SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1681

WEB SERVIS ZA PRIKAZ STATISTIČKIH ZNAČAJKI TEKSTA I EUROVOC POJMOVNIKA

Ivana Zorović

Zagreb, rujan 2007.

Zahvaljujem se prof. dr. sc. Bojani Dalbelo-Bašić
na usmjeravanju i pruženoj potpori.

Zahvaljujem se kolegi Frani Šariću na savjetima i
uvijek spremnoj pomoći.

Zahvaljujem se kolegi Ivanu Vidoviću na ugodnom radu tijekom izrade
zajedničkog projekta.

Hvala svim prijateljima koji su bili uz mene tijekom svih godina studija.

Najveće hvala mojoj obitelji na razumijevanju i
podršci tokom studiranja.

Sadržaj

1. Uvod

Danas nailazimo na sve veći problem učinkovite pohrane tekstualnih dokumenata. Svakodnevno ručno pregledavanje dokumenata i traženje značajki u tekstu te njihova klasifikacija u određene kategorije zahtjeva velik napor. Taj proces, ne samo da je naporan već je i neunčikovit zbog svoje sporosti. Broj dokumenata koje treba indeksirati iz dana u dan sve više raste, a da bi se svakom pohranjenom dokumentu moglo pristupiti, ručna obrada dokumenta mora biti temeljita, a deskriptori kojima se indeksira dokument pažljivo odabrani.

Automatsko ili poluautomatsko indeksiranje dokumenata prestavlja rješenje ovog problema i uvelike pomaže ljudima pri indeksiranju dokumenata. Predloženi dokument se takvim sustavom analizira a na temelju analize, sustav predlaže niz deskriptora iz kontroliranog rječnika (pojmovnika) dobivenih procesom analize dokumenta. Popis deskriptora je dan na odabir čovjeku koji indeksira dokument. Na taj način čovjek može odabrati iz popisa deskriptora one deskriptore koji dobro određuju dokument i svrstati dokument u kategoriju iz pojmovnika. Pridjeljeni deskriptori služe kao ključne riječi te omogućavaju bržu i lakšu pretragu pohranjenih dokumenata po kategorijama pojmovnika.

U tu svrhu je napravljen sustav eCADIS (*eng.* Computer Aided Document Indexing System) koji omogućuje indeksiranje dokumenata prema pojmovniku Eurovoc (*eng.* EUROpean VOCabulary). Sustav omogućuje leksičku i statističku obradu dokumenta te prikaz Eurovoc pojmovnika na hrvatskom i engleskom jeziku. Korisniku je omogućen prikaz n-grama, lema i ponuđenih deskriptora dobivenih analizom dokumenta.

Rezultati koje sustav eCADIS daje su više nego zadovoljavajući i iz tog razloga se javila ideja za implementaciju web verzije eCADIS sustava koja će omogućiti široj publici indeksiranje dokumenata i uvid u funkcionalnosti koje on pruža.

U sljedećim poglavljima detaljnije će biti riječ o eCADIS sustavu da bi se dobio uvid u važnost sustava i mogućnosti koje on nudi. Dalje će se razmatrati prikladne web tehnologije za implementaciju WEB eCADIS sustava. Zbog složenosti eCADIS sustava i funkcionalnosti koje podržava i nudi korisnicima bit će potrebno odabrati web tehnologiju koja će davati najbolje rezultate za premošćivanje jaza između web tehnologija i tehnologije u kojem je razvijen eCADIS sustav. Zatim sljedi proces nastajanja WEB eCADIS sustava i detaljni opis njegovih funkcionalnosti.

2. Dubinska analiza podataka i indeksiranje dokumenata

Dubinska analiza podataka (*eng*. data mining) je proces analiziranja podataka iz različitih perspektiva i sumiranja dobivenih podataka u korisne informacije.

Dubinska analiza teksta (*eng*. text mining) je jedan oblik dubinske analize podataka u kojem su podatci koji se obrađuju tekstualni dokumenti. Dubinska analiza teksta obično uključuje proces strukturiranja ulaznog teksta (parsiranje teksta, uklanjanje nebitnih informacija iz teksta kao što su stop riječi), izvlačenje uzoraka iz struktuiranih podataka i konačno evaluacija i interpretacija izlaznih informacija.

Dubinska analiza teksta je multidisciplinarno područje koje obuhvaća teoriju pretraživanja i pronalaženja informacija (*eng*. information retrieval), dubinsku analizu podataka, strojno učenje, statistiku i lingvistiku.

Proces automatskog indeksiranja dokumenata je proces u kojem računalo pridjeljuje indekse informacijama koje smatra bitnima za dokument.

Prvi sustavi za automatsko indeksiranje teksta se pojavljuju u ’60-ima. Sve do prošlog desetljeća napretka na tom području nije bilo mnogo zbog ograničene moći tadašnjih računala naspram zahtjevima analize teksta. Situacija se počela mijenjati tehnološkim napretkom što je pridonijelo interesu i radu na ovom području..

2.1. Računalna obrada prirodnog jezika

Proces automatskog raspoznavanja prirodnog jezika možemo smatrati zahtjevnim zadatkom. U analizi teksta, osim dobrog tehničkog znanja, potrebno je dobro poznavanje pravila jezika koji se obrađuje, rječnika, semantike i sintakse [27]. Hrvatski jezik je morfološki vrlo bogat u odnosu na npr. engleski jezik. Kompleksnost prirodnog jezika otežava njegovo razumijevanje i to otežava proces računalne obrade teksta [2].

Tekst možemo prestaviti kao skupinu povezanih lingvističkih jedinica, riječi, sa svrhom ostarivanja komunikacije.

Proces računalne obrade prirodnog jezika zahtjeva provođenje nekoliko procesa nad pisanim tekstom:

- **Tokenizacija** je proces rasčlanjivanja teksta na blokove (*eng*. tokens). Blokovi su sastavljeni od niza znakova kojima se kasnije pridjeljuje vrijednost procesom leksičke analize. Tim procesom su sve riječi u tekstu indentificirane i označene.
- **Lematizacija** je proces konvertiranja riječi iz svoje osnovne tekstualne forme u normaliziranu lingvističku formu –lemu.

Lema neke riječi je njen nominativ jednine ako se radi o imenici, ili infinitiv ako se radi o glagolu.

- **Postupak korjenovanja riječi** (*eng.* stemmning) je proces svođenja oblika riječi u tekstu na njen osnovni oblik ili korijen. Korijen riječi dobiven ovim procesom ne mora biti jednak morfološkom korijenu riječi. Ovi algoritmi rade tako da po određenim pravilima uklanjaju sufikse riječi. Rezultat često nije ispravan korijen riječi već samo njegova aproksimacija. Npr. za engleske riječi "fishing", "fish", and "fisher" algoritam daje korijen riječi "fish".

Svaka riječ u tekstu ima svoje značenje. No, kao i u svakom jeziku, pa tako i u hrvatskom jeziku, nailazimo na problem homonima. Homonimi su riječi koje mogu imati više značenja, a to ovisi o kontekstu u kojem su one izrečene. Npr. riječ luk, može značiti povrće a može značiti i oruđe. Osim homonima, pri računalnoj obtradi teksta treba obratititi pozornost i na sinonime. Sinonimi su različite riječi koje imaju isto značenje u tekstu i zbog toga se pri automatizaciji procesa raspoznavanja teksta moraju tretirati jednako. Npr. kuća i stan, oboje označavaju mjesto stanovanja.

U daljnjem tekstu sustretat će se pojam pojavnica i različnica. **Pojavnica** je svako pojedinačno pojavljivanje “riječi” u korpusu. Npr. u nizu: *stol, stola, stol, stola, stol* nalazi se pet pojavnica.

Različnica (*eng.* type), nasuprot pojavnici je jedinstveni oblik pojavnice iz korpusa. Dakle, u nizu: *stol, stola, stol, stola, stol* nalaze se dvije različnice: *stol* i *stola*.

Riječi koje u tekstu imaju isključivo gramatičku funkciju, nazivamo **stop riječi** (npr. da, i, ili). One ne pridonose značenju teksta pa se izostavljaju iz tog procesa indeksiranja.

N-grami su nizovi od n riječi međusobno odvojenih razmacima ili interpunkcijskim znakovima. Posebno zanimljivi n-grami su oni nizovi riječi koji se pojavljuju zajedno u tekstu češće nego slučajno [32]. Takve nizove riječi u računalnoj lingvistici nazivamo kolokacije.

Kolokacije možemo opisati kao izraze sačinjene od dvije ili više riječi koje imaju neko značenje. Ponekad kolokacije ako gledamo svaku riječ od koje su sastavljene posebno, nemaju isto značenje kao kad ih promatramo u cjelini. Kolokacija strojno učenje se može doslovno shvatiti kao spoj značenja riječi strojno i učenje, ali također je naziv za granu računarstva. Ako gledamo odvojeno te dvije riječi , one gube na svom smislu i značenju.

3. Sustav za strojno potpomognuto indeksiranje dokumenata (Pei/eCADIS)

3.1. Razvoj Pei/eCADIS sustava

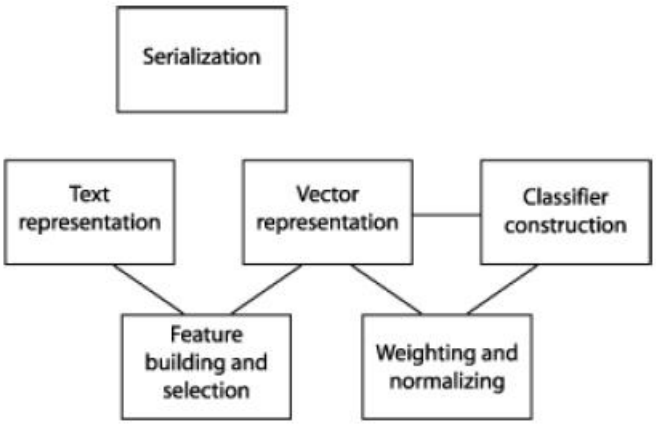
Razvoj PEI (Poluautomatski Eurovoc Indeksator) sustava započeo je u rujnu 2004. godine u timu stručnjaka Fakulteta elektrotehnike i računarstva, Filozofskog fakulteta Sveučilišta u Zagrebu, Hrvatske informacijsko-dokumentacijske referalne agencije (HIDRA) i Zajedničkog istraživačkog centra Europske komisije u Ispri, Italija. Danas je taj sustav, iz poluautomatskog prerastao u automatski sustav za indeksiranje i preimenovan u eCADIS.

Primarni cilj eCADIS sustava je bio olakšati zadaću indeksiranja službenih dokumenata Republike Hrvatske deskriptorima iz Eurovoc pojmovnika (vidi 3.3.). Izrada sustava je završena u travnju 2006. godine. eCADIS sustav obavlja automatsko indeksiranje teksta i omogućuje brže, efikasnije i uniformnije indeksiranje dokumenata u odnosu na ručno indeksiranje [1]. Rad eCADIS sustava se temelji na metodama izgrađenim i sadržanim u TMT biblioteci.

3.2. TMT biblioteka

Svrha TMT (*eng.* Text Mining Tools) biblioteke je da omogući korištenje text-mining tehnika na svim platformama te primjenu u istraživanju i razvoju aplikacija na području text mining-a [11].

TMT biblioteka je razvijena u C++ programskom jeziku iz više razloga. Osim što je njena kompleksnost zahtjevala razvoj u nekom od objektno-orijentiranih jezika, izbor ovog objektno-orijentiranog jezika omogućuje brzo izvršavanje programskog koda, prevođenje na različitim platformama i olakšanu integraciju TMT biblioteke u druge aplikacije kao i njenu nadogradnju.



Slika 3.1. *Relacije među modulima TMT biblioteke*

Funkcionalnosti TMT biblioteke su tokenizacija, lematitacija, postupak korijenovanja riječi, izgrađivanje značajki, selekcija i određivanje težina.

TMT biblioteka koristi tehnike učenja s nadzorom (*eng.* supervised learning) za treniranje klasifikatora pa ulazni dokumenti moraju biti pretprocesirani. Nakon faze treniranja dobiveni klasifikator može se spremiti i označiti, što omogućuje pronalazak učinkovitog klasifikatora koji se može integrirati u druge programe.

Dosad, TMT biblioteka podržava k-NN, Bayes, Rocchio klasifikator, SVM. Implemenitrane metode klasifikacije omogućuju laku implementaciju i iskoristivost funkcionalnosti TMT biblioteke.

TMT biblioteka sadrži oko 50-ak klasa koje su hijerarhijski struktuirane i na taj način se za svaku granu stablaste strukture može izgraditi klasifikator. Prednosti korištenja hijerarhijske klasifikacije su poboljšane performanse i smanjeno vrijeme treniranja i testiranja nad određenim skupom podataka.

Upute za korištenje te detaljniji opis funkcionalnosti koje omogućuje TMT biblioteka dostupan je na službenim stranicama textmining grupe [6].

3.3. Eurovoc pojmovnik

Pojmovnik Eurovoc (*eng.* EUROpean VOCabulary) je hijerarhijski organizirani, multidisciplinarni pojmovnik [2] za indeksiranje dokumenata korišten u mnogim Europskim institucijama.

Sastavljen je od 21 područja i 127 potpodručja koje obuhvaćaju sve djelatnosti Europske unije u obliku strukturiranoga i kontroliranoga popisa naziva za sve važne pojmove koji se javljaju u najrazličitijim dokumentima EU-a [4].

Eurovoc je preveden na 21 službenih jezika EU-a, a dodatno su ga prevele i neke druge zemlje, među njima i Hrvatska [5]. Zbog složenosti hrvatskog jezika i postupka indeksiranja dokumenata Republike Hrvatske, HIDRA je napravila dodatak pojmovniku vezan uz županije, političke stranke, razne institucije kojih nema u Eurovocu, itd.

Područja koje pokriva Eurovoc pojmovnik sa dodatkom koji je napravila HIDRA su:

- poljoprivredno-prehrambena industrija
- okoliš
- Europske zajednice
- financije
- pravo
- proizvodnja, tehnologija i istraživanje
- obrazovanje i komunikacije
- zapošljavanje i radni uvjeti
- ekonomija
- poslovanje i konkurencija
- lokalna i područna (regionalna) samouprava RH
- zemljopis RH
- državne institucije RH
- prijevoz
- međunarodne organizacije
- znanost
- energija
- trgovina
- zemljopis
- industrija
- društvena pitanja
- međunarodni odnosi
- institucije u RH
- poljoprivreda, šumarstvo i ribarstvo
- političke stranke RH
- politika
- diplomatski odnosi RH

Pojava sinonima u dokumentu može biti zbunjujuća za sam proces indeksiranja. Jedna od glavih svrha pojmovnika je rješavanje problema sinonima. Jedan od sinonima je odabran kao deskriptor (*eng*. descriptor, pojam koji opisuje tekst). Samo deskriptor se može koristiti pri indeksiranju teksta, a ostali sinonimi prestavljaju asociate tom deskriptoru (npr. deskriptor "ekonomska analiza" ima i asocijat "gospodarska analiza" itd.).

Deskriptori unutar Eurovoc pojmovnika imaju uređenu strukturu i odnose . Jedna od njih je i hijerarhijska struktura. Na vrhu se nalaze deskriptori koji nemaju širi značenje od svog (*eng*. broader term -BT) a deskriptor koji ima specifičnije značenje ima oznaku NT (*eng*. narrowed term). Asocijativni odnos među deskriptorima je prestavljen skraćenicom RT (*eng*. related term). Definiran je i odnos ekvivalencije između deskriptora i njegovih asocijata.

