SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Tomislav Štritof

VIZUALIZACIJA ZNANJA NA WEBU

ZAVRŠNI RAD

Varaždin, 2009.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Tomislav Štritof
Redoviti student
Broj indeksa: 35393/06-R
Smjer: Informacijski sustavi
Preddiplomski studij

VIZUALIZACIJA ZNANJA NA WEBU

ZAVRŠNI RAD

Mentor:
Mr. sc. Markus Schatten

Varaždin, rujan 2009.

Sadržaj

1. Uvod

Brz razvoj Interneta proteklih godina postavio je nove izazove pred ljude koji se bave standardizacijom Weba, prvenstveno pred W3C[1] konzorcij. Njihova misija je razvoj protokola i smjernica koje će osigurati da Web u potpunosti iskoristi svoj potencijal te da nastavi rast kroz duži period. Ogromna količina informacija, čiji se obujam povećava svakog trenutka, velik je izazov za prosječnog korisnika kada mu je potrebna prava i konkretna informacija. Tu su od velike pomoći pretraživači i stranice već poznate od prije koje su provjereno korisniku pružile ono što je tražio.

Problem se javlja kada taj ogroman fond informacija dostupan na Webu želimo sistematizirati i povezati tako da Web pretraživaču ne postavljamo kao upit ključne riječi nego konkretna pitanja. Računalo teško upravlja informacijama na nekoj višoj razini od pretraživanja ključnih riječi. Tako se javila potreba za semantičkim Webom.

Današnji *de facto* standard za prikaz informacija na Internetu je HTML[2] jezik. Omogućuje jednostavnu vizualizaciju informacija čovjeku kako bi se sadržaj približio što većem broju korisnika. No ta jednostavnost ima negativnu stranu: pretraživanjem Weba sve je teže odvojiti važno od nevažnih i nepovezanih informacija. Današnji Web, osim što nama otežava pronalazak pravih informacija, također onemogućuje računalu da dostupne ključne riječi na nekoj web stranici uspije sistematizirati i kategorizirati po značenju. Kako primjer možemo uzeti ključnu riječ „mačka“ koju čovjek odmah prepoznaje kao životinju dok je to za računalo još samo jedna ključna riječ.

Semantički web je zamišljen kao nadogradnja na postojeći kako bi se omogućio daljnji razvoj Web servisa. Od tehnologija, za semantički web se kao građevni element koriste ontologije. U ontologijama se znanje formalizira na način da se podacima dodaje još informacija – svojstva, pripadnost nekog domeni, odnos s drugim pojmovima.

Vizualizacija znanja se temeljno odnosi na vizualizaciju podataka i informacija zapisanih u nekoj ontologiji. Cilj je prikazati odnos između povezanih pojmova te korisniku omogućiti određenu interakciju da može istražiti koji su pojmovi povezani ukoliko je ontologija velika i možebitno povezana s drugim ontologijama. Upravo to povezivanje ontologija omogućuje izgradnju baze znanja.

[1] W3C - World Wide Web Consortium
[2] HTML - Hypertext Markup Language

Konkretno, u ovom radu bit će predstavljena bogata Internet aplikacija koja učitava ontologiju semantičkog Weba (zapisanu u OWL formatu datoteke) te parsira njen sadržaj u potrazi za klasama koje će prikazati korisniku kao mrežni dijagram. Pod bogatom Internet aplikacijom se podrazumijeva korištenje određenih tehnologija koje za izvršavanje ne ovise o operacijskom sustavu, a unatoč tome pružaju korisniku mogućnosti *desktop* aplikacije.

Uvedeni pojmovi poput semantički Web, ontologija, bogata Internet aplikacija bit će pobliže objašnjeni u slijedećem poglavlju.

2. Vizualizacija znanja

Prostor u kojem se pojavljuju informacije sve se više povećava. Znanje se (kao informacija) distribuira kroz različite medije, bilo digitalne poput TV-a i Interneta, bilo analogne kao što su novine ili knjige. Taj ogroman bazen informacija i činjenica da je informaciju vrlo lako reproducirati kroz već spomenute medije je velik resurs za čovjeka, međutim, baš zbog povećanja fonda znanja dolazimo do problema njegove neorganiziranosti i kaotičnog nastajanja teško je doći do prave informacije određene problematike.

Najveći problem je što današnjim informacijama nedostaje semantičko strukturiranje i navigacija kroz to obilje informacija.

Prednosti vizualizacije

Prikaz informacija preko tekstualnog sučelja je daleko od optimalnog; vizualizacija informacija se nameće kao intuitivna alternativa. Vizualizacija znači smanjenje i translaciju kompleksnih semantičkih veza u interaktivnu vizualnu mapu.[3] Takva mapa je digitalni rođak nekadašnjih kartografskih mapa gdje je informacija sažeta i strukturirana. Koristeći prednosti digitalnog medija poput obrade podataka, prikaza u 2D ili 3D okruženju, ažuriranja u realnom vremenu, odgovarajuću interaktivno na korisnikove akcije, te mape mogu omogućiti korisniku da se kreće i orijentira kroz informacije.

Vizualizacija iskorištava čovjekovu sposobnog prostornog snalaženje; svojstvo ljudskog uma da izradi mentalne mape fizičkog okruženja i zna se kretati po njima. Prednost prostornog snalaženja je u tome što se određivanje puta i prepreka najvećim djelom odvija na podsvjesnoj razini tako da se, primjerice kod vizualizacije znanja, manje kognitivnog (spoznajnog) kapaciteta mozga koristi za čitanje i razumijevanje teksta ili brojeva jer se informacija vizualno prezentira što je čovjeku mnogo intuitivnije nego prepoznavanje slova i brojeva.

Primjer kartografske mape pokazuje da gotovo svaki čovjek može iščitati informacije s karte ili bar prikupiti osnovne informacije s nje. Vizualizacija znanja je korištenje tradicionalne i prokušane tehnike prikazivanja informacija ali na modernom problemu i mediju.

Izazovi vizualizacije

Prednosti same vizualizacije su neosporive, međutim, postoje i određeni problemi i izazovi koji su specifični za vizualizaciju znanja.

[3] Judelman G.B. (2004). *Knowledge Visualization*. [4]

Kao prvo, kod izrade takve mape treba uzeti u obzir da se kod vizualizacije znanje ne može koristiti samo statička mapa, mora se implementirati i prostorna navigacija, interaktivnost te vremenska dinamika.

Drugo, statičke mape poput kartografskih mapa lakše su za razumijevanje jer se njihov sadržaj odražava na ono što vidimo u prirodi. Mape znanja ne možemo percipirati na isti način jer ne postoji neka fizička analogija tog sadržaja u prirodi. Vizualizacija znanja ili informacija, odnosno semantičkog prostora u kojem se one nalaze, u potpunosti je apstraktni fenomen. Znanje ne možemo *vidjeti* kao što vidimo zemlju ili more te je zbog toga procesiranje takve mape mnogo teže za čovjeka.

Treći je problem to što trenutačno postoje praktična, teoretska i tehnička ograničenja da bi se izgradio efektivni sustav vizualizacije znanja koji bi zadovoljio sve kriterije interaktivnosti, intuitivnosti, ergonomije i slično. Svakodnevnim napretkom tehnologije očekuje se da će ti problemi biti sve manji.

Medij gdje je očito najveća koncentracija informacija je upravo Web. Zbog toga se W3C konzorcij usredotočio na razvoj novih standarda koji će omogućiti da se informacije obogate semantičkim vezama. Jedan od jezika za pohranjivanje semantičkih odnosa je OWL (eng. *Web Ontology Language*), no najprije nešto o semantičkom webu.

2.1 Semantički Web

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.[4] 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.

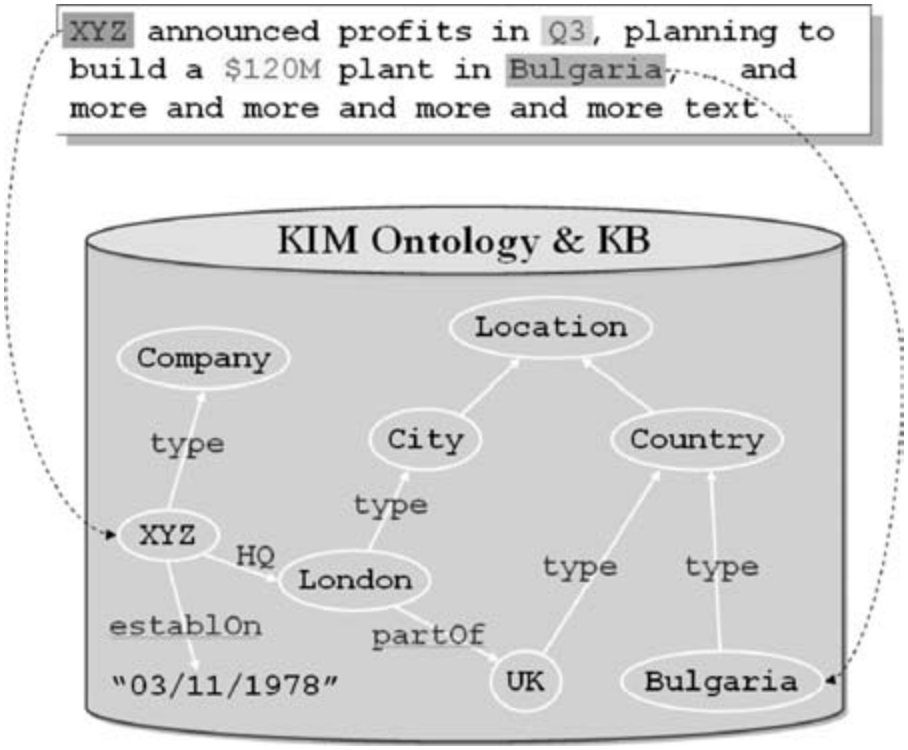
Semantički Web je vizija, to je ideja da se informacije na Web-u definiraju i povezuju na način da mogu biti korištene, sa strane strojeva, ne samo za prikazivanje, već i za automatizaciju, integraciju te ponovnu upotrebu informacija duž kojekakvih aplikacija.

