse,* 05.01.2003

[15] INTERNET The Semantic Web, HTML and XML,
http://dcm.cl.uh.edu/yue/papers/SemanticWebandHTML.asp, 05.01.2003

[16] INTERNET W3C Semantic Web Activity,
http://www.w3.org/2001/12/semweb-fin/w3csw, 05.01.2003

[17] INTERNET What is RDF,
http://www.xml.com/pub/a/2001/05/23/jena.html, 05.01.2003

[18] INTERNET The Semantic Web – LCS Seminar by Tim Berners-Lee,
http://www.w3.org/2002/Talks/09-lcs-sweb-tbl/, 05.01.2003

[19] INTERNET Tutorial On Knowledge Markup And Resource Semantics,
http://www.dfki.uni-kl.de/km/kmrs/ppframe.htm, 05.01.2003

[20] INTERNET Semantic Web Activity Statement,
http://www.w3.org/2001/sw/Activity, 05.01.2003

[21] INTERNET Standard Upper Ontology (SUO) Working Group, 24.01.2002,
http://suo.ieee.org/, 05.01.2003

[22] Ventrone, V. i Heiler, S. (1991) Semantic heterogeneity as a result of domain evolution. *SIGMOD Record (ACM Special Interest Group on Menagement of Data)*

[23] Fensel, D. (2001) Ontologies: Dynamic networks or formally represented meaning.

[24] Berners-Lee, T. (1996) Generic resources. Design Issues.

[25] Berners-Lee, T., Fielding, R., i Masinter, L. (1998) RFC 2396: Uniform Resource Identifiers (URI): Generic syntax. Status: DRAFT STANDARD

[26] Gruber. T.R. (1993) A translation approach to portable ontology specifications, *Knowledge Acquisition*