3.4. Funkcionalnost Pei/eCADIS sustava

3.4.1. Ulaz u sustav

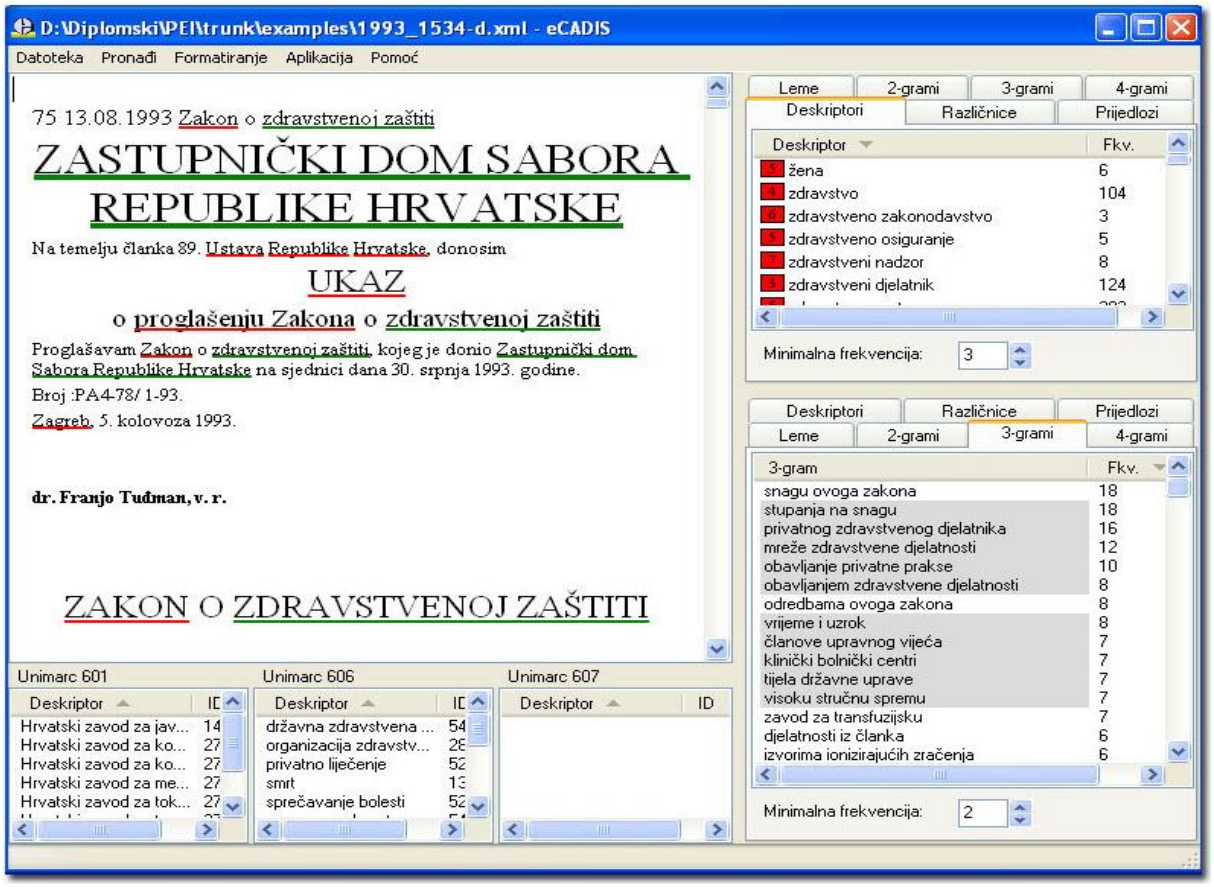
Ulaz u sustav je definiran datotekom zapisanom u XML formatu (Extensible Markup Language) [3]. Zbog nemogućnosti podržavanja svih formata zapisa dokumenata odabaran je XML, standardizirani format zapisa dokumenata kojem je glavna svrha razmjena podataka u informacijskim sustavima i koji zbog svoje dobro definirane strukture zapisa podataka ima široku uporabu.

3.4.2. Poluautomatski Eurovoc indeksator (PEI)

Sustav, na temelju dokumenta dobivenog na ulazu, gradi unutrašnju strukturu podataka. Leksičkom i statističkom obradom dokumenta, korisniku generira informacije o pojavnicama, lemama, deskriptorima i n-gramima. Izgled korisničkog sučelja PEI-a je vidljiv na slici 3.2.

Vizualna rekonstrukcija dokumenta dobivena sustavom omogućuje korisniku praćenje rada sustava i rezultata dobivenih indeksiranjem. Na desnoj strani postoje dva preglednika. Oni nude korisniku uvid u dobivene rezultate. Rezultati koje korisnik može vidjeti su:

- popis deskriptora i asocijata i pripadne frekvencije pojavljivanja unutar dokumenta
- popis lema i pripadne frekvencije pojavljivanja unutar dokumenta
- popis pojavnica i pripadne frekvencije pojavljivanja unutar dokumenta
- popis n-grama (bigrami, trigrami i tetragrami) i pripadne frekvencije pojavljivanja unutar dokumenta



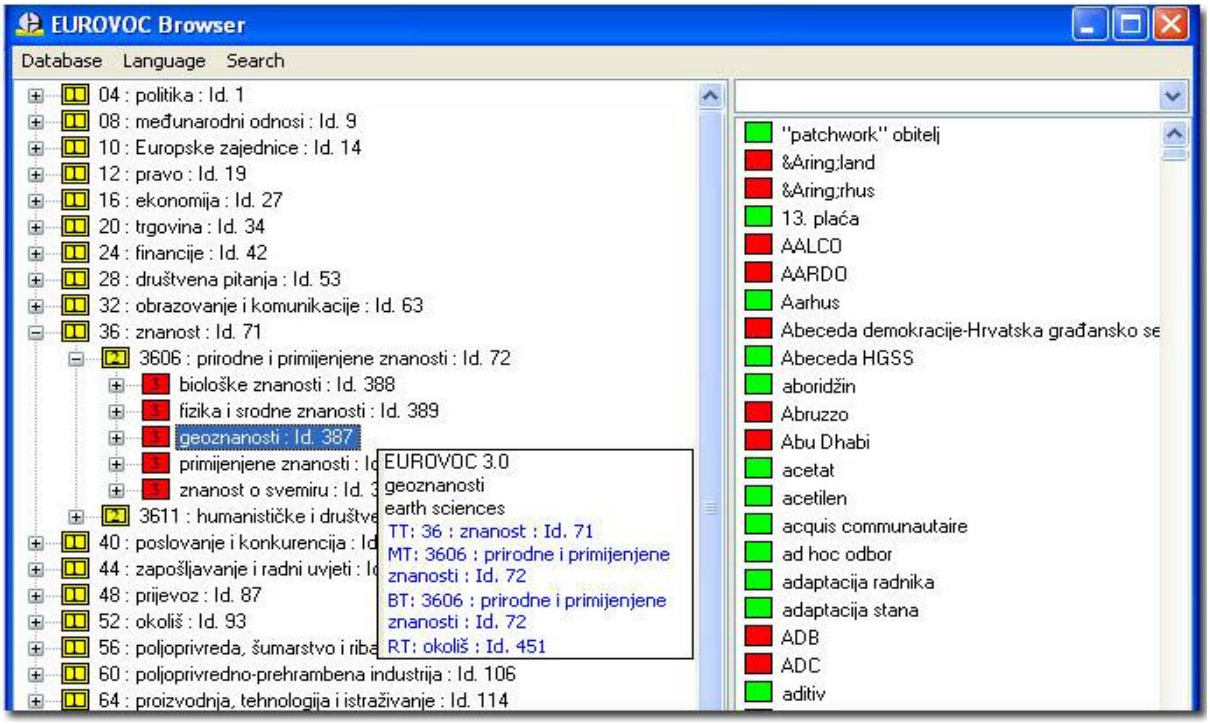
Slika 3.2. *– Korisničko sučelje PEI-a. Na lijevoj strani se prikazuje dokument predan na indeksiranje a na desnoj strani podaci dobiveni obradom algoritama implementiranih u sustav eCADIS. (Prikaz lema, n-grama, deskriptora i pojavnica).*

Odabirom jedne od ponuđenih opcija na izborniku desno npr. bigrama, dobiju se rezultati svih nađenih bigrama u dokumentu i njihova učestalost ponavljanja. Rezultati su vidljivi u tablici, a pomicanjem praga minimalne frekvencije pojavljivanja (minimal frequency), popis nađenih bigrama će se korigirati. Također, program omogućuje sortiranje rezultata u tablici abecedno po nazivu ili po frekvenciji pojavljivanja. Na desnoj strani korisničkog sučelja su vidljiva dva ista izbornika koji omogućavaju korisniku istodobno praćenje različitih rezultata npr. bigrama u prvom izborniku i ponuđenih deskriptora u drugom izborniku.

Sama frekvencija pojavljivanja određenih lema ili n-grama, može ali i ne mora biti dovoljna za prosudbu korisnika o kakvojm je točno dokumentu riječ. Da bi se korisniku omogućilo jednostavno proučavanje konteksta u kojem se pojavljuju pojmovi potencijalno bitni za indeksiranje dokumenta, implementiran je algoritam za bojanje lema, pojavnica, deskriptora ili n-grama unutar dokumenta. U tekstu se riječi oboje plavom bojom pozadine dok daljnjim pretragama selekcija iterativno prelazi s jednog pojavljivanja riječi na drugo.

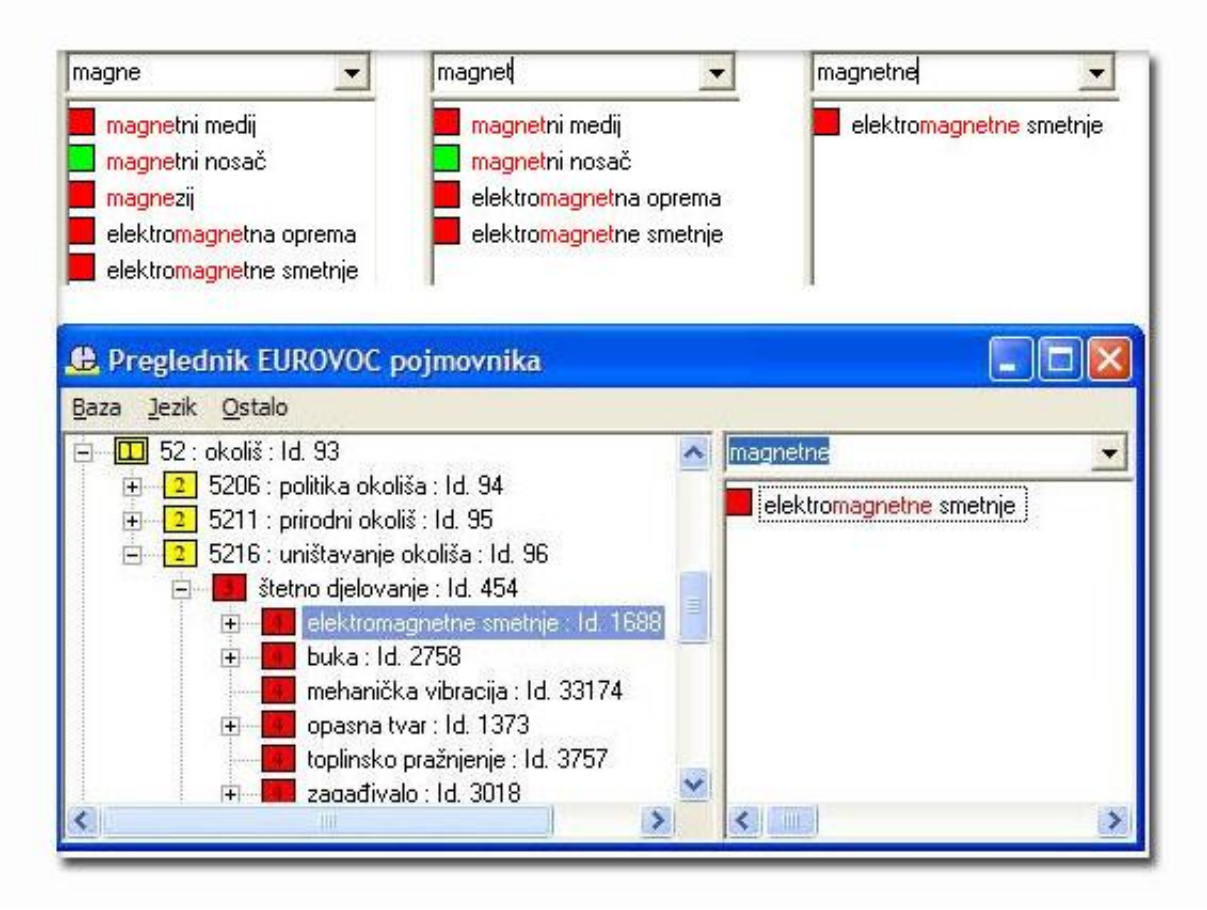
U glavnom izborniku eCADIS sustava moguće je birati jezik sučelja, postavke sustava (izbor klasifikatora i ostale izmjene vezane za vizualnu reprezentaciju dokumenata). Osim višejezičnosti sučelja, podržana je i višejezičnost dokumenata (razvijene su sve potrebne funkcije za obradu dokumenata koje podržavaju višejezičnost) kao i prikaza eurovoc pojmovnika.

3.4.3. Eurovoc prozor



Slika 3.3. *Izgled Eurovoc izbornika unutar eCADIS sustava.*

Na lijevoj strani je hijerarhijski prikaz Eurovoc pojmovnika. Kako Eurovoc pojmovnik sadrži više od 6000 različitih deskriptora korisniku se mora omogućiti što brže i jednostavnije pretraživanje pojmovnika. Osim toga svaki deskriptor može sadržavati i vezu na nadređeni pojam, veze na srodne i podređene deskriptore. Veza sa srodnim, nadređenim i podređenim pojmovima je prestavljena u obliku linka, tako da se klikom na prikazanu vezu, korisniku otvara grana Eurovoc stabla u kojoj se nalazi odabrani deskriptor. Uz pregled svih semantičkih veza u hijerarhijskom pogledu unutar prozora preglednika pojmovnika korisniku je dostupno i pretraživanje pojmovnika.



Slika 3.4. *Prikaz pretraživanje naziva u pojmovniku.*

Nakon svakog utipkanog slova, korisniku se odmah prikazuju rezultati pretrage za uneseni niz. Odabrani deskriptor se odmah prikazuje i u hijerarhiji pojmovnika.

4. Analiza primjenjivih tehnologija

Dinamičke aplikacije karakterizira mogućnost da se dijelovi HTML koda ili cijeli programski kod generira dinamički. Za isti URL može biti isporučen potpuno drugačiji sadržaj. Veličina takvih aplikacija može se kretati od malih skripta do velikih uslužnih programa.

Dinamičke web aplikacije u osnovi možemo podijeliti na one koje se izvršavaju na strani klijenata i one koje se izvršavaju na strani poslužitelja.

Aplikacije koje se izvršavaju na strani klijenta su npr.:

- Java Appleti (prenosi se kod do računala klijenta i ondje se izvršava, uz uvjet da na kllijentu mora postojati JVM – Java Virtual Machine koji izvršava java kod)
- Java Script (podržavaju ga svi browseri)

Prednost ovakvih aplikacija su veća dinamičnost stranica u specifičnim uvjetima koji vladaju zbog http-a ili njemu sličnih protokola (https itd.), no problemi su u tome što se povećavaju zahtjevi prema klijentu, odnosno dolazi do sindroma debelog klijenta (*eng*. fat-client), tj. računalo klijent mora biti snažno i dobrih performansi. Drugi je problem u tome što se opterećuje mreža, a u slučaju da imamo sporu mrežu (npr. spajamo se putem modema), nastaju značajni problemi pri radu.

Aplikacije se izvršavaju na strani poslužitelja

- Kod aplikacije se izvršava na strani poslužitelja (Common Gateway Interface)
- Java Servleti
- Server-side programski jezici koji se integriraju u HTML
    - Professional Home Page (PHP)
    - Active Server Pages (ASP)
    - Perl kao Apache Modul
    - Java Server Pages (JSP)
    - ColdFusion (CF) Allaire (odnedavno Macromedia)

Prednost je skriptnih jezika poput PHP-a ili ASP-a u tome što se sve odvija na poslužitelju koji, istina, mora biti jačih karakteristika, no razlika u cijeni je prihvatljiva. Od klijenta se u takvim uvjetima ne zahtijeva ništa osim da mogu pokrenuti pretraživač. S tim se smanjuje mrežni promet i povećava brzina izvršavanja.