Tehnologije za potporu semantičkog Weba

[4] Gospodnetić L. (2003). *Semantički Web*. [1]

Sam semantički Web koji je zamišljen kao nadogradnja na postojeći Web zahtjeva neke nove tehnologije za izvedbu. Tako se pojavljuju eksplicitni metapodaci, automatska dedukcija, agenti i, za ovaj rad posebno značajne, ontologije.

Slika 2-1[5] Semantičko obilježavanje



Primjer vizualizirane ontologije nalazi se na Slika 2-1 gdje je uspoređen standardni prikaz informacije kao teksta i dio te informacije kao mrežni dijagram.

2.2 Ontologije

Ontologija se često prikazuje kao osnovni građevni element semantičkog Weba. Njihova konkretna uloga je u tome što sadrže dijelove znanja pod određenom domenom i tako doprinose sistematizaciji. Ontologije mogu povećati funkcionalnost Weba na mnogo načina poput poboljšavanja preciznosti u pretraživanju Weba – primjerice pretraživanjem ključnih riječi unutar određenih ontologijskih domena umjesto traženja neodređenih ključnih riječi u bespućima informacija različite kvalitete na Internetu. Očekuje se da će napredne aplikacije koristiti ontologije kako bi povezale informacije na stranicama sa strukturama znanja i pravilima odlučivanja što nam ukazuje na umjetnu inteligenciju kao još jedan od koncepata koji se pojavljuju uz semantički Web.

[5] Preuzeto iz [3], str. 36

Povijest

Ontologija je pojam koji se pojavljuje još u staroj Grčkoj u vrijeme Aristotela kao filozofska disciplina koja se bavi učenjem o bitku i njegovim određenjima.[6]

Značenje ontologije u kontekstu informacijskih znanosti jest reprezentacija entiteta, ideja i događaja zajedno s njihovim svojstvima i odnosima s obzirom na sistem kategorija odnosno domenu. Dok su filozofi većinom raspravljali o mogućim metodama pristupa u izgradnji ontologija bez građenja konkretnih i jasnih ontologija, računalni znanstvenici su izgrađivali masovne ontologije s mnogo manje rasprave kako bi one trebale biti izgrađene.

Iz toga je vidljivo da postoji prostor za interdisciplinarnu suradnju kako bi se zbližili stručnjaci iz dva različita područja.

2.2.1 OWL – Web Ontology Language

Web Ontology Language pripada skupini jezika za reprezentaciju znanja kojima se stvaraju ontologije. OWL je specifičan po tome što ga preporučuje WWW konzorcij (W3C), što ga predstavlja kao standard za izradu ontologija na Webu. Zbog te podrške, OWL se smatra fundamentalnom tehnologijom nadolazećeg semantičkog Weba. Također, OWL je dobro prihvaćen u akademskoj zajednici ali je privukao i kompanije koje vide komercijalan interes u budućem razvoju OWL jezika.

Pojam **reprezentacija znanja** dolazi s područja umjetne inteligencije i bavi se pitanjem kako formalizirati razmišljanje, odnosno kako koristeći sustav simbola predstaviti „domenu razgovora“ – možemo reći i temu razgovora tako da se omogući da se objekti iz razgovora mogu prepoznati i povezati s nekim drugim temama (domenama). Radi se o tome da se upotrijebi logika zajedno s pripadnim operatorima kako bi se razgovor mogao pretočiti u semantički jezik. Razvoju ovog područja pomažu istraživanja o tome kako ljudski mozak sprema i manipulira jezikom, zvukovima, doživljajima, mirisima, emocijama, procedurama, apstraktnim idejama. Na mnoga tako napredna pitanja još nemamo odgovore, no ontologijama se pokušavaju riješiti problemi koje možemo pobliže formalizirati. Tako se za potrebe semantičkog Weba razvio OWL jezik koji koristi deskriptivnu logiku za reprezentaciju znanja.

Glavne značajke

Glavnu arhitekturu jezika čine dokazane Web tehnologije – XML (eng. *eXtensible markup language*), URI (eng. u*niform resource identifier*) i RDF (eng. r*esource description framework*).

[6] *Ontologija u informacijskim znanostima.* [2]

Navedene tehnologije su redom razvijene od strane W3C konzorcija. OWL jezik ima mogućnost kreiranja klasa, svojstva, instanci i operatora nad skupom.

Postoji glavna, korijenska **klasa** `owl:Thing`. Klase su organizirane u podatkovnu strukturu u obliku stabla. Svaka klasa može imati neograničen broj podklasa čime se stvara spomenuta struktura. One mogu imati i različita svojstva koja su definirana OWL specifikacijom. Klasa se definira na slijedeći način[7]:

```
<owl:Class rdf:about="Wine"/>
```

Uz klase, postoje i njihovi članovi, individue odnosno **instance** klasa. Definiraju se na slijedeći način:

```
<Wine rdf:about="Cabernet"/>
<Wine rdf:about="Suavignon"/>
```

Instanca se definira kao XML čvor imena pripadne klase te nazivom instance kao XML atributom rdf:ID.

Primjer klase s nekoliko **svojstava**. Definirana je nadklasa te dvije ekvivalentne klase. Tu se pojavljuje i operator unija jer postoji više (skup) ekvivalentnih klasa:

```
<owl:Class rdf:about="#Fruit">
   <rdfs:subClassOf rdf:resource="#Dessert"/>
   <owl:equivalentClass>
        <owl:Class>
             <owl:unionOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#NonSweetFruit"/>
                  <rdf:Description rdf:about="#SweetFruit"/>
             </owl:unionOf>
        </owl:Class>
   </owl:equivalentClass>
</owl:Class>
```

Svojstva opisuju karakteristike klase. Postoje dvije glavne vrste svojstava koja su definirana u OWL jeziku: svojstva tipa podataka (eng. *datatype properties*) i svojstva objekta (eng. *object properties*).

Svojstvo tipa podataka definira veze između instanci klasa, RDF naputaka i XML-om definiranih tipova podataka, dok se svojstvo objekata odnosi na relaciju između instanci dviju klasa.

[7] Primjer ontologije preuzet s http://www.w3.org/TR/owl-guide/wine.rdf, 5.9.2009.

Od **operatora**, uvedena su tri operatora nad skupom:

- Unija

```
<owl:unionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#SweetFruit" />
</owl:unionOf>
```

- Presjek

```
<owl:intersectionOf rdf:parseType="Collection">
</owl:unionOf>
```

- Komplement

```
<owl:complementOf rdf:resource="#Fruit"/>
```

Njihova namjena je manipulacija ekstenzijama klasa; to su zapravo skupovi individua odnosno instanci i podklasa.

2.3 Bogate internet aplikacije

Kroz povijest bilo je snažnih prevrata u računalnoj industriji. Jedan takav događaj bio je napuštanje sistema velikih centralnih računala i „glupih“ terminala te uvođenje klijent/server aplikacija i sustava. Korisnici su bili na dobitku s produktivnijim računalima, međutim, nekako se sustav opet mijenja jer se razvojem Web industrije sustav klijent/server doima nespretan i glomazan. Pogotovo po pitanju održavanja, nadogradnje i pružanja potpore. Razvoj procesorske snage kreće se u smjeru paralelnog izvršavanja više jezgri što u softverskom smislu povlači prilagodbu programa na višedretveni rad. Višedretvenost također ide više u prilog Web okruženju nego klasičnom desktop okruženju gdje se softver sporije prilagođava.

Teško je zamisliti da će jedan koncept u potpunosti zamijeniti drugi ali možemo očekivati da će se ipak glavnina procesiranja obavljati negdje drugdje, na nekoj „frami“ servera, te da korisnik neće imati muke sa nadogradnjom softvera ili hardvera. Taj novi klijent naziva se i slabi klijent ili tanak klijent. Nove aplikacije koje nam stižu pod zajedničkim nazivom RIA (eng. *rich Internet applications* – bogate Internet aplikacije) donose neke praktične prednosti nad desktop aplikacijama poput jednostavnog nadograđivanja, neovisnost o operacijskom sustavu, interakciju s drugim korisnicima, promjene u realnom vremenu.

Kod „obične“ internetskih aplikacija korisnik za svaku akciju mora ponovno učitati web stranicu i svaki korak ide sporo jer se kompletno sučelje (web stranica) aplikacije svaki puta ponovno preuzima od servera. Bogate Internet aplikacije su stvarne aplikacije koje su pokrenute na klijentskom računalu, a sa serverom se obavlja samo prijenos i procesiranje podataka. Time se desktop doživljaj prenosi na Web uz sve prednosti Interneta.

Pojam RIA prvi put je uvela kompanija Macromedia (danas dio Adobea) 2002. godine. Prve Internet aplikacije pojavile su se još davne 1995. godine kao Java *appleti*, mali java programi koji su bili pogonjeni JVM[8] dodatkom u web pregledniku.

2.3.1 RIA platforme

Postoji nekoliko platformi koje su danas u najširoj upotrebi.[9] To su tehnologije odnosno proizvodi koje se izvršavaju u web pregledniku, donekle se oslanjajući i na gostujući operacijski sustav.

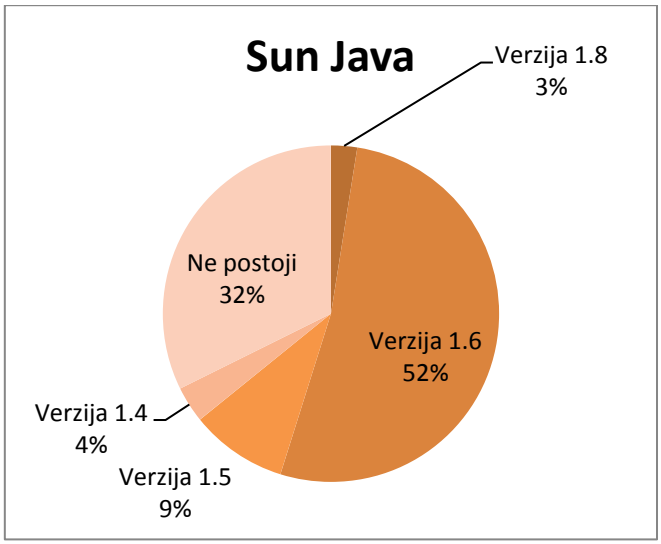
Sun Java

Iako je Java programski jezik postao popularan primarno zbog java appleta (Slika 2-2), appleti ipak nisu postali mjesto glavne primjene Java tehnologije. Primarni razlog tome je prilično visoki zahtjevi za sistemskim resursima od strane Java virtualnog stroja te sama veličina instalacije (oko 16 megabajta). Postoje i drugi nedostaci u odnosu na druge platforme. Nedostatak gotovih UI klasa koje su lijepo dizajnirane, vrlo slaba podrška za zvuk i video. Java API ima podržane samo rudimentarne multimedijske funkcije. Za razliku od Jave, Adobe Flash je dobio na popularnosti upravo zbog svoje jednostavnosti, male instalacije, gotovih dobro dizajniranih UI klasa.

Iako je Java vrlo moćna platforma njezin problem je jednostavnost same izrade aplikacija tako da je za ozbiljan projekt u Javi potrebno angažirati ekspertne programere što poskupljuje izradu iako su u startu troškovi mali jer su svi razvojni alati besplatno dostupni.

[8] JVM – Java Virtual Machine
[9] Fain Y., Rasputnis V., Tartakovsky A. (2007). *Rich Internet Applications with Adobe Flex and Java*. [5]

Slika 2-2[10] Prisutnost Java virtualnog stroja kod korisnika



Microsoft WPF

Microsoftova *Windows Presenation Foundation* ili, skraćeno akronimom – WPF, platforma za izradu bogatih Internet aplikacija pojavila se na tržištu istovremeno s predstavljanjem Windows Vista operacijskog sustava. Koristi XAML deklarativni jezik temeljen na XML standardu za izradu GUI-ja te C# kao programski jezik za poslovnu logiku. WPF je pogodan za izradu web (RIA) aplikacija kao i za izradu desktop aplikacija. WPF je podržan samo na Windows platformi i to samo u Internet Explorer pregledniku što prilično ograničava primjenu ove tehnologije. Izdavanjem .NET frameworka 3.5 SP1 Microsoft je omogućio pokretanje takvih aplikacija i u Mozilla Firefox pregledniku, putem posebne ekstenzije. Zanimljivo je to što se takva aplikacije zapravo ne pokreće u samom pregledniku nego se izvršava u novom procesu *izvan* preglednika što donekle odstupa od same ideje bogate Internet aplikacije.

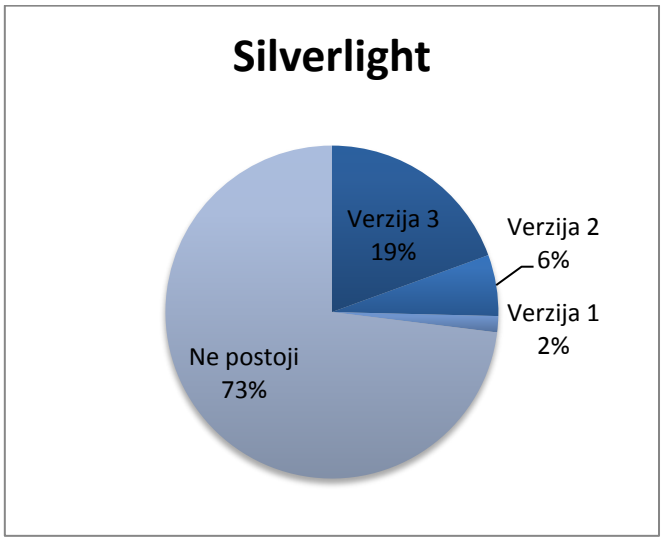
Microsoft Silverlight

Silverlight je, ovog puta ipak, istinska platforma za RIA. Već u kodnom imenu tehnologije (WPF/E) vidimo povezanost sa maloprije predstavljenom Microsoftovom tehnologijom WPF. Iako ih povezuje samo XAML prezentacijski sloj, Silverlight je bio najavljen kao WPF prilagođen za rad na većini platformi. Podržava bogate multimedijske funkcije poput videa, vektorske grafike i animacija i to na različitim operacijskim sustavima (Windows, Mac OS, podrška za Linux je još uvijek u pripremi) kao i unutar različitih web preglednika (Internet

[10] Izvor: http://www.riastats.com/, dostupno 5.9.2009.

Explorer, Mozilla Firefox, Google Chrome, Apple Safari, Opera). Također, u planu je i podrška za mobilne platforme te se očekuje podrška za Windows Mobile i Symbian OS u 2010. godini.

Slika 2-3[11] Prisutnost Silverlight tehnologije kod korisnika



S zadnjom verzijom Silverlight 3 (rujan 2009.), Silverlight je dodatno napredovao te omogućuje pokretanje izvan web preglednika te nove kontrole, klase i funkcije.[12]

Silverlight je vrlo perspektivna platforma jer postoji velika baza programera kojima je Windows razvojno okruženje dobro poznato tako da im prelazak na novu tehnologiju ne predstavlja velik problem jer se Silverlight može ravijati u bilo kojem .NET okruženju. Također, rasprostranjenost raste svakim danom (Slika 2-3).

Glavna zamjerka Silverlightu jer njegova zatvorenost i ne korištenje preporučenih web standard u izradi (ne korištenje SVG vektorskog formata jer primjer). No, i na tome problemu se radi jer se (djelomično) neovisno o Microsoftu razvija platforma otvorenog koda Monolight koja bi trebala zadovoljavati Silverlight specifikaciju 2 i dovesti Silverlight podršku na Linux.

AJAX

Iako AJAX nije platforma, nego skup povezanih tehnologija, ne može se izbjeći kod obrade teme poput ove. AJAX znači asinkroni JavaScript i XML te se kao termin pojavljuje 2005. godine i ubrzo „osvaja“ svijet. Iako tehnika asinkronog učitavanja dijela web stranice postoji još od 1999. godine putem Microsoftove verzije XmlHttpRequest funkcije ili nekih alternativnih tehnika poput iframeova, njegova masovna popularizacija kreće s implementacijom XmlHttpRequest

[11] Izvor: http://www.riastats.com/, dostupno 5.9.2009.
[12] Izvor: http://en.wikipedia.org/wiki/Microsoft_Silverlight#Silverlight_3, dostupno 8.9.2009.

funkcije od strane Mozile. Ta funkcija omogućava sinkronu i asinkroni komunikaciju skripte unutar web stranice i samog web servera, odnosno, koda koji se izvršava na serveru. Glavni problem AJAX-a jer njegova tehnička nedorečenost jer ne postoji AJAX tehnički standard. I dok ostale platforme koriste vlastita okruženja kroz virtualne strojeve, AJAX je implementiran u svakom pregledniku na drugačiji način. Taj problem donekle rješavaju AJAX biblioteke, no to je samo dio poteškoća koje se pojavljuju prilikom razvoja aplikacija u AJAX okruženju. Činjenica je da je JavaScript izvorni kod nemoguće zaštititi i sakriti od konkurencije, ako razvijamo poslovnu aplikaciju. Također, kod koji stiže na taj način do preglednika znači i vrlo sporu obradu jer se kod mora prevoditi za vrijeme učitavanja stranice i to svaki puta kad se stranica ponovno učita.

Prednost AJAX-a je otvorenost i besplatnost korištenih tehnologija – JavaScripta i XML-a. Naravno, tu je neizbježan HTML, CSS i neki jezik koji se izvršava na serveru (PHP, ASP). Sve to zajedno, iako su tehnologije besplatne, uvelike otežava razvoj RIA aplikacija koristeći samo AJAX platformu. Ne postoji integrirano razvojno okruženje za AJAX koje bi obuhvaćalo sve potrebno za razvoj napredne RIA temeljene na AJAX-u (komunikacijski sloj – kolaboracija i mehanizmi udaljene suradnje, HTTP *sniffer*, biblioteka UI kontrola s pratećim objektima, vizualni IDE koji omogućuje upravljanje dizajnom, *debugger* (hrv. razbubnik) za sve te tehnologije, podrška za lokalizaciju okruženja, pristupačnost za ljude s poteškoćama (vid), alati za automatsko testiranje.

Svi ti vrlo ozbiljni nedostaci postavljaju AJAX kao manje atraktivan izbor u odnosu na ostale tehnologije dosad predstavljene.

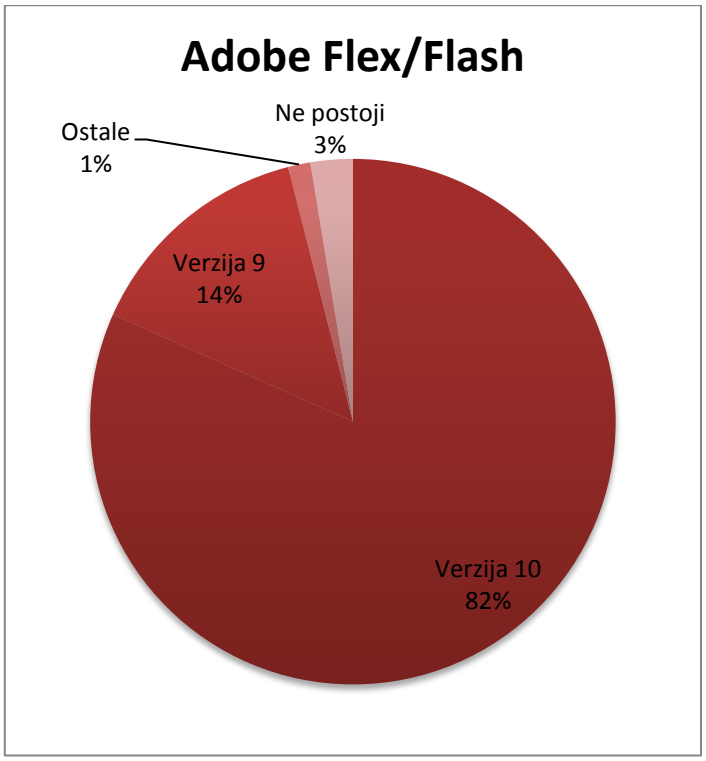
OpenLaszlo

Laszlo Systems je razvio OpenLaszlo kao proizvod otvorenog koda koji omogućuje razvoj bogatih Internet aplikacija u posebnom razvojnom okruženju te pruža mogućnost izdavanja aplikacije kao Adobe Flash datoteke ili kao DHTML kod. Potonja mogućnost je OpenLaszlo stavila u povoljan položaj u konkurenciji alata koji mogu izdati RIA za mobilne platforme. Slično Microsoftovom WPF-u, i OpenLaszlo koristi za GUI poseban jezik temeljen na XML-u nazvan LZX. Za procesnu logiku brine se JavaScript.