No, koju od tih tehnologija odabrati? Sve su one sličnih karakteristika i mogućnosti. Odabir se uglavnom svodi na odabir između OpenSource (kao ideja svodi se na to da je izvorni kod programskog proizvoda dostupan. Primjeri OpenSource proizvoda su: Apache, Tomcat (web poslužitelji), MySQL baza, PHP, Linux , PostgreSQL) od besplatnih rješenja (PHP, JSP) i komercijalnih proizvoda (ASP, CF). Teško je reći koje je od tih rješenja bolje,

svako ima svoje mane i prednosti, no jedno je sigurno; odluka za Open Source rješenja osjetno smanjuje troškove jer su besplatni, što se možda i isplati s obzirom na uštedu od licenci, a i cijena tih proizvoda u našim uvjetima predstavlja značajnu prepreku.

Premda je riječ o tehnologijama koje su besplatne, ne znači da su one nužno i loše, dapače, ostvarena rješenja mogu se mjeriti s komercijalnim rješenjima u svim granama, jedino su malo složenija za konfiguriranje.

4.1. Problemi premošćivanja

Kako se iza eCadis sustava nalazi TMT biblioteka koja omogućuje rad eCadis sustavu, tako je logično za pretpostaviti da bi se i iza WEB eCadis sustava trebala nalaziti TMT biblioteka.

Prvi problem koje se pojavljuje jest taj da je TMT biblioteka napravljena u C++ programskom jeziku koji nije web tehnologija. Odmah se otvaraju dva moguća rješenja:

- napraviti TMT biblioteku u web tehnologiji
- pokušati iskoristiti TMT biblioteku u C++ tehnologiji

Prvo rješenje bi značilo nastavak razvoja TMT biblioteke u više (dvije) tehnologija i puno rada na izradi iste stvari u drugoj tehnologiji sada.

Drugo rješenje je puno prihvatljivije, no bilo je potrebno pronaći najadekvatnije rješenje od kojih niti jedno nije jednostavno.

Drugo rješenje oslobađa jednog dijela programiranja (prepisivanje i implementaciju funkcionalnosti TMT biblioteke), dok povlači za sobom upotrebu tehnologija za premošćivanje native C++ jezika i web tehnologija. To je potrebno napraviti koja god web tehnologija se odabere. Stoga se odabir prikladne web tehnologije za izradu WEB eCadis sustava svodi na odabir tehnologije u kojoj se može na što jednostavniji način premostiti native C++ i odabrana web tehnologija. Naravno, iskustvo i poznavanje odabrane tehnologije također je trebalo uzeti u obzir. Također je bilo poželjno da se odabrano rješenje ne veže za određeni operacijski sustav.

4.1.1. Skriptni jezici

Neke kritike upućivane PHP-u su generalne zamjerke svim skriptnim jezicima i jezicima s dinamičkim/slabim (*eng.* weak) tipovima podataka, no ima i nekih koje su specifične za PHP:

- U PHP-u deklaracija varijabli nije obavezna, što može biti izvor raznih grešaka i sigurnosnih propusta
- Ugrađene funkcije nisu dosljedne po pitanju njihovog nazivlja ni po pitanju redosljeda argumenata među sličnim funkcijama (primjer nazivanja: strip_tags i html_entity_decode nasuprot stripslashes, htmlentities)
- Funkcije nisu dosljedne u vraćanju rezultata - false, ali mogu vratiti i 0 ili ""
- Broj ugrađenih funkcija je velik (preko 3000) pa to otežava razvoj pogotovo jer dijele isti namespace
- Opcija "magic quotes" koja dodaje backslasheve u stringove se može uključiti ili isključiti u konfiguraciji pa kad se programeri oslanjaju na nju može biti problema na sistemima gdje je isključena
- Opcija "register_globals" automatski kreira varijable iz obrazaca pa to može postati sigurnosni rizik
- Konfigurabilnost PHP-a koja je istovremeno i prednost i nedostatak jer jedna skripte na jednom serveru može raditi dok na dugom ne
- Stabilnost PHP-a mnogo ovisi o vanjskim bibliotekama funkcija [28]

Kako je navedeno na primjeru PHP-a kao skriptnog jezika, a dosta stavki se odnosi i na ostale skriptne jezike, vidljivo je da ima dosta nedostataka u slučaju odabira skriptnog jezika.

Također poznavanje skriptnih jezika nije bilo na zavidnoj razini, a za spajanje sa native jezicima je potrebno koristiti međukod.

4.1.2. .NET

Aplikacije za .NET platformu mogu se pisati u raznim programskim jezicima, gotovo svim poznatijim. CLR, međutim, ne poznaje niti jedan taj jezik - on dobiva naredbe isključivo u jeziku nazvanom Microsoft Intermediate Language (sraćeno MSIL), temeljen na pravilima koja se nazivaju Common Language Specifications (CLS). Stoga je jasno da mora postojati prevodioc (kompajler) koji će programski jezik u kojem programer piše kod prevesti u MSIL kako bi ga CLR razumio. Ovi kompajleri nazivaju se IL-kompajleri te su dostupni za velik broj programskih jezika. Microsoft iz izdao kompajlere za pet jezika: C#, J#, C++, Visual Basic i JScript, dok su se ostali proizvođači softvera potrudili oko brojnih drugih kao što su: Perl, Python, Cobol, Eiffe itd. [28].

Logičan odabir bi se činio korištenje .NET platforme za izradu WEB eCadis sustava, no tu je uvjet da aplikacija bude neovisna o serverskoj platformi, odnosno da ne bude nužno da server na kojem će se aplikacija kasnije „vrtiti“ bude Windows server.

Iako postoji platforma (Mono) koja bi trebala omogućivati rad .NET aplikacija na različitim operacijskim sustavima, tehnologija je nova i nije dovoljno razrađena te nije bilo sigurno da li će aplikacija na kraju raditi.

4.1.3. Java

Java je postala vrlo popularna zbog “write once - run anywhere” sintagme i prenosivog međukoda, koja omogućava da pružatelj usluge može promjeniti hardware sustava bez potrebe za ponovnim prevođenjem programa.

Izvođenje takvog koda je znatno brže od interpretiranih jezika, jer međukod je po strukturi blizak instrukcijama procesora, samim time i znatno jednostavniji od programa u skriptnim jezicima. Drugi čimbenik koji značajno utječe na bolje performanse izvršavanja međukoda je upotreba “just-in-time” prevodilaca. Tiprevodioci su sastavni dio Java izvršne okoline, iza vrijeme izvršavanja programa “u letu” prevode u izvršni kod (engl. “machine executable”) dijelove koda koji se često pozivaju ili uzrokuju usko grlo u izvođenju programa.

Činjenica da Java programski jezik zahtijeva da se programi izvršavaju unutar zaštićene okoline virtualnog stroja (engl. “virtual machine”) značajno poboljšava sigurnosne karakteristike Jave. Java izvršna okolina je vrlo robustna i tolerantna na pogreške. Ako program u Javi počini pogrešku za vrijeme izvođenja baca se iznimka, ili točnije objekt tipa Exception.Takav objekt se može koristiti u manipulaciji pogreškama kako bi cijeli sustav ostao funkcionalan [28].

Rješenje pisano u Javi rješava probleme koji postoje u skriptnim jezicima, Java je potpuni objektno orjentirani jezik, stabilan, robustan, neovisan o platformi, no ne omogućuje jednostavno premošćivanje C++ native koda i Java međukoda kao što to omogućuje .NET.

U Javi postoji JNI (Java Native Interface, vidi poglavlje 5.2.) koji omogućuje pisanje posebnog međukoda za spajanje Jave i native programskih jezika.

4.2. Odabrana tehnologija

Korištenjem skriptnih jezika i Jave potrebno je premostiti C++ native kod i odabrani jezik. Java kao tehnologija je mnogo razvijenija i stabilnija no ujedno i složenija za korištenje od skriptnih jezika. No osobno poznavanje Jave je bilo puno bolje od poznavanja skriptnih jezika, tako da ni složenost korištenja nije mogla prevagnuti u odabiru između Jave i skriptnih jezika.

Usporedbom Jave i .NET okruženja bi došli do zaključka da se usporedba tih dvaju tehnologija ne svodi na usporedbu njihovih prednosti i nedostatke nego na mogućnost implementacije zahtjeva WEB eCADIS sustava.

- Java – problem premošćivanja C++ native koda i Java međukoda
- .NET – problem izvedivosti .NET međukoda na neWindows operacijskim sustavima

Zbog Mono platforme koja nije u potpunosti sigurna i još je u razvoju, rješenje je bilo napraviti aplikaciju u .NET-u i prihvatiti da će još neko vrijeme raditi samo na Windows serverima dok Mono platforma upotpunosti ili pokušati vidjeti kakvo se rješenje nudi u Javi , koliko je razvoj kompliciran i da li se isplati izgradnja takvog rješenja.

Nakon analize JNI koda zaključak je bio da se ne isplati izgradnja sustava u Java tehnologiji, jer je JNI kod zapravo novi jezik, zahtijevao bi učenje potpuno novog jezika i pisanje ne malo koda u nepoznatom jeziku je ravno pisanju TMT biblioteke u Javi.

U tom trenutku je odluka pala na .NET no nakon još malo vremena istraživanja, saznali smo za SWIG program. SWIG program je zapravo generator međukoda, koji smo nakon analize odlučili koristiti u kombinaciji s Javom (SWIG omogućuje generiranje međukoda za više jezika PHP, Tcl, Perl, Java, i td.).

Nakon upoznavanja sa SWIG-om može se reći da nije bilo razlike u kompliciranosti rada aplikacije u .NET okruženju ili Javi stoga je odluku nije bilo teško promijeniti i krenuti razvijati aplikaciju u Java tehnologijama.

5. Korištene tehnologije za razvoj aplikacije u Java programskom jeziku

Za razvoj web aplikacije u Java programskom jeziku, trebalo je prije toga razraditi tehnologije koje će se koristiti u međukoracima spajanja C++ sloja i Java sloja. Detaljniji opis korištenih tehnologija slijedi u idućim podpoglavljima.

5.1. Cmake

Bit Cmake „open-source“ programa jest da omogućuje kreiranje projekata za različite operacijske sustave i za različita razvojna okruženja na njima (*eng.* cross-platform make system). Koristi se za kontrolu procesa kompajliranja softvera koristeći jednostavne platformski-neovisne i kompajlerski-neovisne konfiguraciske datoteke. Cmake generira projektne i izvršene datoteke koje su prirodne za računalo na kojem se žele pokrenuti (*eng.* native makefiles) te se mogu koristiti za kompajlesku okolinu koja se odabere. Cmake je sofisticiran, omogućuje korištenje kompleksnih okolina koje zahtjevaju sistemske konfiguracije, generiranje predprocesorskih direktiva, generiranje koda i instanciranje predložaka. [10]

Npr. razvojno okruženje korišteno na Windows operacijskom sustavu je .NET. U konfiguracijskim datotekama za Cmake program se opiše projekt. Postoje različite naredbe, od toga za koji operacijski sustav da se stvori projekt, za koje razvojno okruženje, koje datoteke taj program uključuje i sl.

Glavne značajke koje omogućuje Cmake su:

- Podržava složena, velika razvojna okruženja.
- Generira „native“ izvršne datoteke (make datoteke za Unix, workspaces/projects za MS Visual C++). Iz tog razloga se alat može koristiti na bilo kojoj platformi i za bilo koji kompajler.
- Posjeduje jake komande za lociranje „include“ datoteke, biblioteke, izvršne datoteke. Uključuje vanjske Cmake datoteke koje omogućuju standardne funkcionalnosti, sučelja prema sustavima za testiranje, podržava rekurzivni obilazak direktorijasa nasljeđivanjem varijabli, može pokretati vanjske programe, podržava uvjetnu izgradnju, podržava regularne izraze, i td.
- Podržava izgradnju na licu mjesta (ako je potrebno izgraditi datoteke u istom direktoriju u kojem se nalazi Cmake datoteke) i izgradnju u drugim direktorijima. Više kompajlerskih verzija je moguće iz jednog izvora.
- Omogućuje lako dodavanje novih naredbi.

- Cmake je „open-source“ alat.
- Cmake je napravljen tako da omogućuje povezivanje s grafičkim editorima [28].

```
FIND_PACKAGE(SWIG REQUIRED)
INCLUDE(${SWIG_USE_FILE})

FIND_PACKAGE(Java REQUIRED)
FIND_PACKAGE(JNI REQUIRED)

INCLUDE_DIRECTORIES(${JAVA_INCLUDE_PATH})

IF(WIN32 OR MINGW OR UNIX)
  INCLUDE_DIRECTORIES(${JAVA_INCLUDE_PATH2})
ENDIF(WIN32 OR MINGW OR UNIX)

INCLUDE_DIRECTORIES(${JAVA_AWT_INCLUDE_PATH})
LINK_DIRECTORIES(${JAVA_AWT_LIB_PATH})

SET(CMAKE_SWIG_OUTDIR ${CMAKE_CURRENT_SOURCE_DIR}/WebPei/src/org/fer/tmt)

SET_SOURCE_FILES_PROPERTIES(WebJava.i PROPERTIES CPLUSPLUS ON)
SET_SOURCE_FILES_PROPERTIES(WebJava.i PROPERTIES SWIG_FLAGS "-includeall;-package;org.fer.tmt")
SWIG_ADD_MODULE(WebJava java WebJava.i WebJava.cpp Classifier.cpp XMLDocumentSpanifier.cpp)
SWIG_LINK_LIBRARIES(WebJava tmt)
```

Slika 5.1. *dio konfiguracijske datoteke kojim se daju upute Cmake programu da uključi SWIG u projekt i upute SWIG programu kako da napravi JNI omotač.*

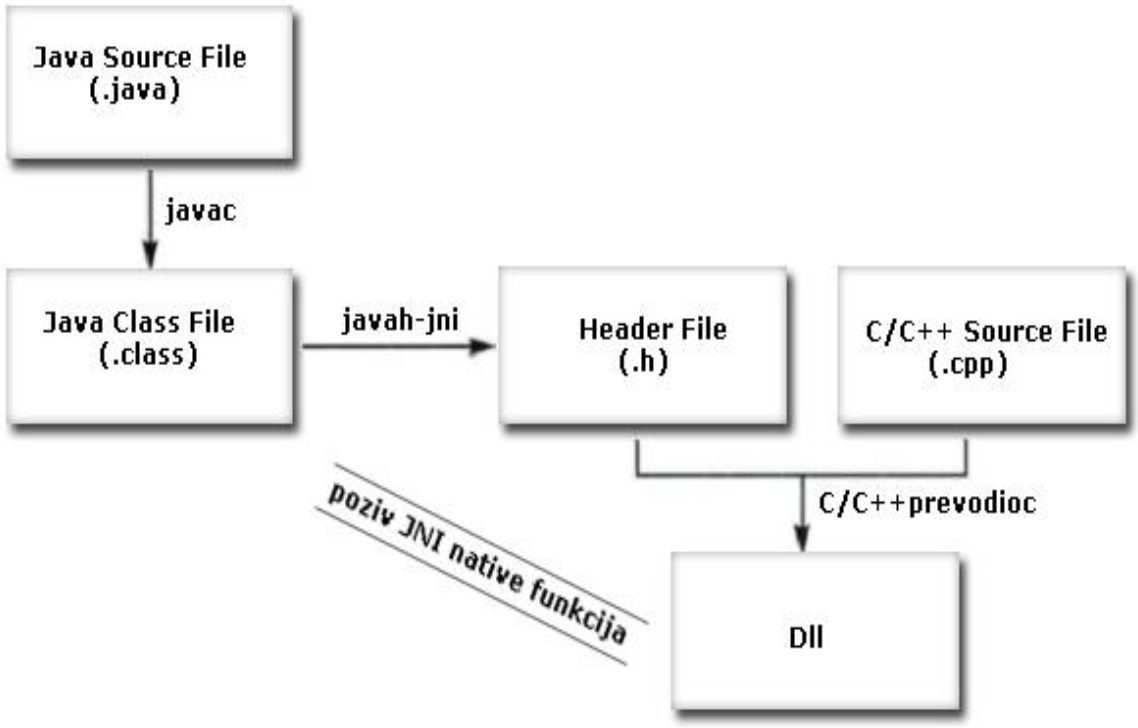
5.2. JNI

Java Native Interface (JNI) je jedini standardni mehanizam koji omogućuje interakciju programskog jezika Jave sa jezicima čije se naredbe prevode izravno u strojni kod [9].