2.3.2 Adobe Flex/Flash

Osim što je odabrana tehnologija u ovom projektu, Adobe Flex je i najraširenija platforma za izradu bogatih Internet aplikacija. Danas je ona prisutna u gotovo svakom pregledniku kao što je prikazano na Slika 2-4.

Slika 2-4[13] Penetracija Flash Playera među korisnicima



Flex aplikacije pokreću se u sveprisutnom Flash Playeru koji je u stvari virtualni stroj male veličine. Platforma se sastoji od jezika zaduženog za GUI nazvanog MXML, a baziranog također na XML tehnologiji što je slučaj u kod Silverlighta te OpenLaszloa.

Zatim, za procesnu logiku koristi **ActiveScript 3.0**, objektno orijentirani programski jezik vrlo sličan Javi, temeljen na ECMAScript specifikaciji. U tom je programskom jeziku izvedena kompletna poslovna logika izrađene aplikacije o čemu će biti detalja u slijedećem poglavlju.

Od ostalih sastojaka platforme valja spomenuti integraciju sa serverom preko tehnologije Flex Data Services (FDS) koji klijentskoj aplikaciji omogućuje komunikaciju sa J2EE[14] aplikacijama, komponente za crtanje grafova, pristup multimedijskim kontrolama, Eclipse bazino IDE sučelje.

Iako je većina razvojnih alata za Flex/Flash okruženje besplatna, Flex Data Services se isključivo može koristiti kao plaćena usluga što umanjuje privlačnost same platforme, no popularnost Flash Playera donosi ozbiljnu prevagu u korist Adobeovog rješenja za razvoj bogatih Internet aplikacija.

[13] Izvor: http://www.riastats.com/, dostupno 5.9.2009.
[14] Java Platform Enterprise Edition – platforma za programiranje na strani poslužitelja

3. Izrada aplikacije – Vizualizacija znanja na Webu

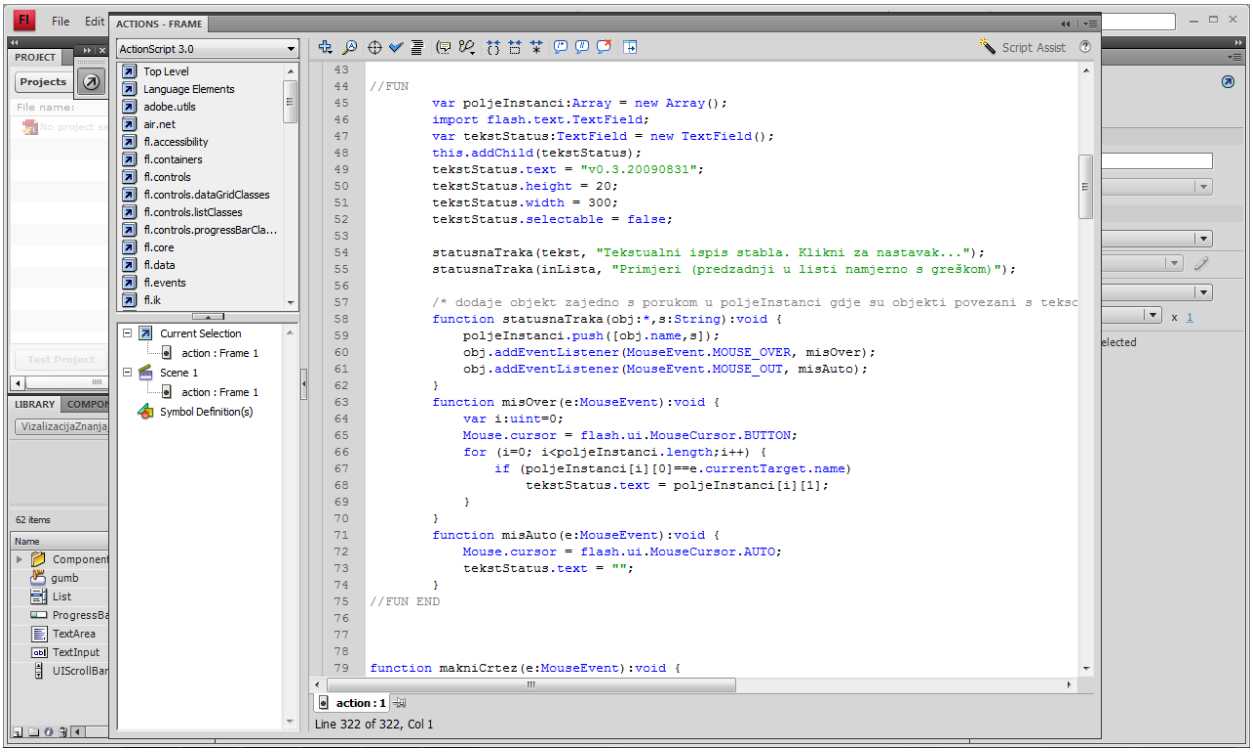
Zadatak je bio izraditi bogatu Internet aplikaciju putem nekih od dostupnih tehnologija koja će biti u stanju učitati neku ontologiju semantičkog weba i prikazati je u obliku mrežnog grafikona. Preporuka je bila korištenje OWL ontologija pošto su one u najširoj upotrebi i XML temeljene jer većina platformi za izradu bogatih Internet aplikacija podržava manipulaciju XML datotekama. Korisniku je potrebno omogućiti dodavanje drugih ontologija u vizualizaciju što povlači odgovarajuće grafičko sučelje.

Kod izbora platforme odlučio sam se za Adobe Flash tehnologiju zbog ActionScript 3.0 programskog jezika koji je zadužen za procesnu logiku. Druga opcija bio je Microsoftov Silverlight, međutim, veća dostupnost literature i veći broj gotovih klasa kao i podržavanje većeg broja operacijskih sustava pridonio je izboru.

3.1 Razvojno okruženje

Za IDE odabrao sam Adobe Flash CS4 (nekad Macromedia Flash) ispred Adobe Flex 3 okruženja jer sam imao iskustva s radom u tom alatu (Slika 3-1).

Slika 3-1 Adobe Flash CS4 razvojno okruženje



Alat je nastao 1995. godine te je danas najraširenija platforma za dodavanje interaktivnosti Web stranicama. Iako izvorno namijenjen izradi animacija kao dodataka web stranicama, u novije vrijeme koristi se i kao RIA razvojno okruženje.

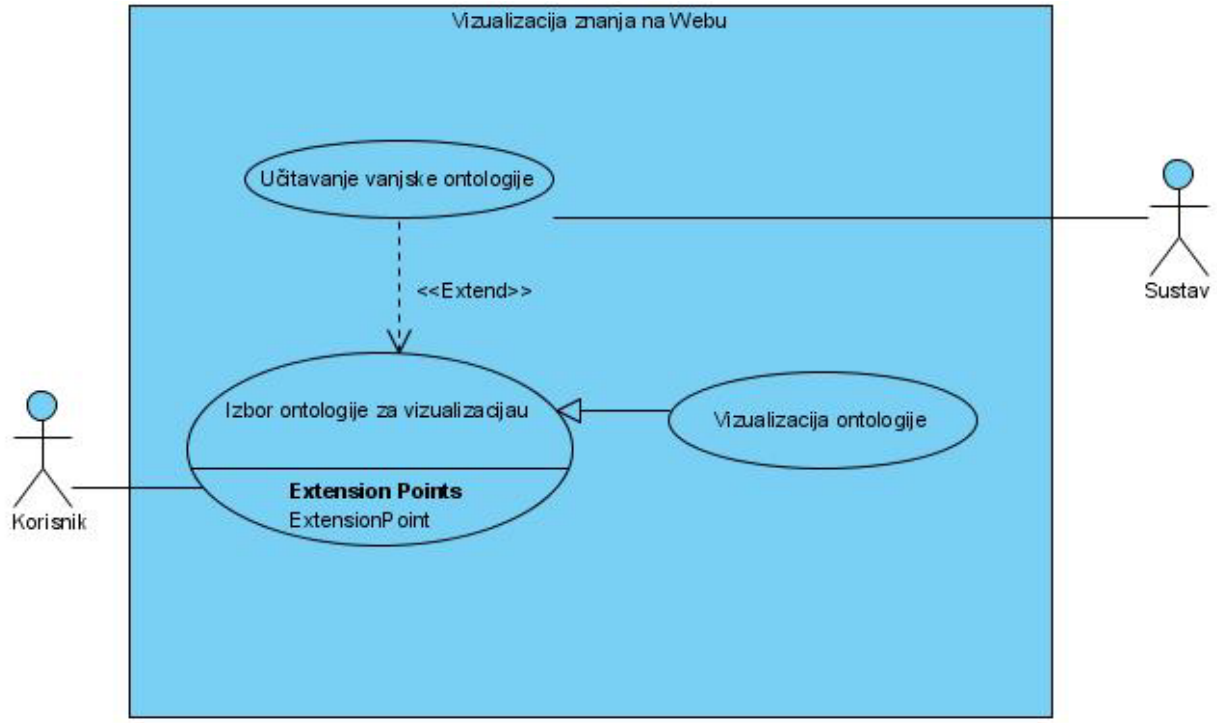
ActionScript 3

ActionScript je gotovo u potpunosti novi jezik u usporedbi s ActionScriptom verzije 2. Zahtjeva novi virtualni stroj u usporedbi s prethodnom verzijom. Donosi organizaciju koda preko paketa (kao što je to izvedeno u Javi), poboljšanu objektnu orijentaciju (klasno nasljeđivanje), podršku za imenski prostor (eng. *namespace*), regularne izraze.