Strojni kod (*eng.* native code) je kod koji se izvršava izravno u kontrolnoj procesnoj jedinici – CPU.

5.2.1. Kada koristiti JNI ?

Postoje velik broj aplikacija napisanih u drugim programskim jezicima i namjenjenih izvršavanju na jednom računalu. Razvojem Interneta, javila se potreba korištenja tih aplikacija i na internetu. Zbog kompleksnosti većine napisanih aplikacija i zbog velikih troškova njihove reimplementacije, koriste se mehanizmi koji mogu interaktirati između web tehnologija i programskih jezika koji se direktno prevode u strojni kod, od kojih je jedan JNI.



Slika 5.2. *Prikaz korištenja JNI u procesu povezivanja Jave i C++ programskog jezika.*

Poziv JNI native funkcija se odvija na Java sloju. JNI funkcije pozivaju DLL (*eng.* Dinamic Link Library) u kojem su zapisane prevedene metode korištene na C++ sloju. Na taj način je ostvarena veza između Java sloja i C++ sloja.

Proces prikazan na slici 5.2. možemo zapisati po koracima:

1. Kreiranje DLL na temelju metoda napisanih na C/C++ sloju
2. Generiranje JNI klasa
3. Implementacija JNI native metoda na Java sloju

4. Prevođenje programskog koda napisanog u Javi i njegovo izvršavanje

5.2.2. Prednosti i nedostatci JNI arhitekture

Prednosti JNI arhitekture su:

- Binarna kompatibilnost- JNI programski kod je kompatibilan sa svakom JVM (Java Virtual Mashine) istog operacijskog sustava. To znači da se Java aplikacija može prevesti na jednoj JVM, a izvoditi na bilo kojoj drugoj JVM koja podržava JNI.

- Ograničenje pristupa- JNI ograničava pristup JVM-u iz programskih jezika koji se direktno prevode u strojni kod.

- Napredni mehanizam regulacije zagušenja toka programa

Nedostatci korištenja JNI:

- Portabilnost- korištenje JNI u Javi ograničava korištenje Java aplikacije na određeni operacijski sustav. Za korištenje Java aplikacije na Windows sustavu potreban je drugačiji format zapisa DLL-a.

- Kompleksnost – Aplikacija koja koristi JNI je kompleksna iz razloga što zahtjeva od programera dobro poznavanje Java programskog jezika i C/C++ jezika.

- Sigurnost – Java aplikacije koje koriste funkcije koje se direktno prevode u strojni kod nisu toliko sigurne kao čiste Java aplikacije

5.3. SWIG

SWIG (*eng*. Simplified Wrapper and Interface Generator, Deavid Beazly) je razvojni alat (program development tool) nastao istraživanjem u Los Alamos National Laboratory-u. Namjenjen je korisnicima koji žele koristiti skriptne jezike ili Javu zajedno sa svojim C/C++ programom a ne žele se zamarati detaljima implementacije korištenih programskih jezika [7].

SWIG automatski generira sav potreban programski kod za povezivanje C/C++ funkcija sa skriptnim jezicima ili Javom koristeći, pri tom, samo ulaznu datoteku koja sadrži deklaracije varijabli i funkcija (*eng*. header file). Na temelju toga, SWIG generira omotač oko (*eng*. wrapper code) oko C/C++ klasa koji se onda kasnije koristi iz skriptnog jezika ili Jave.

SWIG nudi mogućnost podešavanja generiranog omotača u svrhu bolje prilagodbe aplikaciji koja ga koristi [8].

SWIG trenutno uspješno premošćuje C/C++ i sljedeće jezike:

- Allegro CL
- C#
- Chicken
- Guile
- Java
- Modula-3
- Mzscheme
- OCAML
- Perl
- PHP
- Python
- Ruby
- Tcl

Na ulaz SWIG prima ulaznu datoteku koja sadrži nekoliko specifičnih direktiva SWIG-a a ostatak datoteke čine ANSI C funkcije i deklaracije varijabli. Ulazna datoteka ima ekstenziju -.i, npr. WebJava.i.

Ulazna datoteka koju zahtjeva SWIG je praktički ista kao ulazna datoteka za C++ programski jezik, kao što je vidljivo na slici 5.3. Sa slike možemo isčitati deklaracije funkcija i varijabli napisane u C++, a dodatak koji je potreban SWIGU da napravi „*wrapper file*“ je -%module directiva. Ona postavlja ime inicijalne funkcije koju SWIG poziva. Programski kod unutar %{, %} bloka, kopira se direktno na izlaz omogućujući uključivanje izvornog C/C++ koda (header file).

```
%module swigjava
%include "std_string.i"
%include "std_vector.i"
%{
#include "wrappers/webjava/WebJava.h"
%}


class EurovocElemJava {
public:
          EurovocElemJava();
          ~EurovocElemJava();
          int getID();
          void setID(int ID);
          std::string getName();
          void setName(std::string NM);

private:
          int id;
          std::string name;
};

class ClassifierJava{
```

Slika 5.3. *Prikaz WebJava .i datoteke korištene u projektu izgradnje Web eCadis sustava.*

U datoteci WebJava.i se nalaze sve deklaracije varijabli i korištenih funkcija iz TMT biblioteke pri izradi WEB eCadis-a. Korištene funkcije je bilo potrebno prilagoditi potrebama Java sloja.

Na slici 5.4. je prikazan izgled datoteke dobivene na izlazu SWIG-a. Datoteka ima ekstenziju -_wrap.cxx i mnogo je veća od datoteke na ulazu. Zbog toga ju je bilo nemoguće čitavu prikazati, ali iz ovog dijela koji je na slici vidi se rezultat dobiven pretvorbom.

```
#ifdef __cplusplus
extern "C" {
#endif

SWIGEXPORT jlong JNICALL
Java_eCadis_swigjavaJNI_new_1EurovocElemJava(JNIEnv *jenv, jclass jcls)
{
  jlong jresult = 0 ;
  EurovocElemJava *result = 0 ;

  (void)jenv;
  (void)jcls;
  result = (EurovocElemJava *)new EurovocElemJava();
  *(EurovocElemJava **)&jresult = result;
  return jresult;
}


SWIGEXPORT void JNICALL
Java_eCadis_swigjavaJNI_delete_1EurovocElemJava(JNIEnv *jenv, jclass jc
 {
  EurovocElemJava *arg1 = (EurovocElemJava *) 0 ;

  (void)jenv;
  (void)jcls;
  arg1 = *(EurovocElemJava **)&jarg1;
  delete arg1;

}
```

Slika 5.4. *Prikaz datoteke WebJava_wrap.cxx koju je izgenerirao SWIG na temelju ulazne datoteke WebJava.i*

SWIG pruža podršku gotovo svim funkcijama C++ jezika. Podržava sve tipove podataka, reference, klase, nasljeđivanje među klasama, template, statičke metode itd. U slučaju nestandardnog tipa podataka, SWIG pretpostavlja da je ta specifična struktura podatka definirana ranije u ulaznoj datoteci u protivnom on ju neće moći prepoznati. Takav slučaj deklaracije specifičnog tipa podatka je vidljiv na slici 5.5. i ima oznaku %template.

```
class EurovocJava {
public:
        EurovocJava(std::string eurovocPath);
        ~EurovocJava(){};
        void setEurovocTree(std::string eurovocPath);
        std::vector<int> EurovocJava::getElementPath( int id);
        std::vector<EurovocElemJava *> getKids(int id, bool hr );
        std::vector<EurovocElemJava *> getRootKids( bool hr );
        std::string getDescription(int id, bool hr );
        bool hasKids(int id);
        TMT::Eurovoc *getEurovoc();
private:
        TMT::Eurovoc *eurovoc;
};


%template(vectorEuroElem) std::vector<EurovocElemJava *>;
%template(vectorInt) std::vector<int>;

%template(pairCW) std::vector < std::string >;
```

Slika 5.5. *Prikaz korištenja specifičnog tipa podatka std::vector<EurovocElemJava *> u ulaznoj datoteci WebJava.i.*

Samo neke od mogućnosti C++ programskog jezika nisu u potpunosti podržane u SWIG-u (virtualne metode i generiranje omotača za ugnježđene klase).

5.4. Web tehnologije

Iako web tehnologije ne omogućuju toliko naprednu izradu sučelja kao standardne „desktop“ aplikacije, u zadnjih par godina su se pojavile metode koje tu razliku maksimalno umanjuju.

5.4.1. HTML

„Hypertext Markup Language“ je tekstualni format zapisa dokumenta, prepoznatljiv web preglednicima. Web preglednici prepoznaju određene dijelove teksta i vizualno ih predočavaju korisnicima [33].

Npr. element:

<input width=“150px“ type="submit" value="Index">

predstavlja input elementa tipa „submit“ što pregledniku govori da ga prikaže kao gumb a vrijednost atributa *„value“* govori pregledniku koji tekst da napiše na gumb te vrijednost atributa *„width“* predstavlja širinu gumba. Tako se pomoću elemenata (*input*) i njihovih atributa (*type, value, width*) u HTML-u opiše izgled web stranice.

Osnovna struktura HTML stranice je sljedeća:

```
<html>
  <head></head>
  <body></body>
</html>
```

Unutar *<head>* elementa se upisuju metatagovi, naslov stranice, uključuju se JavaScripte, CSS – ovi i sl. dok u element *<body>* dolaze elementi koji predstavljaju sadržaj stranice.

5.4.2. CSS

„Cascading Style Sheets“ je jezik koji se koristi prilikom opisivanja prezentacijskog dijela HTML dokumenta ili nekog sličnog formata. Također prepoznatljiv web preglednicima i koristi se za opisivanje izgleda HTML elemenata, npr. definiranje boje, fonta, visine, širine i ostalih aspekata vizualnog izgleda. CSS je prvenstveno dizajniran da se može napraviti podjela između sadržaja i prezentacije. Do tada se izgled HTML elemenata opisivao pomoću atributa elemenata i nije se mogao ponovno iskoristiti već jednom napisani dio programskog koda, npr.izgled gumba. To je naravno i povećavalo veličinu dokumenata, tako da je uvođenjem CSS-a popravljena dostupnost sadržaja, omogućena veća fleksibilnost i kontrola prilikom specificiranja prezentacijskih karakteristika i smanjena je kompleksnost i ponavljanje u strukturi [33].

Uključivanje CSS – a u HTML stranice se može vršiti na više načina, no najbolje je izdvojiti CSS stilove u posebne dokumente te ih uključiti u HTML stranicu sljedećom linijom koda:

<link type="text/css" rel="stylesheet" href="css/design.css"/>

koja se upiše u *<head>* element HTML stranice. Unutar datoteke *„design.css“* se zatim pišu CSS stilovi, npr.

```
.example {
  font: bold 2.5em "Arial", Sans-Serif;
  margin: 0;
  letter-spacing: -1px;
}
```

koji se mogu dodavati HTML elementima preko atributa *„class“* :

* *

Na ovaj način je omogućeno korištenje jednog sila na više mjesta, smanjena količina texta koje korisnik mora učitati, HTML stranice postaju preglednije [28].

5.4.3. DOM

„*Document Object Model*“ sadrži strukturu XML/HTML dokumenata. Naime, prilikom učitavanja web stranice na klijentskoj strani web preglednici pohranjuju XML/HTML dokument u memoriju tako da izgrade hijerarhijsku strukturu, stablo od svih elemenata dokumenta. Na takav način se dobije sučelje koje omogućuje pristup elementima XML/HTML dokumenta koje koriste korisnički skriptni jezici poput JavaScript i Vbscript jezika.

Na vrhu hijerarhije se nalazi objekt *window* koji predstavlja prozor preglednika u kojem je otvorena web stranica. On omogućuje manipulaciju sa navigacijom, povijesti, lokacijom trenutno otvorenog prozora. Najvažniji objekt je objekt *document* koji predstavlja sadržaj web stranice. Unutar njega se nalaze svi elementi web stranice, počevši od elementa <*head*> i <*body*> pa prema svim njihovim unutarnjim elementima, slika 5.6 [28].

Slika 5.6. Hijerarhija objekata u DOM - u

5.4.4. JavaScript

JavaScript je skriptni jezik koji se izvršava na klijentskoj strani. Postoji više načina za uključivanje JavaScript koda unutar XML/HTML stranica, no u praksi se pokazalo da je najbolje izdvojiti JavaScript funkcije u zasebne datoteke i onda ih uključiti u stranicu, nešto poput „#include“ u C, C++ i sličnim programskim jezicima [34].

Uključivanje JavaScript-a se vrši dodavanjem sljedeće linije koda u *<head>* element HTML stranice, kao i prilikom dodavanja CSS datoteka:

<script language="JavaScript" type="text/JavaScript" src="js/test.js"></script>

JavaScript-om se može pristupati DOM elementima i ima definirane funkcije za manipulaciju DOM elementima, npr. dohvaćanje elementa s id-em 102:

var elem_102 = document.getElementById(„102“);

U JavaScripti postoje razne funkcije koje omogućuju manipulaciju dohvaćenim elementom. Može se kopirati, pristupati njegovoj djeci, roditeljima, mijenjati sadržaj, atributi, stilovi, itd. Npr. ako želimo promijeniti elementu CSS stil, odnosno njegov atribut koji mu određuje širinu to ćemo uraditi na sljedeći način:

elem_102.style.width = „200px“;

5.4.5. AJAX

„Asynchronous JavaScript and XML“ je tehnika izrade web aplikacija koja se koristi za kreiranje interaktivnih web aplikacija. Web aplikacije se čine puno interaktivnije tako da se razmjenjuju male količine podataka sa serverom u pozadini, tako da se cijela web stranica ne mora ponovno učitavati svaki put kada korisnik napravi neku akciju/promjenu. Povećava interaktivnost web stranica/aplikacija, brzinu, funkcionalnost i iskoristivost.

AJAX je asinkroni HTTP zahtjeva jer se AJAX učitavanje ne miješa sa normalnim učitavanjem stranica. Iz JavaScript programskog jezika se pozivaju AJAX funkcije, dok se podaci koji se se vraćaju uglavnom formatiraju u XML format i to pomoću XMLHttpRequest objekta, kojim je ostvaren AJAX poziv [28].

JavaScript funkcije za manipulaciju AJAX pozivima je dovoljno jednom napisati:

```
function loadAjax( url, callbackFunc, post_data, targetID){
   var xmlhttp = new XMLHttpRequest();
            if( post_data == null){
                xmlhttp.open( 'GET', url, true);
              }else{
                xmlhttp.open( 'POST', url, true);
```

```
            xmlhttp.setRequestHeader('Content-type', 'application/x-www-form-
        urlencoded');
            xmlhttp.setRequestHeader('Content-length', post_data.length);
            xmlhttp.setRequestHeader('Connection', 'close');
          }
          xmlhttp.setRequestHeader( 'If-Modified-Since', 'Sat, 1 Jan 2000 00:00:00
        GMT');
          xmlhttp.onreadystatechange = function(){
            if(xmlhttp.readyState == 4)
              callbackFunc( xmlhttp.responseText, targetID);
          }
          xmlhttp.send( post_data);
          return false;
}

function loadModuleContent( url, target){
  loadAjax( url, loadModuleContentCallback, null, target);
}

function loadModuleContentCallback( loadedAjax, target) {
   if(target != null)
      document.getElementById(target).innerHTML = loadedAjax;
}
```

Ove tri funkcije omogućavaju gotovo sve AJAX mogućnosti, bar one najvažnije. Prva funkcija ima 4 parametra:

1. URL za koji se želi napraviti AJAX poziv
2. callback funkcija koja se poziva nakon završenog učitavanja i kojoj se predaju dva parametra
    a. xmlhttp.responseText - učitani text
    b. targetID – element u koji se želi spremiti učitani tekst
3. post_data – koji ako je *null* onda otvara GET request a inače POST
4. targetID – koji se proslijedi callback funkciji

Može se raditi za svaki AJAX poziv posebna *callback* funkcija ili se može koristiti standardna, treća u gore navedenim funkcijama, tako da se učitavanje podataka sa *URL* – a u element sa id = *targetID* može napraviti jednostavnim pozivom druge funkcije:

loadModuleContent(URL, targetID);

AJAX je također i neovisan o platformi i iskoristiv na svim operacijskim sustavima, kompjuterskim arhitekturama i web preglednicima jer je temeljen na otvorenim standardima kao i JavaScript i XML.