3.2 Dijagram slučajeva korištenja

Dijagramom na Slika 3-2 je prikazan je slučaj korištenja aplikacije na najvišoj razini. Sama aplikacija se sastoji od dva glavna djela. Prvi dio je parsiranje OWL datoteke i spremanje u podatkovnu strukturu, a drugi dio je vizualizacija podataka spremljenih u strukturu. Tako je za korisnika određeno samo da upiše naziv ontologije koja se nalazi u istom direktoriju kao i aplikacija ili da upiše punu putanju do ontologije (ukoliko se nalazi u istoj sigurnosnoj domeni).

Slika 3-2 Use case dijagram



Interakcija sustava i aplikacije događa se samo kada je potrebno dohvatiti neku vanjsku ontologiju s druge domene.

3.3 Isječci izvornog koda

Cijela aplikacija zajedno sa elementima korisničkog sučelja je instancirana kroz ActionScript, uz poneku iznimku. Skriptni kod se veže uz jedan kadar (eng. *frame*) jer Flash aplikacije su koncipirane kao film sa sličicama zbog svoje primarne namjene izrade animacija. Instanciranje objekata na scenu se izvodi preko objekta `this` i metode `addChild(referenca)`. Scena je glavni dio Flash aplikacija gdje se nalazi trenutno aktivan kadar. Slijedi primjer instanciranja UI objekta koji ispisuje tekst korisniku:

```
import fl.controls.*;
var tekst:TextArea = new TextArea();
tekst.visible=false;
tekst.alpha = 0;
tekst.editable = false;
tekst.textField.selectable = false;
this.addChild(tekst);
```

Ključnom riječi “`import pathTo.packageName.as`“ vanjski paketi i klase postaju dostupni unutar postojećeg koda.

3.3.1 Učitavanje OWL datoteke

Za učitavanje OWL datoteke napravljena je funkcija koja kao parametar prima URL putanju do OWL datoteke. Funkciju pokreću dva događaja (`event`). Jedan je klik na gumb „crtaj“, a drugi je odabir neke od primjera ontologije na početnom ekranu aplikacije.

```
function ucitajPodatke(url:String):void {
   tekst.text="";
   pokaziTekstArea();
   if (url != "[URL]") {
        var xmlLoader:URLLoader = new URLLoader();
        xmlLoader.load(new URLRequest(url));
        xmlLoader.addEventListener(Event.COMPLETE, ucitajXML);
   }
}
```

Funkcija `pokaziTestArea()` služi za pripremu tekstualnog okvira u kojem se ispisuju podaci o učitanom ontologiji.

```
function pokaziTekstArea():void {
   tekst.visible = true;
```

```
   TweenLite.to(tekst,0.5,{alpha:1});
   var stil:TextFormat=new TextFormat("Courier New",13);
   tekst.setStyle("textFormat", stil);
   tekst.wordWrap=true;
   tekst.move(20,20);
   tekst.setSize(660,460);
   tekst.addEventListener(MouseEvent.CLICK, makniTekstArea);
}
```

Ovo je i nastavak primjera kako su UI elementi instancirani i manipulirani iz koda, što se u ranijim verzijama Flasha i ActionScripta nije moglo.

Objekt `xmlLoader` je instanciran iz klase `URLLoader` koje ja dio standardne ActionScript 3 biblioteke i vrši posao preuzimanja vanjskih podataka u aplikaciju. Kada se datoteka uspješno učita, pokreće se funkcija `ucitajXML()`:

```
function ucitajXML(e:Event):void {
   xmlData=new XML(e.target.data);
   if (tekst.visible==true) {
        Parsiraj(xmlData);
        drvo.prebaciPoljeStablo(poljeParovaKlasa);
        if (poljeParovaKlasa.length != 0) {
             ispis(drvo.ispis());
             ispis("Broj klasa: ",drvo.korijen.size);
        }
        //ponovno inicijaliziraj
        poljeParovaKlasa=poljeParovaKlasa.slice(0,0);
        drvo = new Stablo("things");
        dat_ok = false;
   }
}
```

Ova funkcija pokreće slijedeću glavnu funkciju u programu: `Parsiraj(xmlData)` gdje se kao argument prosljeđuje XML objekt, također dio standardne biblioteke za manipulaciju s XML podacima. Objekt se kreira prosljeđivanjem sadržaja OWL datoteke XML konstruktoru što je prikazano u drugoj liniji prethodnog isječka koda.

```
function Parsiraj(owlInput:XML):void {
 var k,i:int=owlInput.children().length();
 var nadklasa:String;
 var j:int=0;
```

Slijedeći isječak provjerava da li je format datoteke zadovoljen, odnosno, da li je datoteka OWL ontologija tipa RDF/XML:

```
while (!dat_ok && j<i) {
  if
((owlInput.children()[j].localName().toString()=="Ontology")
&& (dubina==0)) {
   dat_ok = true;
   ispis("OWL (RDF/XML)");
  }
  j++;
 }
```

Početak parsiranja. Parser prolazi kroz cijelu XML strukturu OWL dokumenta i traži klase. Ako klasa nema djece onda je veže za korijenski element owl:Thing.

```
 if (dat_ok) {
  for (j=0; j<i; j++) {
   nadklasa="";//resetiranje varijable
   switch (owlInput.children()[j].localName().toString()) {
    case "Class":

     //provjerava da li je klasa bezimena
     if (owlInput.children()[j].attributes().toString()=="")
      break;//bezimena

     //ako nema djece onda je nadklasa THING
     if (owlInput.children()[j].children().length()==0) {
      if (dubina==0) nadklasa=drvo.korijenElement;//Thing
     }
     else {

      //zbog prethodnog uvjeta znamo da ima djece
      for (k=0; k<owlInput.children()[j].children().length(); k++) {
     //da li je djete subClassOf, jer ako je onda je to naziv nadklase
     if (owlInput.children()[j].children()[k].localName().toString()=="subClassOf") {
      nadklasa=owlInput.children()[j].children()[k].attributes().toString();
      if (nadklasa=="" && owlInput.children()[j].children()[k].children().length()!=0)
       nadklasa=owlInput.children()[j].children()[k].children()[0].attributes().toString();
       if (nadklasa=="") nadklasa=drvo.korijenElement;//Thing     }  }  }
```

U ovom trenutku, na kraju case-a koji se izvršava kad dođe do elementa `owl:Class`, poziva se funkcija `dodajKlasu()` koja za parametre prima dva naziva klasa (string): prvo trenutnu klasu (naziv u atributu trenutnog elementa u XML strukturi) te zatim njezinu nadklasu čiji je naziv pronađen prethodnik provjerama.

```
     dodajKlasu(owlInput.children()[j].attributes().toString(),nadklasa);
     break;
```

U nastavku funkcije za parsiranje obrađuju se vanjske klase koje se koriste u dokumentu, a povezane su sa određenim objektima unutar OWL-a. Iako bi se te klase mogle i ignorirati (jer se nalaze u nekoj drugoj ontologiji), odlučio sam se i za njihov pronalazak jer se u programima slične namjene one također prepoznaju (primjer: OwlViz dodatak za Protegé).

```
    case "ObjectProperty": //isto se trazi klasa!
     //ako je klasa bez djece onda moze biti definirana na khm khm, aa cudan
nacin!! primjer: person.owl klasa: PersonProjectAssociation
     for (k=0; k<owlInput.children()[j].children().length(); k++) {
      if (owlInput.children()[j].children()[k].namespace().prefix=="rdfs" &&
          owlInput.children()[j].children()[k].localName().toString()=="range"){
       if (dodajKlasu(owlInput.children()[j].children()[k].attributes().toString(),
                      drvo.korijenElement))/*Thing*/ { trace("klasa dodana");
       }
      }
     }
     //trace();
     break;
    default:
     break;
   }//switch end
  }//for end
 }
}
```

Nakon izvršavanja funkcije `Parsiraj(xmlData)`varijabla `poljeParovaKlasa` sadrži polje sa povezanim klasama i njihovim nadklasama (roditeljima u strukturi stabla). No, za popunjavanje varijable `poljeParovaKlasa` zadužena je slijedeća funkcija kojoj podatke predaje parser, `dodajKlasu()` :

```
function dodajKlasu(klasa:String,nadKlasa:String, ...
ostalo:*):Boolean {
```

```
    var klasaDodana:Boolean=false;
    var i:uint=0;
    klasa=stripString(klasa,"#");
    klasa=stripString(klasa,";");
    nadKlasa=stripString(nadKlasa,"#");
    nadKlasa=stripString(nadKlasa,";");
    var pomocnoPolje:Array = new Array(klasa,nadKlasa);
    if (drvo.stabloGaVecIma(klasa,nadKlasa)) {
         trace("duplikat!");
    }
    else {
         poljeParovaKlasa.push(pomocnoPolje);
         klasaDodana=true;
    }
    return klasaDodana;
}
```

Ovdje je gotova poslovna logika prvog dijela programa. Rezultat je varijabla `poljeParovaKlasa` koja ima pripremljene podatke za klasu Stablo.