5.4.6. JSP

„Java Server Pages“ je Java tehnologija koja omogućuje ubacivanje poziva Java funkcija, metoda, klasa, unutar HTML koda.

Unutar JSP stranice se mogu nalazizi HTML elementi i/ili JSP elementi. Standardni JSP elementi su *direktive, deklaracije, izrazi i skriptleti*.

Direktive se uključuju na vrh stranice i sadrže posebne instrukcije za web server. Najčešća direktiva je *page* direktiva, npr.

<%@ page import=“java.util.Date, java.io.“ %>*

Ova *page* direktiva kaže web serveru da uključi *java.util.Date* klasu i *java.io* paket, i onda možemo koristiti te klase unutar drugih JSP elemenata.

Deklaracije služe za deklariranje metoda i varijabli. Npr.

<%! int godine = 23; %>

<%! public int getGodine(){

return godine;

}%>

Ono što se deklarira unutar ovih JSP elemenata je vidljivo ostatku JSP stranice.

Izraz je element koji kaže web serveru da prije nego što vrati rezultat korisniku *izraz* zamijeni rezultatom koji on vraća. Npr.

....

* Ja imam <%= getGodine() %> godina. *

....

Ova linija će se prikazati u pregledniku kao *„Ja imam 23 godine.“* jer će web server prije nego vrati rezultat korisniku izračunati sve *izraze* i uglaviti ih u rezultat.

Skriptlet element se može koristiti bilo gdje unutar JSP stranice. *Skriptlet* predstavlja Java kod okružen JSP elementom <% %>. Npr.

```
<%  for(int i=0; i<10;i++){

out.println(i);

} %>
```

6. Aplikacijsko rješenje sustava WEB eCADIS

6.1. Opis Sustava

WEB eCADIS sustav podržava većinu funkcionalnosti desktop aplikacije eCADIS, pa je „mozak“ i ovog sustava TMT biblioteka. Izvorni kod TMT biblioteke nije izravno korišten zbog mogućnosti nastanka pogrešaka pri prenosu različitih struktura podataka kroz JNI sloj. Kao što je već rečeno, JNI podržava većinu podataka, ali radi sigurnosti, napravljeno je nekoliko klasa na C++ sloju koje služe kao sučelje prema Javi, naravno preko JNI sučelja. U tih nekoliko klasa su implementirane sve funkcionalnosti koje omogućuju rad WEB eCadis aplikacije.

Dalje, bilo je potrebno pretvoriti klase koje koriste funkcionalnosti TMT biblioteke u .DLL (*eng.* Dinamic Link Library) za Windowse ili u .SO za Unix/Linux operacijske sustave. U tom slučaju dolazi do izražaja CMAKE razvojni alat, koji za određene klase napisane u C++ programskom jeziku, pravi .dll biblioteku koja se kasnije uglavljuje u Java programski kod za vrijeme njegovog izvršavanja.

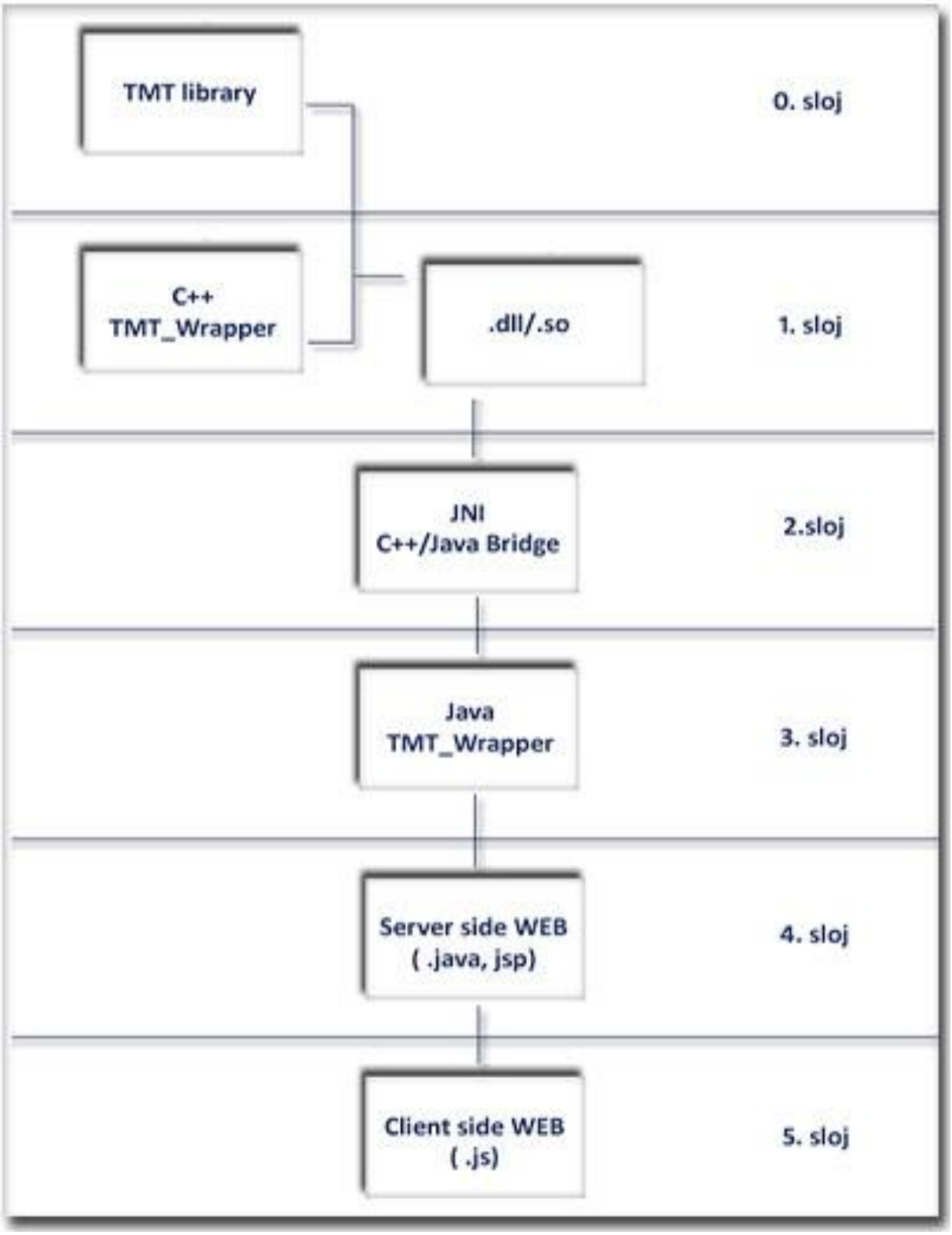
Za prijenos podataka između C++ sloja i Java sloja služi izgradnja JNI programskog koda pomoću SWIG alata. Već prije spomenuto, SWIG-u se preko konfiguracijskih .i file-ova daju naredbe koje mu kažu za koje klase će izgenerirati Java klase i JNI kod za komunikaciju s dll-om.Tada dolazi na red implementacija metoda na Java sloju, testiranje aplikacije i izgradnja prezentacijskog sloja.

Vjerojatno se ovaj proces običnom promatraču može učiniti kompliciranim, ali zahtjevi da se aplikacija može izvršavati na različitim operacijskim sustavima je rezultitrala ovakvim izborom rješenja. Prilikom izgradnje dll-a i Java-prezentacijskog sloja preporuča se testiranje implementiranih funkcija na C++ sloju i Java sloju. Ova aplikacija je na taj način i testirana, da bi se osigurao siguran prijenos podataka preko JNI sloja. JNI programski kod je dosta nečitljiv i bilo bi krajnje nesuvislo u njemu tražiti pogreške prenosa informacija. Stoga je najbolje izraditi testni program na C++ sloju i testni program na Java sloju, ispitati sve implementirane metode te dobivene rezultate uspoređivati.

Naravno, čitav ovaj posao se mogao odraditi i na Java sloju, tj. pozivati izravno funkcije iz TMT biblioteke na Java strani. U tom slučaju, aplikacija bi se mnogo sporije izvršavala a i upitno je da li SWIG podržava sve tipove podataka iz C++ prog. jezika (pokazivači, pokazivači na funkcije i dr.). Programski JNI kod koji SWIG izgenerira ograničava mogućnosti prijenosa podataka, a zbog svoje nečitljivosti i složenosti nije bilo prihvatljivo ručno pisanje JNI koda.

Stoga se kao najlogičniji i najsigurniji izbor činio upravo ovakav način implementacije. Kompleksnost tipova podataka i metoda iz TMT biblioteke su na C++ strani dovedene na minimum i od tih klasa je napravljen .dll koji se naknadno uglavljuje u Java

programski kod. Nakon sažetog opisa razvoja aplikacijskog web rješenja, slijedi detaljniji opis razvoja aplikacije po slojevima.



Slika 6.1. *Prikaz strukture WEB eCADIS aplikacije*

6.2. Sloj C++

TMT biblioteka je već opisana u ranijim poglavljima, tako da će se detaljno opisati samo sloj u kojem su nastajale klase koje su koristile za prilagodbu funkcionalnosti TMT biblioteke. Sve funkcionalnosti potrebne za rad web aplikacije su napisane u *WebJava.cpp* datoteci [28]. Unutar nje su implementirane četiri klase:

- EurovocElemJava
- EurovocJava
- ClassifierJava
- PairOfStrings

Prve dvije klase služe za rad s Eurovoc pojmovnikom. *EurovocJava* klasa prestavlja strukturu Eurovoc pojmovnika i ima sve potrebne metode za dohvat elemenata Eurovoc pojmovnika, dohvat dodatnih informacija o pojedinom elementu (čvoru stabla), dohvat svih užih, širih, i relacijski povezanih pojmova sa pojedinim čvorom stabla. Klasa *EurovocElemJava* je sastavni dio klase EurovocJava jer označava element Eurovoc pojmovnika koji je predstavljen svojim id-om i nazivom. Indentifikacija pojedinog elementa u stablu se upravo vrši preko njegovog id-a. Metode koje se nalaze u klasi EurovocJava su dolje navedene:

```
EurovocJava(const std::string &eurovocPath);
        // konstruktor
~EurovocJava();
        // destruktor
void setEurovocTree(const std::string &eurovocPath);
        //metoda za učitavanje Eurovoc stabla kojoj se predaje putanja Eurovoc stabla
std::vector<int> getElementPath(int id) const;
        //dohvaća putanju do određenog čvora u stablu
std::vector<EurovocElemJava> getKids(int id, Language language) const;
        //dohvaća djecu nekog čvora
std::vector<EurovocElemJava> getRootKids(Language language) const;
        //dohvaća 0-tu razinu Eurovoc stabla
std::string getDescription(int id, Language language) const;
        //dohvaća opis pojedinog elementa
std::vector<PairOfStrings> getAllTerms(int id, Language language) const;
        //dohvat svih pojmova vezanih za element
bool hasKids(int id) const;
        //provjera da li je element u zadnjoj razini Eurovoc stabla
TMT::Eurovoc *getEurovoc() const;
        //učitavanje Eurovoc-a
```

Klasa *ClassifierJava* implementira metode TMT biblioteke za prikaz, indeksiranje teksta i za dobivanje rezultata indeksiranja.

```
ClassifierJava();
  // konstruktor
~ClassifierJava(){};
        // destruktor
void setLematizator(std::string lemmatizationPath, std::string stopWordsPath);
        //postavlja lematizator
void setClassifierData( std::string classifierPath, std::string featureBuilderPath);
        //postavlja podatke bitne za indeksiranje
void classify(bool language, std::string text, EurovocJava *eur);
        //indeksira dani tekst
```

Ovakve metode postoje i u TMT biblioteci, samo što su one vraćale vrijednosti preko referenci, a u našem slučaju, zbog zahtjeva SWIG-a, rezultati se vraćaju preko jednostavnih tipova podataka(vektor, string, int) ili preko jednostavnih klasa. Ovaj TMT_wrapper upravo služi tome da se izbjegne povrat rezultata peko složenih tiova podataka. Sve ove metode u sebi pozivaju izvorne metode i klase iz TMT biblioteke, te se unutar istih, rezultati pohranjuju u privatne varijable:

```
TMT::Lemmatization *lemmatization;
ClassifierData *classifierData;
XMLDocumentSpanifier *spanifier;
std::vector<PairOfStrings *> suggestedDescriptors;
std::vector< PairOfStrings *> descriptors;
```

Redosljed poziva funkcija za inicijalizaciju kasifikatora je sljedeći:

```
cj.setLematizator("vjesnik-10m.molex-3.3-1-ex-2way-utf8.fsa","hrvatski_stoprijeci.txt");
        //učitavanje lematizatora sa parametrima seta dokumenata i stop riječi za hrv. jezik
cj.setClassifierData("nn-classifier.bin", "nn-fb.bin");
        //učitavanje klasifikatora
cj.classify(LanguageCroatian, "1996_0997-d.xml", &e);
        //klasificiranje dokumenta
```

Preko JNI mosta Java sloju je izloženo, na ovom primjeru, tri ulazne metode setLemmatizator, setClassifierData, classify i tri metode kojima se mogu dohvatiti vrijednosti getSuggestedDescriptors, getIndexDescriptors, getTextXML.

Klasa PairOfStrings je napravljena radi lakšeg rada s vektorima i poljem vektora na Java sloju.

U ovom sloju je još navedeno stvaranje .dll datoteke od klasa izgrađenih na C++ sloju. Taj proces se odvija pokretanjem i izvršavanjem projekta kojeg je napravio CMAKE alat. U ovom slučaju, kreirao se projekt za .NET okruženje.

6.3. Sloj JNI (most između C++ i Java arhitekture)

Omotač (JNI kod) za C++ kod, točnije za dll/so generira SWIG program. Pokreće se iz komandne linije ili se može uključiti u razvojno okruženje ako razvojno okruženje to dozvoljava (drugi sloj prikazan na slici 6.1.)

Kao što je već rečeno u prethodnom poglavlju, za korištenje SWIG-a je potrebno napisati konfiguracijsku datoteku sa ekstenzijom .i koja se predaje SWIG-u i on na osnovu te konfiguracijske datoteke i DLL/SO-a izgenerira omotač.

Kako SWIG nije program koji generira samo JNI wrapper za premošćivanje Jave i C++, nego služi za premošćivanje raznih viših programskih jezika (skriptnih, interpretiranih i sl.) i jezika što se prevode izravno u strojni kod(*eng.* native, jezici poput C++ i C), zato je potrebno naglasiti prilikom SWIG poziva koji od viših programskih jezika se koristi. U ovom slučaju je to Java [28].

U slučaju da se SWIG poziva iz komandne linije naredbe su sljedeće:

- %swig -c++ -java example.i
- %g++ -c -fpic example.cxx
- %g++ -c -fpic example_wrap.cxx -I/usr/java/j2sdk1.4.1/include -I/usr/java/j2sdk1.4.1/include/linux
- %g++ -shared example.o example_wrap.o -o libexample.so

Prilikom svake izmjene na C++ sloju je bilo potrebno upisivati ove naredbe iznova, stoga je napravljena jedna skripta *rebuild_wrapper* čijim se pokretanjem ostvaruje poziv CMAKE programa i kreiranje projekta za .NET okruženje. Igled skripte *rebuild_wrapper* prikazan je na slici 6.2.

Za kreiranje .dll datoteke je potrebno otvoriti napravljeni projekt u .NET okruženju i pokrenuti ga. Kako .NET okruženje omogućuje uključivanje SWIG-a u projekt, u CMAKE datoteci su dodane naredbe koje uključuju SWIG u projekt te se izvršavanjem projekta pozove i SWIG koji izgenerira JNI most i Java omotač(Java klase). Na ovaj način se razvoj aplikacije i dodavanje novih funkcionalnosti pojednostavnio. Problem dobivanja funkcionalnosti TMT biblioteke u web tehnologiji je ovim riješen.

```
@echo off

pushd
cd /d %~dp0
if exist wj rmdir /s /q wj
mkdir wj
mkdir wj\wrappers
cd wj
rem cmake -G "MinGW Makefiles" -D CMAKE_BUILD_TYPE:STRING=Release ..\src
cmake -G "Visual Studio 8 2005" -D CMAKE_BUILD_TYPE:STRING=Release ..\src
popd
```

Slika 6.2. *Izgled skripte rebuild_wrapper. Njenim se pokretanjem ostvaruje poziv CMAKE programa i kreiranje projekta za .NET okruženje*

6.4. Java sloj

Funkcionalnost TMT biblioteke je izložena Javi preko Java klasa (omotača) izgeneriranih SWIG programom. Od ovog trenutka razvoj WEB eCadis aplikacije se mogao u potpunosti nastaviti u Java tehnologiji. Osim ako je potrbeno novu funkcionalnost TMT biblioteke izložiti Javi, u tom slučaju je potrebno tu funkcionalnost implementirati u C++ omotaču i ponoviti postupak iz prethodnih poglavlja.

Java sloj je podijeljen u tri sloja koja su opisana u ovom poglavlju (treći, četvrti i peti sloj arhitekture web aplikacije sa slike 6.1). Nakon dobivenih funkcionalnosti TMT biblioteke u obliku Java klasa, WEB eCadis aplikacija se mogla razvijati kao i svaka druga web aplikacija. Pomoću Java klasa se radi računanje i pozivanje TMT biblioteke, pomoću JSP datoteka se pripremaju rezultati računanja za prikaz u korisnikovom pregledniku, a pomoću JavaScript funkcija se omogućuje korisniku bolji vizualni dojam WEB eCadis aplikacije.

6.4.1. Java klase (treći sloj arhitekture)

U ovaj sloj spadaju klase izgenerirane SWIG programom i još dvije klase:

- Eurovoc.java
- Classifier.java

U tim klasama su implementirani pozivi SWIG-om izgeneriranih klasa koje su preslike C++ klasa tj. preko statičkih metoda izlažu JSP datotekama pozive C++ metoda.

Metode koje su implemetirane u Eurovoc.java su:

```
public static synchronized void initialize(String path)
    //inicijalizacija Eurovoc pojmovnika
```

```
public static EurovocJava getEurovocJava()
    //dohvat Eurovoc pojmovnika
public static boolean hasKids(int id)
    //za provjeru da li je element u zadnjoj razini Eurovoc stabla
public static ArrayList getDescription(int id, Language lang)
    //dohvat opisa Eurovoc elementa (pojedinog čvora u stablu)
public static VectorPairStrings getTerms(int id, Language lang)
    //dohvat užih, širih i relacijskih pojmova nekog elementa
public static HashMap getKids(int id, Language lang)
   //dohvat sve djece nekog elementa
public static ArrayList<Integer> getElemPath(int id)
    //dohvat putanje do određenog elementa u stablu
```

Metode koje su implemetirane u Classifier.java su:

```
public static synchronized void initialize(String path)
      //inicijalizacija klasifikatora i lematizatora za hrvatski i engleski jezik
private static ClassifierJava getClassifier(Language lang)
      //dohvat klasifikatora za zadani jezik
public static boolean Classify(String filePath, Language lang)
      //poziv kasifikacije za zadani dokument
public static VectorPairStrings getResult(Language lang)
      //dohvat deskriptora koje automatski indeksator predlaže za taj dokument
public static VectorPairStrings indexDescriptora(Language lang)
      //dohvat deskriptora nađenih u tekstu i pripadnih frekvencija pojavljivanja
public static String getSpanifiedText(Language lang)
      //dohvat učitanog teksta s označenim elementima unutar njega
```

Ono što je bitno u ovim klasama je sam proces inicijalizacije Eurovoc pojmovnika i učitavanje lematizatora i klasifikatora. Pošto je WEB eCADIS web aplikacija, trebalo je računati da će u istom trenutku njoj pristupati više korisnika. Korisnici bi koristili istu aplikaciju, ali za svakog od njih bi se kreirala nova instanca Eurovoc pojmovnika, kao i lematizatora i klasifikatora. U tom slučaju, aplikacija bi sporije izvršavala zahtjeve korisnika i jednom trenutku bi bilo nemoguće raditi na njoj. Uvelo se rješenje da korisnici uvijek koriste iste metode objekata (u Javi nazvane *static metode*) Eurovoc pojmovnika, lematizatora i klasifikatora. Nakon provedenog testiranja, uvidjelo se da to rješenje nije dovoljno dobro te je uvedena *synchronized* metoda. Dva razloga zbog čega je ova vrata metode dobra su:

- Nije moguće da se dva ili više poziva *synchronized* metode isprepliću. Kada jedna dretva izvršava *synchronized* metodu, sve ostale dretve koje pristupaju istoj metodi su blokirane dok se ova koja je prva pozvala metodu ne izvrši.

- Promjene koje je *synchronized* metoda napravila su odmah vidljive svim ostalim dretvama.

Kada jedna dretva pozove *initialize* metodu, sve ostale dretve će čekati dok ona ne završi s učitavanjem lematizatora i klasifikatora za hrvatski i engleski jezik, nakon čega postavlja varijablu initialized na true vrijednost. Nakon toga, ostale dretve koje budu htjele pozvati istu ovu metodu, doći će do linije koda gdje se provjerava stanje varijable initialized. U tom trenutku će izaći iz petlje i izbjeći će se ponovno učitavanje klasifikatora i lematizatora (neće se usporiti rad apliakcije).

```
import java.io.FileWriter;

public class Classifier {

    private static boolean initialized = false;
    private static ClassifierJava cj_hr, cj_en;
    private static String extra = null;

    public static synchronized void initialize(String path) {
        if (!initialized){
            extra = path;
            cj_hr = new ClassifierJava();
            cj_hr.setLematizator(extra + File.separator + "vjesnik-10m.molex-3.3-1-
            cj_hr.setClassifierData(extra + File.separator + "nn-classifier.bin", e

            cj_en = new ClassifierJava();
            cj_en.setLematizator(extra + File.separator + "engleski_rjecnik-2way.fs
            cj_en.setClassifierData(extra + File.separator + "nn-classifier.bin", e

            initialized = true;
        }
    }
```

Slika 6.3. *Primjer synchronized metode za inicijalizaciju lematizatora i klasifikatora*

6.4.2. JSP (četvrti sloj arhitekture)

JSP sloj se može podijeliti na dva dijela:

- Prezentacijske JSP datoteke
- JSP datoteke koje pozivom Java metoda uzimaju podatke i formatiraju ih u HTML/XML strukturu pogodnu za prikaz u WEB preglednicima

U prezentacijske JSP datoteke spadaju *eCadis.jsp*, *Home.jsp* i svi ostali JSP – ovi koji predstavljaju kompletne stranice ili dijelove stranica. One uglavnom sadržaju elemente HTML-a i CSS-a (*eng. Cascading Style Sheets*) zaslužene za izgled aplikacije. Unutar njih se pozivaju ostale JSP stranice i JavaScripte koje izvršavaju zadane operacije i vraćaju rezultat koji se potom uglavljuje u HTML kod prezentacijske JSP stranice.

U drugu skupinu JSP datoteka spadaju sve ostale JSP datoteke koje pozivom Java metoda dobivaju potrebne podatke i pripremaju ih za prikaz. Većina tih JSP datoteka se

poziva s JavaScript sloja pomoću AJAX tehnologije, te se rezultati uključe u sadržaj stranice pomoću JavaScript – i koje se izvršavaju na klijentskoj strani.

Na primjeru ćemo pokazati poziv JSP datoteke koje pozivom Java metoda uzimaju podatke i formatiraju ih u HTML/XML strukturu pogodnu za prikaz u web preglednicima. Na temelju id-a elementa dobivenog putem zahtjeva, poziva se metoda iz Eurovoc.java klase *getElemPath* kojoj se preda taj id i ona vrati polje id-a elemenata koji su prethodili tom elementu. Povratna vrijednost (putanja do određenog čvora u stablu) će se spremiti u *String* tip podatka i kao takva vratiti prezentacijskom JSP koji ju je pozvao.

```
<%@ page import="java.util.ArrayList, org.fer.tmt.*" %>

  <%
  request.setCharacterEncoding("UTF-8");
  int pom = Integer.parseInt(request.getParameter("id").trim());
  response.setContentType("text/xml;charset=UTF-8");
  ArrayList djeca;
  djeca = Eurovoc.getElemPath(pom);

  String value = djeca.get(djeca.size()-1).toString();
  for(int i=djeca.size()-2; i>=0; i--){
      value += "," + djeca.get(i);
  }

  %>
  <%=value%>
```

***Slika 6.4**. Prikaz datoteke jfGetEuThesaurusElementPath.jsp.*

6.4.3. JavaScript (peti sloj arhitekture)

Bilo koja akcija koja je omogućena korisniku i koja zahtijeva komunikaciju sa serverom je napravljena pomoću asinkronog HTTP zahtjeva (AJAX) i na taj način se izbjeglo konstantno osvježavanje stranice/a i omogućeno je da se cijela aplikacija napravi kao jedna web stranica. Svaka korisnikova akcija poziva određenu JavaScript funkciju koja je zadužena za tu akciju. Ako je potrebna komunikacija sa serverom JavaScript funkcija je odradi [28].

Na primjeru ćemo pokazati jedan takav poziv JavaScripte iz JSP datoteke za otvaranje grane stabla kada korisnik klikne na ikonu za otvaranje Eurovoc stabla.

Upotreba AJAX tehnologije je omogućila prikaz Eurovoc stabla. Dohvat svih elemenata Eurvoc stabla, prilikom otvaranja Eurovoc prozora bi bio jako spor i predugo bi se

učitavao na korisničkoj strani. Da bi se izbjegao taj problem i problem ponovnog učitavanja čitave strukture Eurovoc stabla, napravljena je JavaScript funkcija koja je zadužena za pozivanje samo određenog nivoa stabla. Ta funkcija AJAX pozivom prima podatke sa servera preko posebno napravljene JSP datoteke. JSP datoteka je zadužena za poziv Java funkcija koje vraćaju Eurovoc elemente. Java funkcije pak komuniciraju s TMT bibliotekom na već prije opisan način.

```
function direktni(param){


    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open( 'POST', 'jfGetEuThesaurusElementPath.jsp?id=' + param, false);
    xmlhttp.send( null);

    pomocni = xmlhttp.responseText.split(',');

    if(pomocni.length==1){
        var strTMP = "" + parseInt(pomocni[0]);
        time = setTimeout("scrollToTreeNode('nodex" + strTMP  +"_a')", 700);
        return false;
    }if(pomocni.length>1)
      var path = parseInt(pomocni[1]);
    if(pomocni.length>2){
      for(var i=2;i<pomocni.length;i++){
        path +="," + parseInt(pomocni[i]);
      }
    }
```

Slika 6.5. *prikazano je pozivanje JavaScript funkcije za dohvat određenog elementa u stablu i pozicioniranje u Eurovoc prozoru na taj element.*

U prve tri linije programskog koda nalazi se poziv AJAX tehnologije. Dohvaća se samo taj element u stablu i putanja koja vodi do tog čvora. Slijedi pozicioniranje u Eurovoc prozoru na taj čvor sa pozivom funkcije scrollToTreeNode.

6.5. Dodavanje novih funkcionalnosti WEB eCADIS aplikaciji

Ako bi se htjela dodati nova funkcionalnost WEB eCADIS aplikaciji potrebno je napraviti sljedeće korake:

1. Doda se nova funkcionalnost - metoda u postojeću klasu u WebJava.cpp datoteci ili se napravi nova klasa u WebJava.cpp datoteci

2. Nova funkcionalnost se doda u .i konfiguracijsku datoteku za SWIG

3. Ako nisu dodani novi .cpp fileovi nije potrebno mijenjati Cmake konfiguracijsku datoteku, no ako je onda je potrebno uključiti novi file u projekt. (izmjeniti CMAKE listu)

4. Pokrene se Cmake program koji stvori projekt za operacijski sustav na kojem radite i razvojno okruženje u kojem radite

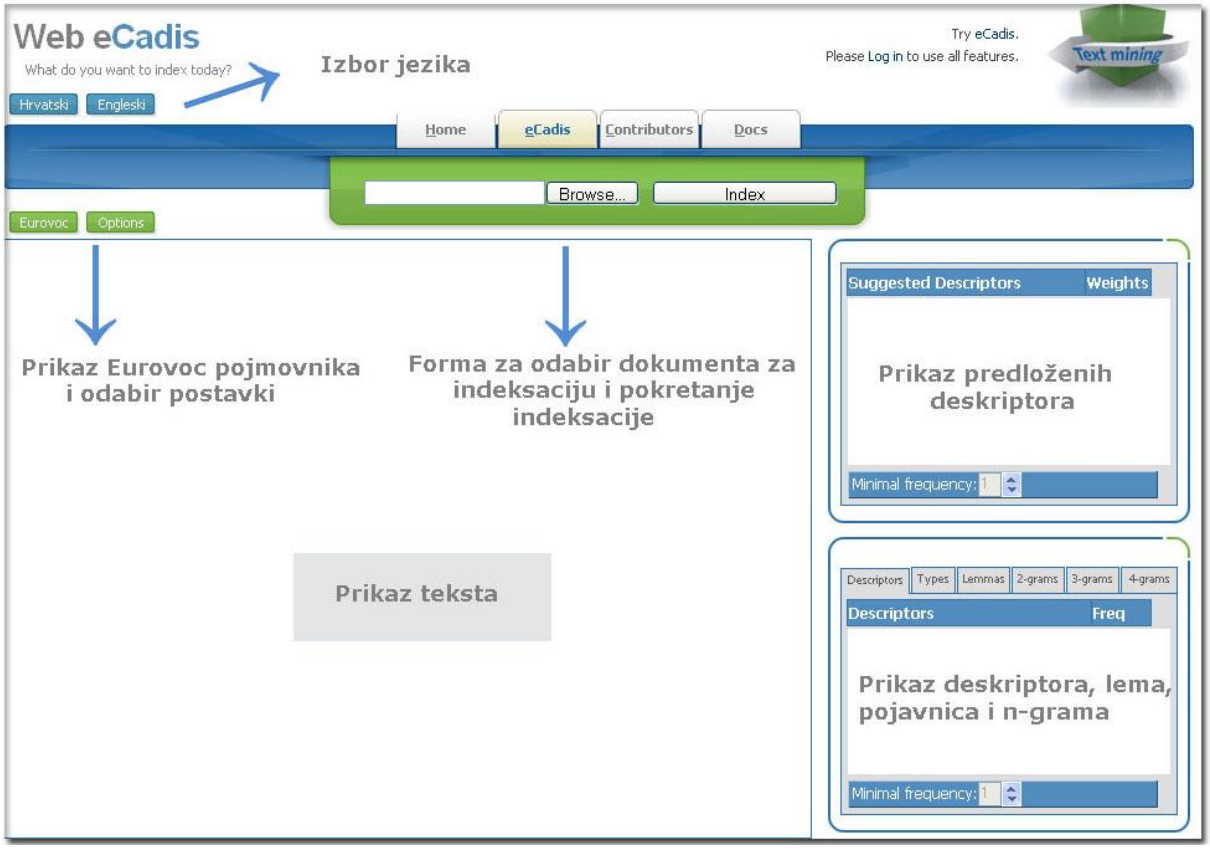
5. Pokrene se projekt iz razvojnog okruženja

6. Implementira se nova funkcionalnost na Java sloju

7. Napravi se prezentacijski dio nove funkcionalnosti

7. Implementacija podsustava za prikaz lema, n-grama, deskriptora i Eurovoc pojmovnika

Nakon uspješnog premošćivanja C++ i Jave, bilo je moguće nastaviti razvijati WEB eCADIS aplikaciju u Java web tehnologijama. Prikaz aplikacije i mogućnosti koje ona nudi korisnicima se vidi na slici 7.1.