3.3.2 Obrada uvezenih podataka

Stablo je klasa koja proširuje klasu TreeNode[15]. TreeNode je klasa koja daje podršku za strukture podataka (stablo, red, liste). Klasu sam proširio dodatnim funkcijama potrebnim za popunjavanje stabla, provjere te ispis.

```
package
{
    import de.polygonal.ds.TreeIterator;
    import de.polygonal.ds.TreeNode;
    import de.polygonal.ds.DLinkedList;
    import flash.display.DisplayObject;
    import flash.display.DisplayObjectContainer;
    import flash.display.Sprite;

    public class Stablo extends TreeNode {
         private var ploca:Sprite;
         public var korijen:TreeNode;
         public var korijenElement:*;
         //konstruktor
```

[15] AS3 Data Structures For Game Developers, dostupno 20.04.2009. na http://code.google.com/p/as3ds/

```
        public function Stablo(obj:* = "owl:Thing"):void {
             korijen = new TreeNode(obj);
             korijenElement = obj;
             ploca = new Sprite();
        }
```

Objekt ove klase u programu je `drvo` te se nakon izvršenja prvog dijela programa pokreće metoda `prebaciPoljeStablo()` gdje se kao parametar prosljeđuje varijabla `poljeParovaKlasa`. Primjer:

```
drvo.prebaciPoljeStablo(poljeParovaKlasa);
```

Ta metoda za svaki par (klasa,nadklasa) prolazi kroz stablo u potrazi za nadklasom; ako pronađe nadklasu dodaje joj dijete klasa. Tako se gradi struktura stabla. Primjer funkcije:

```
public function prebaciPoljeStablo(a:Array):void {
 var i:int;
 var drvoiterator:TreeIterator = korijen.getTreeIterator();
 var klasa,nadKlasa:String;
 var dodano:Boolean;
 var pomocnoPolje:Array = new Array();
 while (a.length!=0) {
  for (i=0; i<a.length;i++) {
   dodano=false;
   klasa=a[i][0];
   nadKlasa=a[i][1];
   if (!stabloGaVecIma(klasa,nadKlasa)) {
    TreeIterator.preorder(korijen, function(node:TreeNode):void
    {
     if (node.data==nadKlasa) {
       drvoiterator=node.getTreeIterator();
       drvoiterator.appendChild(klasa);
       dodano=true;
     }
    });
    if (!dodano) pomocnoPolje.push([klasa,nadKlasa]);
   }
  }
  a=pomocnoPolje;
  pomocnoPolje=pomocnoPolje.slice(0,0);
 }}
```

Uvijek izlaska iz petlje je da su sve klase dodane u stablo odnosno kad je veličina polja `a` jednaka nuli.

Funkcija `stabloGaVecIma()` je pomoćna funkcija koja za parametar prima par (klasa,nadklasa) i provjerava da li taj par već postoji u strukturi stabla.

3.3.3 Vizualizacija

Sama vizualizacija podataka spremljenih u strukturu stabla ima dvije opcije: tekstualni ispis stabla putem metode `ispis()` u klasi Stablo i grafički ispis putem metode `crtaj()` u istoimenoj klasi.

Metoda `ispis()`:

```
        public function ispis():String {
             return this.korijen.dump();
        }
```

Vraća oblik stabla formatiran u običnom tekstu koji zatim popunjava TextArea objekt.

Metoda `crtaj()`:

```
public function crtaj(obj:*):UIComponent {
   var uiref:UIComponent = new UIComponent;
   poljeY = new Array();
   var maxdubina:uint = dubina();
   for (var i:uint = 0; i<=maxdubina; i++ ) {
        poljeY.push(0);
   }
   //crtanje svih klasa
   TreeIterator.preorder(this.korijen, function(node:TreeNode):void
   {
        uiref.addChild(nacrtajKlasu(node));
   });
   //popravljanje Y pozicije klasa
   TreeIterator.preorder(this.korijen, function(node:TreeNode):void
   {
        var djeca:Array = new Array();
        djeca = node.children.toArray();
        var newY:uint = 0;
        djeca.forEach(function callback(item:TreeNode, index:int,
array:Array):void{
             newY+=item.data[1].y;
```

```
        });
        newY=Math.round( newY/djeca.length );
        //trace(node.data[0] + newY + " deca?"+djeca.length);
        if (newY!=0) node.data[1].y=newY;
        if (djeca.length==1) {
            djeca[0].data[1].y=node.data[1].y;
        }
    });
    //crtanje linija koje povezuju djecu i roditelje
    TreeIterator.preorder(this.korijen, function(node:TreeNode):void
    {
        if (!node.isRoot()) {
            node.data[1].graphics.lineStyle(1,0xFFCC11);
            var x:int = node.data[1].x - node.parent.data[1].x-100;
            var y:int = node.data[1].y - node.parent.data[1].y-22;
            node.data[1].graphics.moveTo(0,22);
            node.data[1].graphics.lineTo(-x,-y);
        }
    });
    //vraća kontejner koji sadrži sve nacrtane klase
    return uiref;
}
```

`TreeIterator.preorder` je metoda koja prolazi kroz stablo prema zadanom ključu. Važna metoda koja se poziva kod crtanja klasa je nacrtajKlasu() koja za argument prima čvor u stablu koji u biti predstavlja klasu. Računa x i y poziciju gdje treba nacrtati elipsu prema položaju klase u stablu. Isječak koda:

```
private function nacrtajKlasu(n:TreeNode):Sprite {
    var elipsa:Sprite = new Sprite;
    var djeca:uint = n.numChildren;
    var razina:uint = n.depth;
    var naziv:TextField = new TextField();
    naziv.text = "\n\t" + n.data[0];

    var x:uint = 140*razina;
    var y:uint = poljeY[razina];
    poljeY[razina]+=50;
    elipsa.x = x;
    elipsa.y = y;
    elipsa.graphics.beginFill(0xFFCCDD);
```

```
    elipsa.graphics.drawEllipse(0,0,100,45);
    elipsa.addChild(naziv);
    //u stablo dodaj referencu na nacrtanu elipsu kako bi child mogao
manipulirati s parentom
    n.data[1]=elipsa;
    return elipsa;//vrati sprite koji se dodaje u UIComponent
}
```

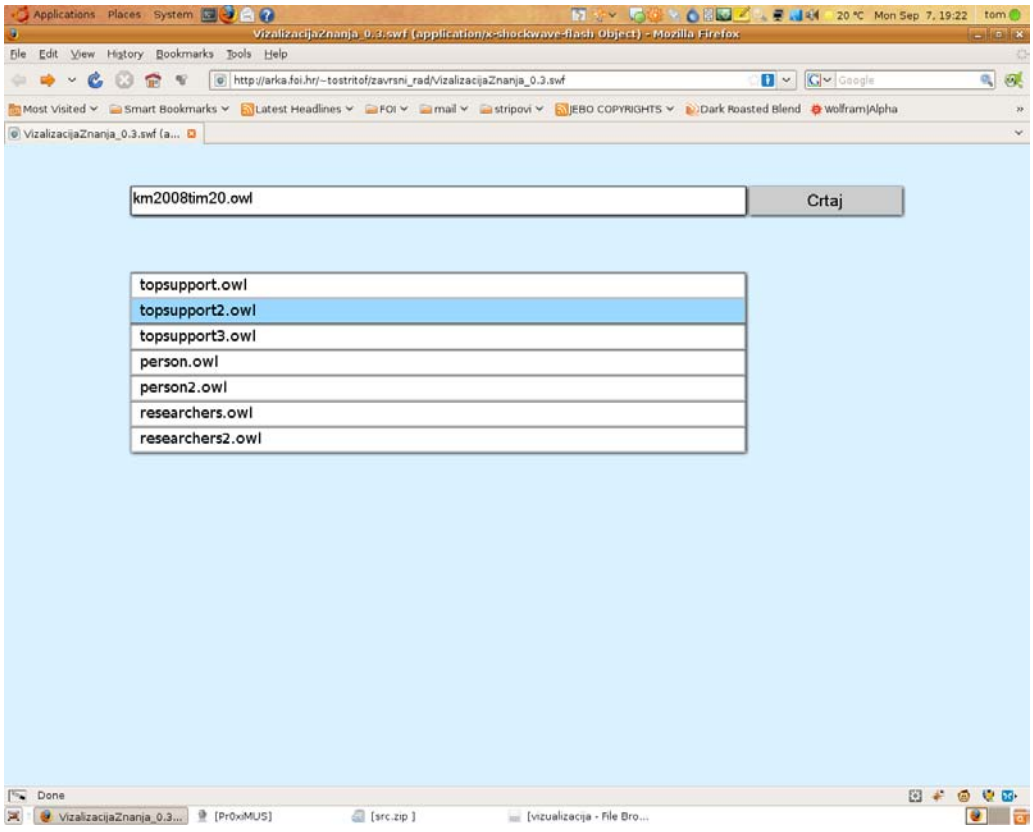
U funkciji se prije izlaska dodaje u stablu referenca na nacrtanu elipsu koja predstavlja klasu. Detaljno: `n.data` je varijabla tipa polje koje na prvom mjestu sadrži ime klase (string), a na drugom mjestu referencu na grafički element elipsu. Tako je prolaskom kroz stablo uvijek moguće naknadno manipulirati s elipsom preko reference što je iskorišteno za kasnije popravljanje Y pozicije klase u vizualizaciji te u zadnjem koraku crtanja linija koje povezuju klase.

3.4 Demonstracija rada aplikacije

Aplikacija podržava rad na više platformi te može raditi i izvan preglednika. U slijedećim redovima biti će slike ekrana aplikacije u različitim sustavima:

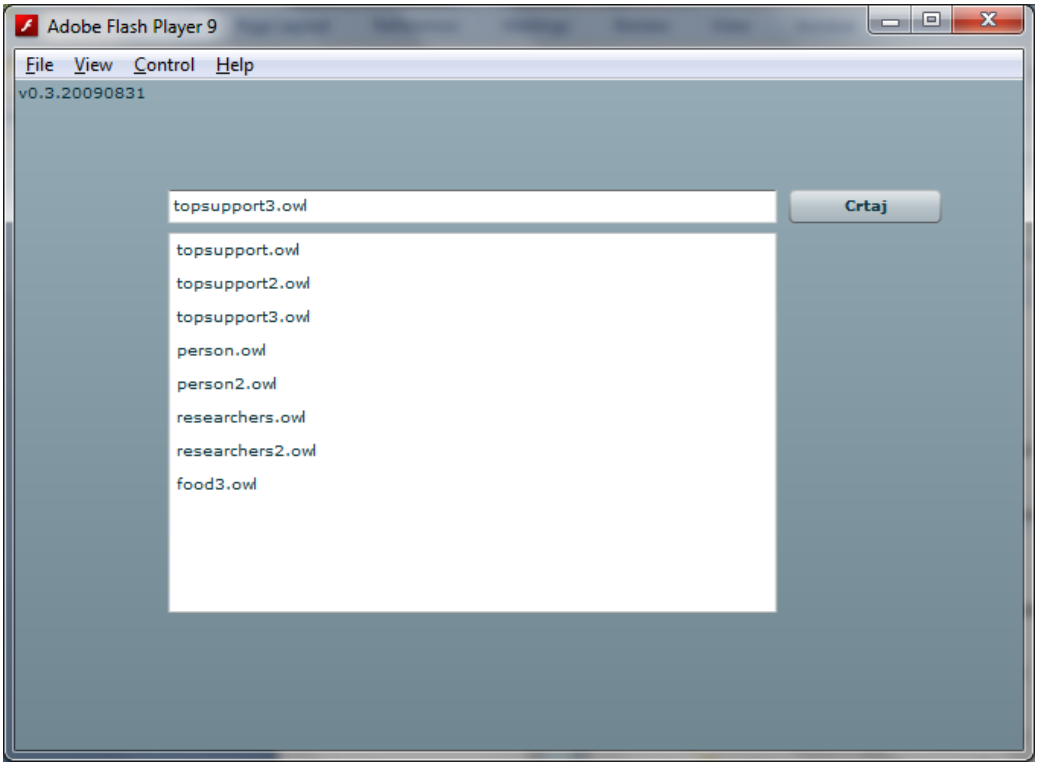
- Rad pod Linuxom (Slika 3-3).
- Početni ekran, Windows okruženje izvan preglednika (Slika 3-4).
- Tekstualni ispis stabla, Windows okruženje unutar preglednika (Slika 3-5).

Slika 3-3 Rad pod Linuxom



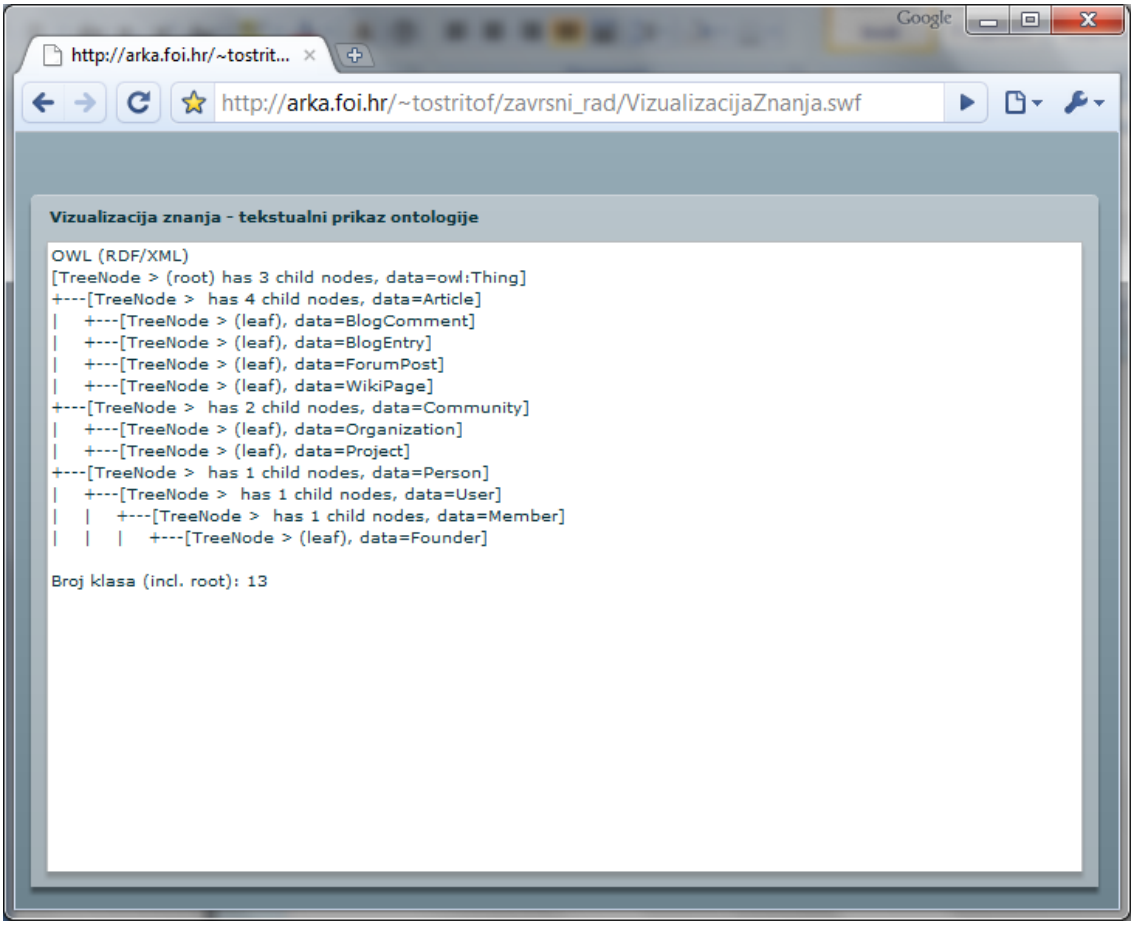
Aplikacija se jednako na svim podržanim operacijskim sustavima, jedino ovisi o virtualnom stroju Flash playera.

Slika 3-4 Početni ekran, Windows okruženje izvan preglednika



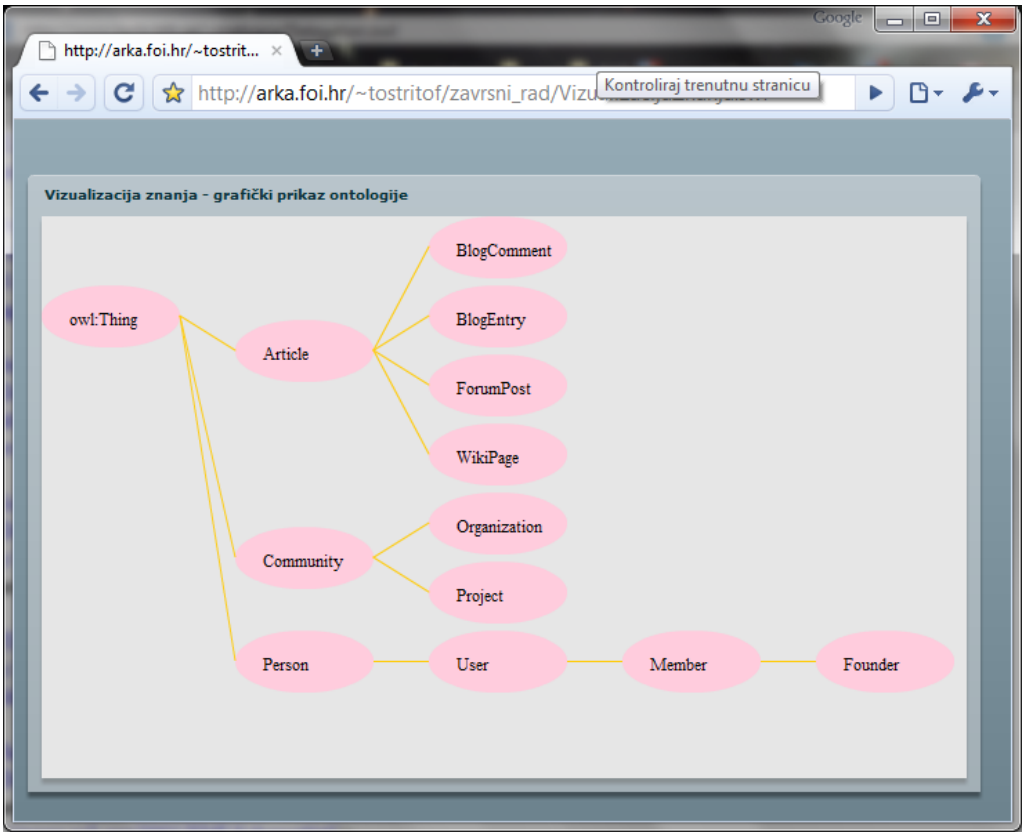
Na slici ispod (Slika 3-5) prikazan je početni ekran koji se sastoji od 3 kontrole: padajuća lista s primjerima ontologija gdje se klikom na primjer pokreće ista akcija kao i klik na gumb 'Crtaj' samo što se u potonjem slučaju koristi putanja do ontologije koja se može ručno upisati. Čim se akcija crtaj pokreće korisnik više nema puno opcija osim pregledavati vizualiziranu ontologiju.

Slika 3-5 Ispis stabla, Windows okruženje, rad iz preglednika



I konačno, na Slika 3-6 prikazana je konkretna vizualizacija znanja što je bio i cilj ovog projekta. Prikazane su klase i njihov međusoban odnos na intuitivan način gdje je lako pronaći povezane pojmove.

Slika 3-6 Vizualizacija znanja



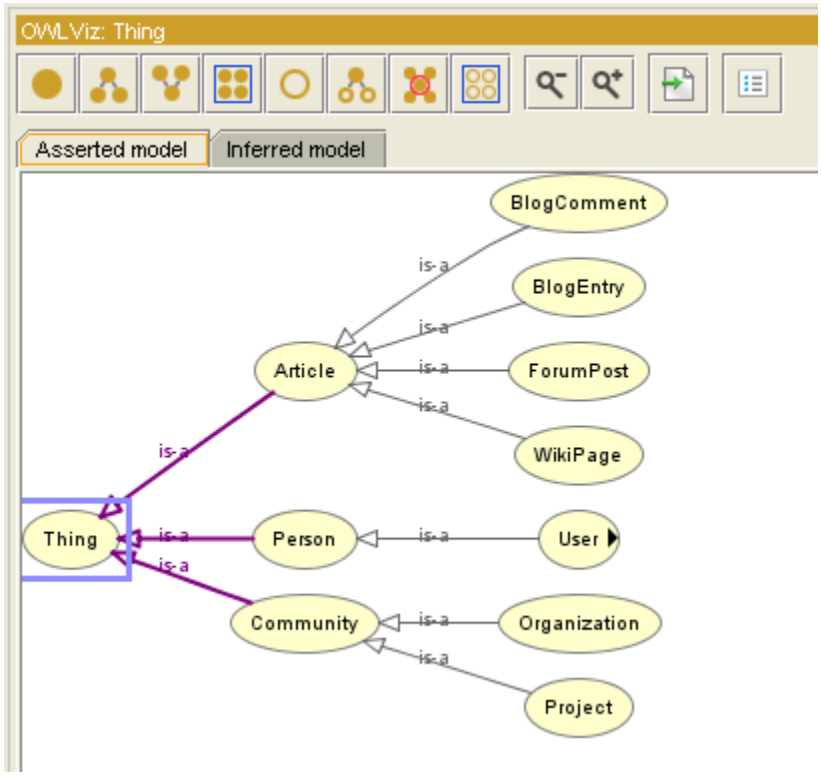
3.5 Alternativna rješenja

Protégé

Postoje različiti alati koji se bavi obradom i prezentacijom sadržaja zapisanih u ontologijama. Najpoznatiji alat je Protégé, razvijen na američkom sveučilištu Stanford u suradnji sa sveučilištem iz Manchestera (Engleska). Protégé je izrađen u Javi i služi kao okosnica za mnoge dodatke. U kontekstu ovog rada, najzanimljiviji dodatak je Protégé-OWL koji donosi podršku za OWL ontologije. Omogućuje učitavanje i spremanje OWL i RDF ontologija, promjenu i vizualizaciju klasa i svojstava te mnoge druge opcije.