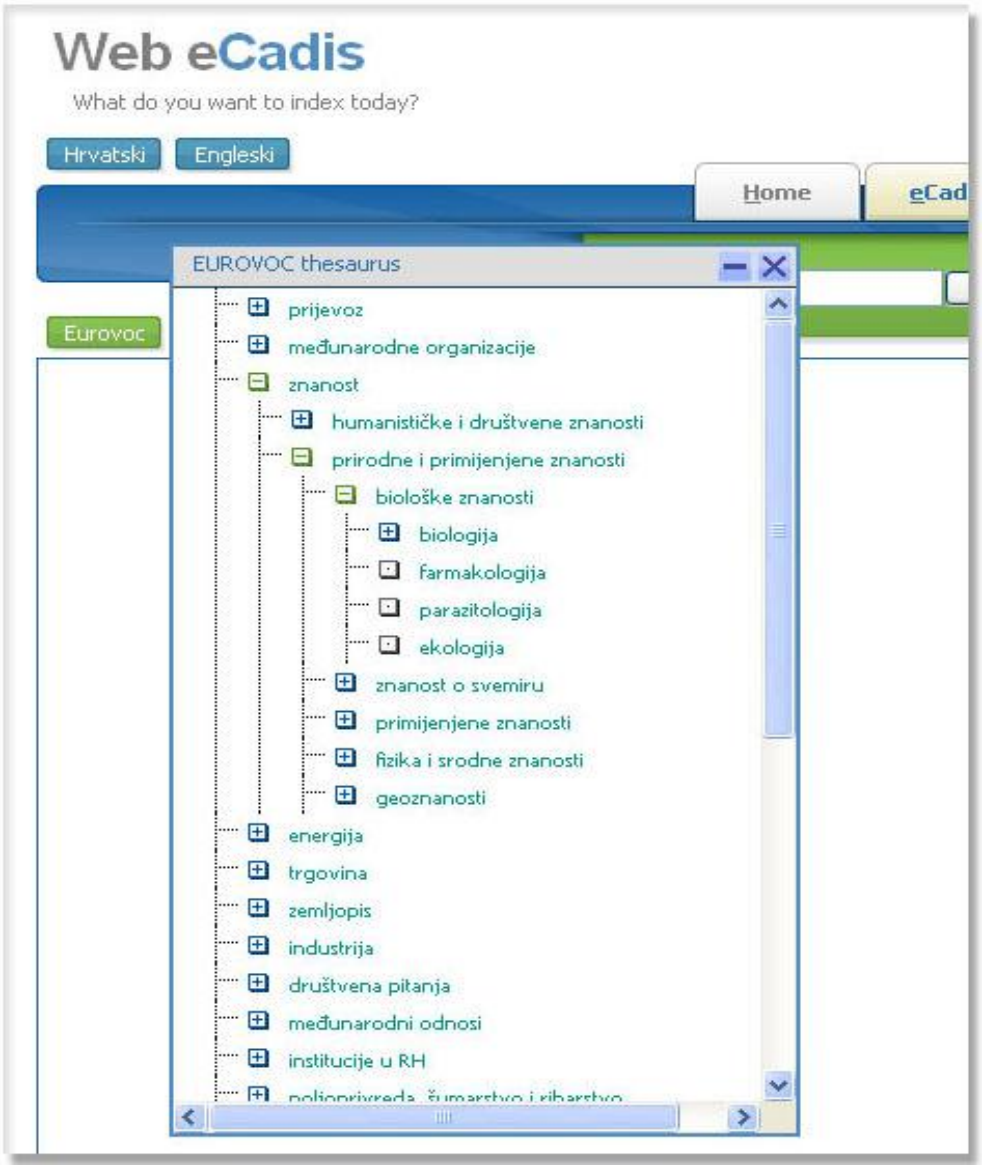


Slika 7.1. *Prikaz WEB eCADIS aplikacije*

WEB eCADIS aplikacija nudi izbor jezika sučelja i jezika indeksacije (trenutno podržava samo hrvatski i engleski jezik kao i PEI sustav). Prikaz Eurovoc pojmovnika je omogućen klikom na gumb „Eurovoc“ a izbor postavki na gumbu „Options“. Izbor postavki je bio aktivan u prošloj verziji WEB eCADIS sustava. Bilo je moguće mijenjati jezik indeksiranja, jezik prikaza, bilo je moguće mijenjati indeksatore, stop riječi i sve ostale parametre koje je moguće mijenjati u standardnoj verziji aplikacije. Zbog same namjene web aplikacije širem krugu korisnika i da se omogući lakše korištenje i shvaćanje biti ove aplikacije, izbor postavki je postavljen na izvorne postavke za indeksaciju dokumenata na englesko i hrvatskom jeziku.

7.1. Implementacija podsustava za prikaz Eurovoc pojmovnika

Početak razvoja WEB eCADIS sustava je bio prikaz Eurovoc pojmovnika kao temelja ove web aplikacije.

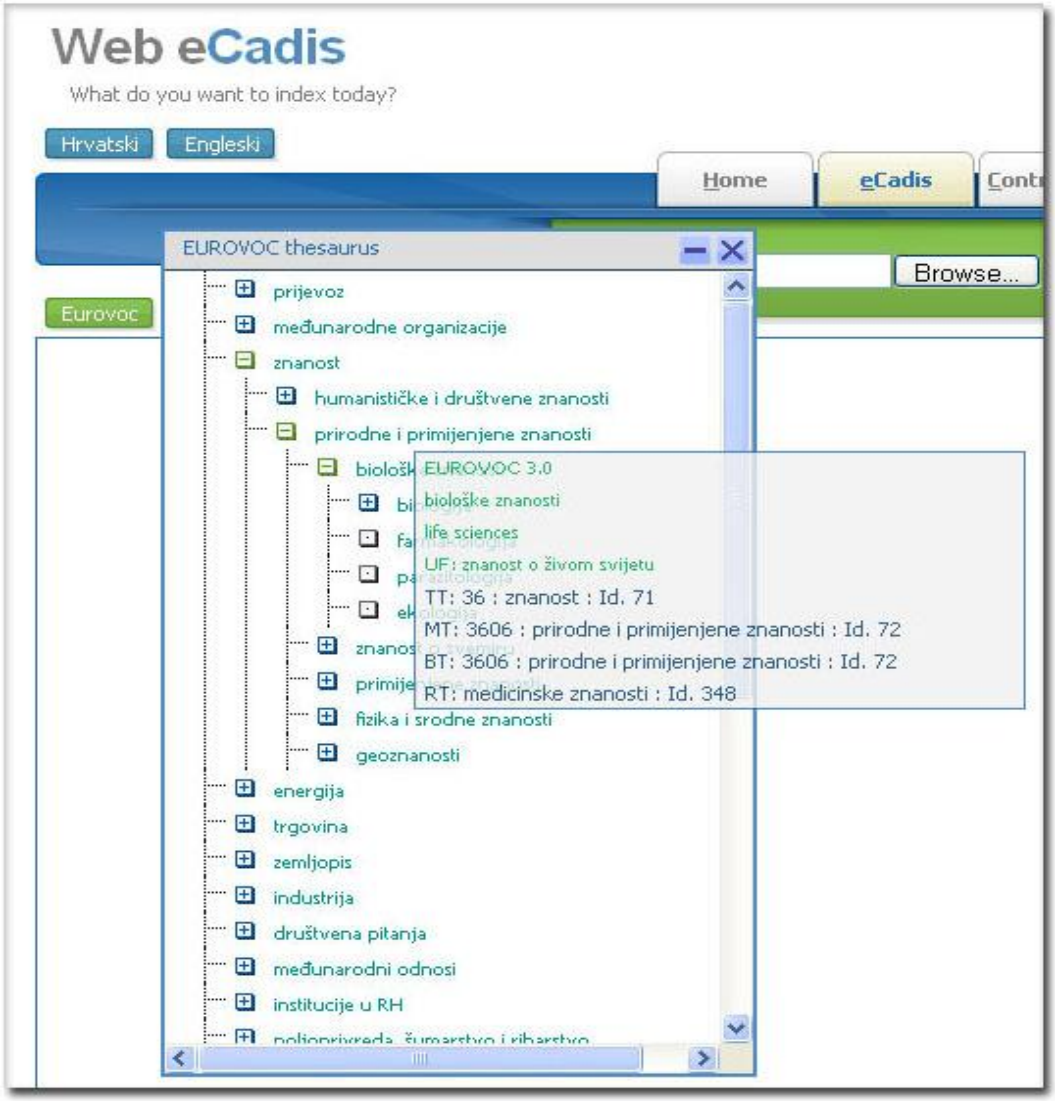


Slika 7.2. *Prikaz Eurovoc pojmovnika*

Prikaz Eurovoc pojmovnika u WEB eCADIS sustavu je dan slikom 7.2. gdje su otvoreni elementi pojmovnika znanost→prirodne i primjenjene znanosti→biološke znanosti. Na slici se vidi hijerarhijska struktura ostavrena među elementima pojmovnika, element znanost je širi pojam od pojma prirodne i primjenjene znanosti. Svi elementi koji imaju djecu

imaju pored svog naziva, sliku „+“, dok oni koji nemaju djecu (nalaze se u zadnjem sloju pojmovnika) imaju sliku „·“. Klikom na sliku „+“ prikazuju se djeca odabranog elementa, a klikom na sliku „-“ zatvara se grana stabla.

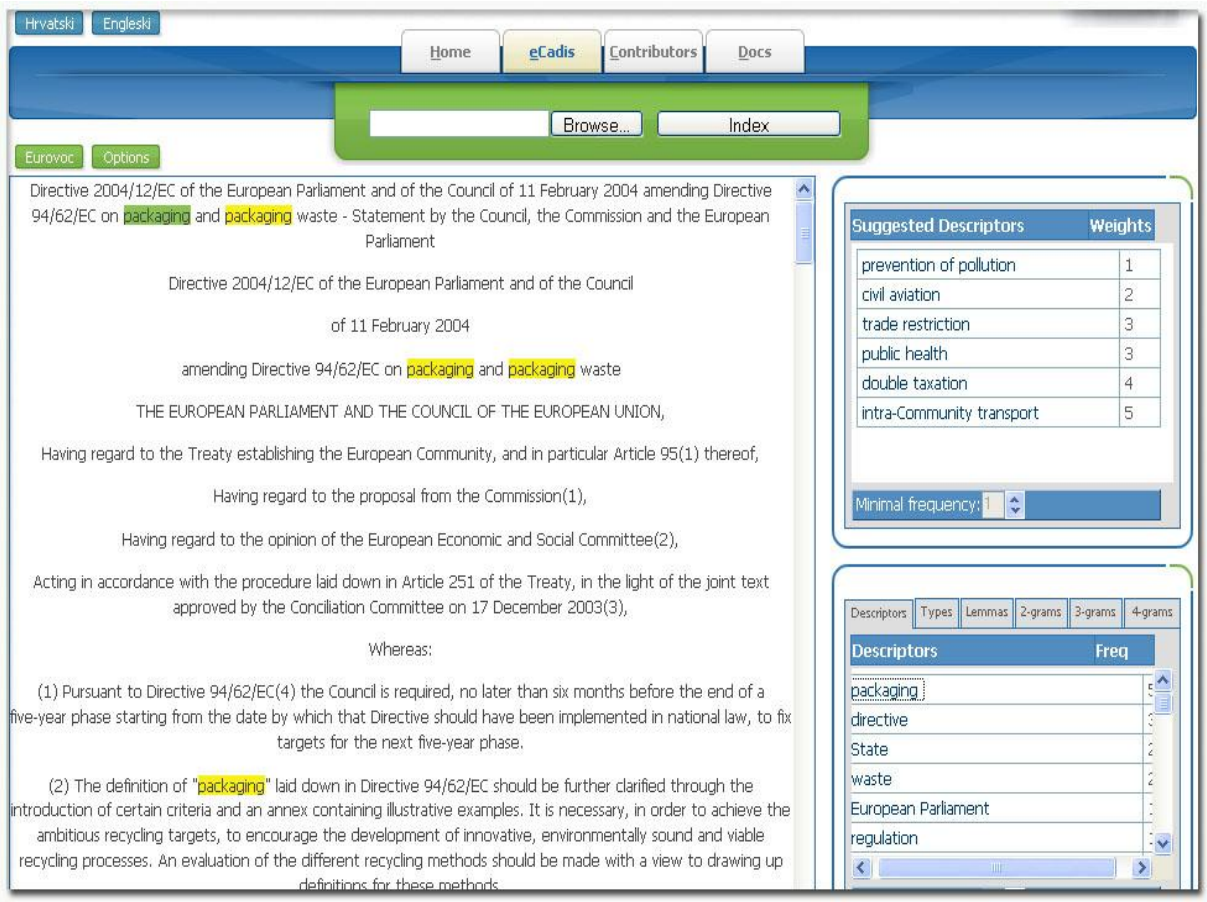
Prelaskom miša preko nekog elementa u stablu pojavljuje se prozor sa detaljnim opisom tog elementa. Opis elementa sadrži njegov naziv na hrvatskom i engleskom jeziku, uže, šire i ostale relacijski povezane pojmove (TT, MT, BT i RT). Svaki od tih pojmova je prikazan kao link, čijim se odabirom otvara element u stablu. Ovakav prikaz Eurovoc pojmovnika je omogućen upotrebom tehnologije AJAX koja omogućuje da se otvaranjem i zatvaranjem grana u stablu te dohvatom opisa elementa ne osvježava prozor u kojem se prikazuje Eurovoc pojmovnik čime je ubrzano prikazivanje Eurovoc pojmovnika i rad na njemu.



Slika 7.3. *Prikaz elementa u pojmovniku i njegovog detaljnog opisa te relacijskih elemenata.*

7.2. Implementacija podsustava za prikaz deskriptora, lema, pojavnica i n-grama

Forma za unos dokumenta za indekiranje (vidi sliku 7.1.) omogućuje izbor dokumenta koji će se klikom na gumb „Index“ indeksirati. Indeksirani tekst će se pojaviti u prozoru a rezultati indeksiranja na mjestu za prikaz predloženih deskriptora te mjestu za prikaz deskriptora, lema, pojavnica i n-grama. Primjer indeksiranja dokumenta te prikaz sobivenih rezultata vidi se na slici 7.4.