Postoje i dodaci za sam Protégé-OWL kao što je OWLViz koji omogućava interaktivan prikaz klasa (Slika 3-7).

Slika 3-7 OWLViz vizalizacija OWL ontologije



Pellet

Pellet je alat otvorenog koda koji omogućava razumijevanje OWL ontologija (OWL *reasoner*).

Ispod se nalazi ispis naredbe `pellet.bat realize topsupport3.owl` primjera koji je korišten i na Slika 3-7 u alatu Protégé te na Slika 3-5 u aplikaciji VizualizacijaZnanja.

```
owl:Thing
    support:Article
       support:BlogComment
       support:BlogEntry
       support:ForumPost
       support:WikiPage
    support:Community
       support:Organization
       support:Project
    support:Person
       support:User
          support:Member - (support:__nibs__, support:emina987)
             support:Founder - (support:markus_schatten)
```

4. Kritički prikaz iskorištene tehnologije

Adobe Flash platforma je danas najpopularniji virtualni stroj koji se izvršava na različitim platformama. Analizu korištene tehnologije u kontekstu vizualizacije znanja možemo promatrati s dva osnovna aspekta; to je kao prvo vizualizacija znanja, a kao drugo web komponenta cijelog projekta.

Po pitanju tehnologije za vizualizaciju znanja mislim da nema drugih prioriteta kod izvedbe osim one da se korisniku pruži najbolji doživljaj uz najmanji utrošak vremena prilikom izrade rješenja. Mislim da je Adobe Flash dobar izbor u ovom slučaju. Problemi na koje sam nailazio najviše su bili uzrokovani mojim slabim poznavanjem odabrane tehnologije te se tu krio najveći izazov. Kao platforma za izradu bogatih Internet aplikacija Flash ne postavlja mnoga ograničenja pred inženjera, osim možda sigurnosnih ograničenja koja su specifična za aplikacije koje su neovisne o operacijskom sustavu te mogu „živjeti“ na Webu. Primjerice, bolja je komunikacija s nekim Web poslužiteljem nego sa podacima na lokalnom disku jer je zbog sigurnosnih razloga, da se ne bi izvršila neka zloćudna sekvenca na računalu domaćinu, onemogućen jednostavan pristup datotekama na lokalnom disku.

S druge strane, u kontekstu Weba kao glavnog distributivnog mjesta bogate Internet aplikacije, postavio bi zahtjev da razvojna platforma mora biti besplatna i otvorenog koda, jednako dostupna svima. Zbog situacije u kojoj se Internet polako uvlači u sva područja ljudskog djelovanja, ne bih htio da se takav prostor monopolizira od strane jedne kompanije, u ovom primjeru Adobe. Smatram da postoji potreba za kvalitetnom platformom otvorenog koda s kojom će se moći razvijati aplikacije koje će jednako dobro raditi na sva tri prevladavajuća operacija sustava i na svim glavnim Internet pretraživačima – nekakav virtualni stroj koji bi mogao raditi u svim uvjetima, a opet podržavati i napredne tehnologije poput iskorištavanja hardverske akceleracije za grafiku i video. To bih naveo i kao glavni nedostatak Flash platforme, zatvorenost te cijena razvojnih alata.

Iako postoji besplatni *command line* kompajler i razbubnik (eng. *debugger*), MXML jezik za kreiranje GUI-ja, ActionScript za procesnu logiku, Flash player za pokretanje aplikacije i Flex biblioteka sa GUI komponentama tako da se može uz male troškove krenuti u razvoj ozbiljne aplikacije temeljene na Flashu, u određenom trenutku je potrebno je koristiti integrirano IDE rješenje te je onda to potrebno i platiti. Iako, naravno, smatram da softver nije i ne može biti potpuno besplatan, kad jedna tehnologija poput Flasha postane praktično standard bilo bi bolje da se za razvoj aplikacija ne mora platiti početna svota. A i daljnji razvoj ovisi o željama i

interesima jedne korporacije, a ne globalne Internet zajednice što svakako nije pozitivna činjenica.

Pa ipak, da krećem iz početka najvjerojatnije bih opet krenuo istim putem, razvijajući aplikaciju u Flashu zbog već navedenih razloga i prednosti u odnosu na druge tehnologije navedene i u prethodnim poglavljima.

5. Zaključak

Kada sam krenuo u izradu aplikacije imao sam puno poteškoća. Nisam ništa znao o ActionScriptu niti sam imao ideju kako ću ja to zapravo napraviti. Imao sam samo osnovnu ideju da je potrebno napraviti parser koji će klase iz neke XML bazirane ontologije prebaciti u neku podatkovnu strukturu te zatim pokušati vizualizirati dobiveno. Krenuo sam u potpunosti od početka proučavajući „Hello World!“ primjere. To su bile poteškoće tehničke prirode, no one nisu bile jedine.

Sam pojam semantičkog Weba i vizualizacije znanja prilično je apstraktan. Odgovarajući kolegama na pitanje o čemu se točno radi u ovom radu, već bi velike probleme imao objašnjavajući što je to semantički Web, što je ontologija. Možda je baš to i odgovor na pitanje zašto nam je potrebna vizualizacija znanja. No, vizualiziranje nekog apstraktnog pojma nije niti približno jednostavno kao, primjerice, vizualizacija nekog pojma kojeg možemo vidjeti u prirodi. Zato su nam potrebni vrlo napredni vizualizatori koji će znanje prikupljeno semantičkim vezama prikazati na intuitivan i čovjeku prirodan način.

Kao ozbiljan problem, osim nedostatne tehničke razine za takav pothvat, više vidim u vrlom malom broju ontologija na webu. Teško je očekivati da će se postojeće informacije na Webu retroaktivno obogaćivati semantičkim vezama, osim možda u nekim posebnim slučajevima kao što su povijesne informacije i slično. Dakle, problem je u kreiranju sadržaja koji će već u početku biti semantički obogaćen. Trebalo bi razvijati alate koji bi širokim masama – korisnicima blog servisa ili društvenih mreža, gdje se generira velika količina informacija, omogućili da već prilikom unosa sadržaja informacije obogaćuju semantičkim vezama, možda i nesvjesno! Takav trend bi trebali pratiti i portali te svi ostali koji profesionalnu objavljuju sadržaj. Poučeni iskustvom Web 2.0 servisa koji su doživjeli ogroman rast u vrlo kratkom razdoblju, velik je potencijal korisnika interneta da oni sami kreiraju sadržaj pošto mnoge stranice funkcioniraju na taj način.

Visoku motivaciju za razvoj takvoj alata sigurno imaju današnji pretraživači jer informacije obogaćene semantičkim vezama imaju i komercijalni potencijal. Upravo bi to mogla biti pokretačka snaga za bržim razvojem semantičkog weba ako bi svoj komercijalni uspjeh u tome vidjela neka korporacija poput Googlea.

Možda nas čeka novi *dot com boom* kao krajem devedesetih ali ovog puta na krilima semantičkog Weba.

6. Literatura

6.1 Izrada aplikacije

1. Orchard D. (2004). *RDF and OWL for extensibility and versioning*. Dostupno 17.02.2009. na http://www.pacificspirit.com/Authoring/Compatibility/OWLRDFExtensibility.html
2. *Overview of the ActionScript 3.0 Language and Components Reference*. Dostupno 18.02.2009. na http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3
3. McCauley T. (2006). *Event handling with ActionScript 3*. Dostupno 02.04.2009. na http://www.senocular.com/pub/kirupa/Event%20Handling%20with%20ActionScript%203.doc
4. *Programming ActionScript™ 3.0*, 2007., Adobe Systems Incorporated. Dostupno uz softver Adobe Flash CS4.
5. Shupe R., Rosser Z. (2008). *Lerning ActionScript 3.0*. Canada, O'Reilly Media, Inc.
6. Baczynski M. (2007.) *AS3 Data Structures For Game Developers*. Dostupno 20.04.2009. na http://www.polygonal.de/ds/api/
7. Za demonstraciju rada aplikacije korišteni primjeri ontologija preuzetih s http://autopoiesis.foi.hr i http://www.w3.org/TR/owl-guide/

6.2 Izrada dokumentacije

1. Gospodnetić L. (2003). *Semantički Web*. Dostupno 27.08.2009. na http://www.zemris.fer.hr/predmeti/mr/arhiva/2002-2003/seminari/finished/pdf/semantic_web.pdf
2. Ontologija u informacijskim znanostima. Dostupno 27.08.2009. na http://hr.wikipedia.org/wiki/Ontologija_%28informacijske_znanosti%29
3. Davies J., Studer R., Warren Paul. (2006). *Semantic Web Technologies.* England, West Sussex: John Wiley & Sons Ltd
4. Judelman G.B. (2004). *Knowledge Visualization*. Dostupno 28.08.2009. na http://www.gregjudelman.com/media/judelmanThesis2004.pdf

5. Fain Y., Rasputnis V., Tartakovsky A. (2007). *Rich Internet Applications with Adobe Flex and Java.* USA, SYS-CON

6. *Pellet: The Open Source OWL Reasoner*, Clark & Parsia, LLC. Dostupno 07.09.2009. na http://clarkparsia.com/pellet/