Slika 7.4. *Prikaz indeksiranog dokumenta i rezultata indeksiranja dokumenta engleskom jeziku*

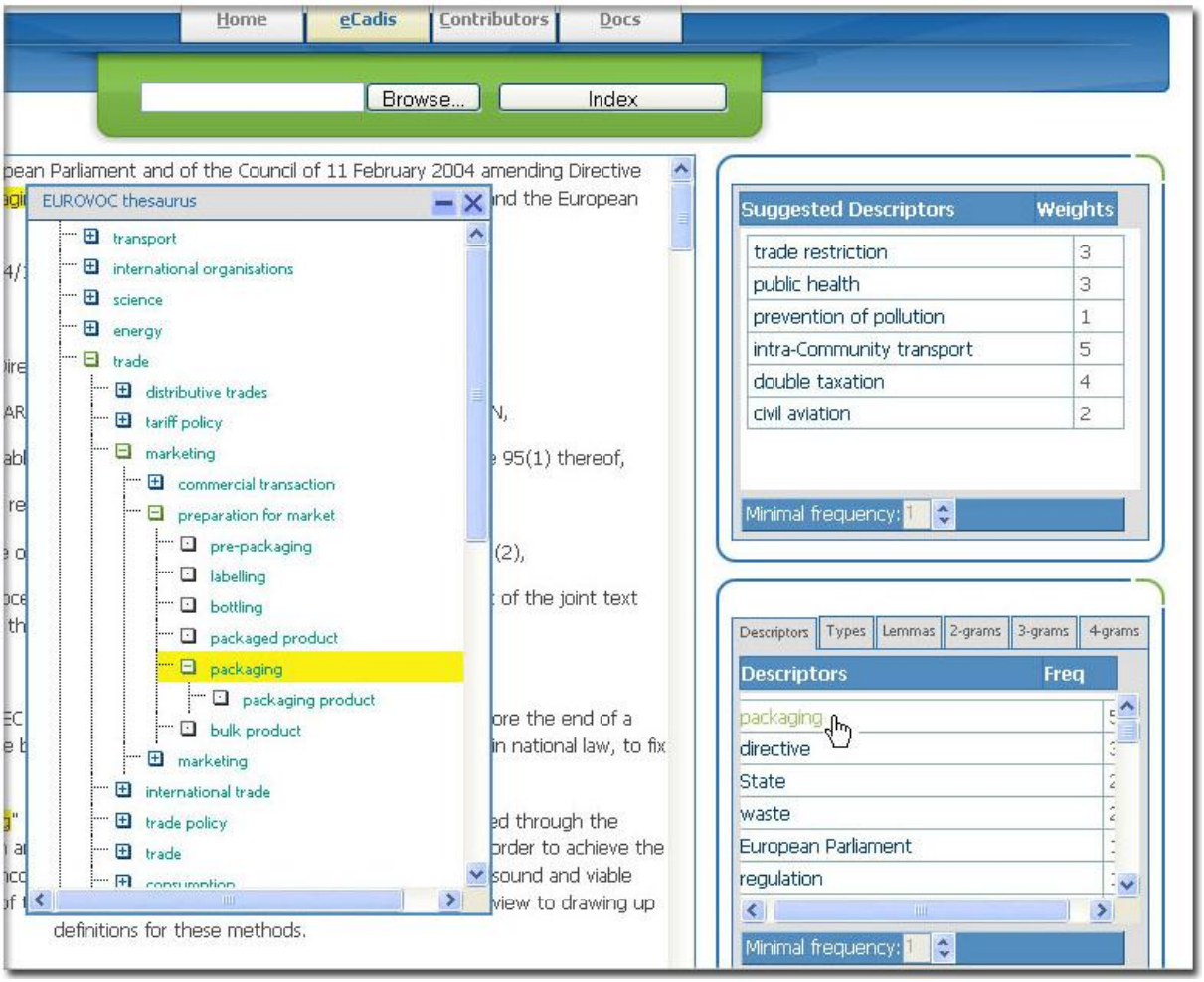
Na desnoj strani postoje dva prozora, jedan za prikaz predloženih deskriptora, drugi za prikaz deskriptora nađenih u tekstu, lema, pojavnica i n-grama.

Predloženi deskriptori su bit čitavog procesa automatskog indeksiranja. Ovisno o tome koliko je „dobar“ klasifikator odnosno koliko je dobro istreniran i naučen toliko će sustav dobro korisniku predlagati deskriptore. U tablici predloženih deskriptora se osim njihovih

naziva nalaze i težine koje o određenom deskriptoru u obliku postotka govore koliko je deskriptor „dobar“ za indeksirani dokument.

Ostali prikaz rezultata se temelji na učestalosti pojavljivanja riječi u tekstu, te se osim naziva riječi, prikazuje i frekvencija pojavljivanja u indeksiranom dokumentu. Povećanjem/smanjenjem minimalne frekvencije pojavljivanja (opcija ispod prikazanih rezultata) korigiraju se i rezultati u tablici. Također je implementirana funkcija za sortiranje rezultata abecedno i po broju pojavljivanja u tekstu. Odabirom rezultata u tablici (klikom na naziv riječi), u tekstu se oboje pojavljivanja te riječi, te se iterativno prelazi sa jednog pojavljivanja u tekstu na drugo. Time je omogućeno korisniku uvid kolika je važnost te riječi u danom tekstu.

Tablični prikaz lema, pojavnica i n-grama, sortiranja rezultata te mogućnost pretraživanja u tekstu je indentičan prikazu deskriptora. Odabirom određene leme, pojavnice ili n-grama dobiva se uvid u važnost odabrane riječi u tekstu.



Slika 7.5. *Povezivanje prikaza rezultat deskriptora u tablici sa prikazom deskriptora u Eurovoc prozoru.*

Povezanost prikazanih deskriptora i Eurovoc pojmovnika je ostvarena dvostrukim klikom na naziv deskriptora u tablici rezultata. Otvara se grana pojmovnika u kojoj se nalazi odabrani deskriptor. (uvjet je da je otvoren prozor Eurovoc pojmovnika). Osim pozicioniranja u pojmovniku tj. fokusiranje prikaza stabla na mjesto ozačenog deskriptora, deskriptor u pojmovniku će se označiti žutom bojom.

7.3. Podržani formati datoteka u sustavu

TMT biblioteka podržava XML format datoteka za indeksiranje. No kako je XML format rijetko korišten za pisanje običnih tekstualnih dokumenata i zbog namjene web aplikacije širem krugu korisnika, postojala je potreba da WEB eCadis aplikacija podržava još dva, češće korištena, formata za zapis tekstualnih dokumenata, a to su .html i .doc.

Podržani formati dokumenata koje je moguće indeksirati pomoću WEB eCadis aplikacije:

1. XML
2. HTML
3. DOC

Ako se WEB eCADIS aplikaciji na ulaz postavi dokument drugačijeg formata zapisa od gore navedenih, sustav obavještava korisnika da n epodržava taj format zapisa [28].

8. Srodni radovi

Razmatranja i početci pristupa problemu automatskog indeksiranja dokumenata javili su se 1957. godine kada Luhon [12] u svom radu predlaže statistički pristup u rješavanju problema, počevši od statističke analize skupa dokumenata, formiranja rječnika i izvlačenja relavantnih informacija iz teksta.

Maron 1961. godine izdaje rad u kojem opisuje automatsko klasificiranje dokumenata na temelju naslova dokumenata [13].

Plaunt i Norgard **Error! Reference source not found.**, sa Kalifornijskog sveučilišta u Berkeleyu, su razvili algoritam za automatsko indeksiranje koji pomoću statistike sličnosti kreira relacije između leksičkih termina, koji opisuju dokument (naslov, autori, sažetak), i naslova kategorija u rječniku. Rječnik naslova nije slobodno formiran već se konstruira iz naslova dokumenata u skupu dokumenata za učenje.

Burnside i ostali dio tima [28] su predstavili automatski sustav razvijen u svrhu indeksiranja doktorskih citata rezultata mamografija. Sustav za indeksiranje koristi klasični BI-RADS rječnik termina a za metodu indeksiranja je uzeta metoda minimiziranja razlike kvadrata.

Ripplinger i Schmidt **Error! Reference source not found.** su predstavili AUTINDEX, sustav za automatsko indeksiranje i obradu prirodnog jezika koji je u potpunosti razvijen za njemački i engleski jezik. Sustav koristi niz sofisticiranih metoda obrade prirodnog jezika da bi dokumentu pridružio skup ključnih riječi. Ovakav sustav je moguće poboljšati korištenjem posebnog skupa jezičnih pravila kao što je pojmovnik ili klasifikacijska shema kojima se nalaze informacije kao što su podređeni i nadređeni pojam, sinonim i slično.

Projekt CONDORCET [31] razvijan na Sveučilištu u Twenteu je imao za cilj razvoj i implementaciju sustava za povrat informacija, dio kojeg je i poluautomatski sustav za indeksiranje dokumenata. Sustav analizira naslove i sažetke dokumenata pomoću koncepata i relacija definiranim u ontološkom pojmovniku.

Montejo Ráez u svom radu iz 2002. godine [30] predstavlja program čiji je cilj olakšavanje rada ljudskih indeksatora. Sustav je razvijen kako bi pomogao u radu prilikom indeksiranja tematski uskog skupa dokumenata vezanih uz energetsku fiziku. Sustav koristi statistički pristup problemu i korisniku predlaže skupove iz poznatog DESY pojmovnika.

Prvi rad koji za klasifikaciju dokumenata koristi metode strojnog učenja je rad Ferbera iz 1997. godine u kojem opisuje izgradnju aplikacije koja koristi OECD višejezični pojmovnik. OECD pojmovnik je sličan Eurovoc pojmovniku zbog višejezičnosti. OECD podržava samo četiri jezika i nije toliko opsežan kao Eurovoc pojmovnik [14].

Korištenje Eurovoc pojmovnika u indeksaciji dokumenata nalazimo u radu Pouliquena i Steinbergera (1997. - 2001. godine) [15]. U svom radu prezentiraju sustav koji klasificira dokumente napisane na jezicima koje podržava Eurovoc pojmovnik. Dio sustava koji taži pojmove unutar dokumenta koji se klasificira, koristi zbirku ručno indeksiranih dokumenata [16]. Na temelju njih, za svaki deskriptor unutar Eurovoc-a , daje popis pripadnih pojmova u dokumentu čije pojavljivanje u tekstu se veže za određeni deskriptor iz pojmovnika s određenom težinom. Na temelju dobivenog popisa pojmova i pripadnih težina te frekvencije pojavljivanja pojma u dokumentu određuju se pojmovi koji će biti predloženi za indeksiranje dokumenata [17].

Potaknuti dobrim rezultatima, isti tim stručnjaka 2003. godine nastavlja rad koji rezultira potpunim višejezičnim automatskim sustavom za pridjeljivanje deskriptora koji koristi Eurovoc pojmovnik.

Lathinen [20] predlaže u svom radu metodu kreiranja automatskog indeksatora koji proučava sadržaj dokumenta i gradi indeks kombinirajući podatke o frekvenciji riječi i podatke dobivene korištenjem analizatora rečenične strukture. Mnogi komercijalni sustavi poput Indexicona, CIDEX-a i MACREX-a također obavljaju sličnu funkciju.

CISMeF(fra Catalogue et Index des Sites Médicaux Francophones) je sustav za indeksiranje zdravstvenih dokumenata na francuskom jeziku kako bi se omogućila pretraga na internetu zdravstvenim stručnjacima i pacijentima, kao i svim ljudima koje to područje zanima. Područje koje CISMeF pokriva je znanstvena medicina te briga i skrb o pacijentima[21].

Projekt je započeo 1995. godine sa kreiranjem internet stranice RUH-a (*Rouen University Hospital*) koji je i bio iniciator ovog projekta i trajao do 1999. godine. Da bi se omogućila kompatibilnost i razmjena podataka sa ostalim Internet servisima, sustav koristi prihvaćene internet standarde za razmjenu podataka. Za organizaciju informacija, CISMeF koristi kontrolirani pojmovnik MeSH(*eng.* Medical Subject Heading) preuzet od US Nacionalne Knjižnice za medicinu i set meta-podataka baziranog na nekoliko setova podataka uključujući Dublin Core za opis i indeksaciju svih medicinskih dokumenata uključenih u CISMeF, nekim elementima IEEE 1484 i HIDDEL skupu meta-podataka. CISMeF sustav obuhvaća 4 sljedna procesa: prikupljanje informacija, filtriranje, generiranje kratkog opisa dokumenta i indeksiranje. Prikupljanje dokumenata se odvija dnevno, a dohvaćaju se sa stranica koje imaju kvalitetne dokumente sa područja medicine, praktične savjete, edukacijske dokumente i dr.

Od 1982. do 1993. u NASA-i razvijan je sustav za strojno potpomognuto indeksiranje MAI (Machine Aided Indexing System) [22]. Bitna osobina ovog sustava jest što se za indeksiranje dokumenta ne koriste metode strojnog učenja i statističke obrade teksta za predlaganje deskriptora [2]. Za indeksiranje, sustav koristi pojmovnik te bazu koja sadrži parove ključeva i pojmova iz NASA-inog pojmovnika i odvija se proces uspoređivanja niza znakova iz dokumenta sa nizom znakova iz pojmovnika.

NASA-in pojmovnik pokriva široko područje znanosti, inženjerstva i ostalih specijaliziranih tehnologija te sadrži oko 17 800 pojmova. Najbolje se klasificiraju dokumenti iz ovih područja, ali i za dokumente iz drugih područja, MAI daje jako dobre rezultate.

Web aplikacija je napravljena kasnije da ne samo stručnjacima već i običnim korisnicima omogući korištenje sustava koji određuje sadržaj dokumenta i indetificira ključne riječi i pojmove unutar njega. Aplikaciji se može predati bilo kakav tekst kao ulaz u sustav, npr. Kratke sadržaje, čitave dokumente pa čak i web stranice.

Dokument predan na indeksaciju sustavu, obrađuje se po skupovima riječi. Uzorci nizova riječi unutar dokumentu se uspoređuju sa zapisima u bazi. U slučaju preklapanja sa nekim zapisom u bazi, korisnik dobiva popis deskriptora koji odgovaraju tom zapisu. Proces indeksiranja teksta traje samo nekoliko sekundi, a lista dobivenih deskriptora je sortitrana po frekvenciji i dana su na procjenu korisniku.

Nalazimo sličnost WEB eCADIS sustava sa gore navedenim sustavima, ali najsličniji je MAI sustavu iz više razloga. Oba sustava rade preko web sučelja i na taj način su dostupni većem broju korisnika. Podržavaju automatsko pronalaženje deskriptora unutar teksta i pretraživanje ponuđenih deskriptora unutar pojmovnika. Sustav WEB eCADIS osim toga, podržava i višejezičnost, što mu daje dodatnu prednost nad gore navedenim sustavima i njegov pojmovnik nije uže ograničen na neka područja djelovanja. Također, za bolji uvid u indeksaciju dokumenta, nudi korisniku popis nađenih pojavnica, n-grama i deskriptora unutar teksta. Osim popisa pojmova nađenih u tekstu, moguće je jednostavno pretraživanje tih pojmova unutar teksta i Eurovoc pojmovnika.

9. Zaključak

U ovom radu opisan je sustav za automatsko indeksiranje dokumenata WEB eCADIS temeljen na funkcionalnostima implementiranim u TMT biblioteci. Ideja za razvoj web aplikacije se javila jer su rezultati koje sustav za automatsko indeksiranje dokumenata eCADIS daje više nego zadovoljavajući te da se omogući široj publici brže i uniformnije indeksiranje dokumenata i uvid u funkcionalnosti koje on pruža.

Izrada ovakvog sustava je zahtjevala detaljnu analizu tehnologija potrebnih za povezivanje C++ tehnologije i web tehnologija. Također je trebalo razmotriti koju web tehnologiju izabrati za implementaciju funkcionalnosti koje TMT biblioteka nudi, imajući na umu da WEB eCADIS sustav bi trebao biti prenosiv na različite platforme.

Nakon analize utvrđeno je da najbolja web tehnologija za razvoj WEB eCADIS sustava Java tehnologija najviše iz razloga što je Mono platforma još u razvoju i što je Java tehnologija mnogo razvijenija i stabilnija od skriptnih jezika.

Pri izradi WEB eCADIS sustava bilo je potrebno prilagoditi se zahtjevima web aplikacije tj. funkcionalnostima koje je sustav trebao imati. Također je trebalo obratiti pozornost na zahtjeve i pravila korištenja tehnologija za premošćivanje C++ programskog jezika i Jave.

Nakon analize tehnologija za razvoj WEB eCADIS sustava slijedio je proces implementacije funkcionalnosti TMT biblioteke za rad s Eurovoc pojmovnikom, indeksiranje dokumenata kao i dohvat rezultata indeksacije.

Web aplikacija je napravljena i pojednostavljena tako da se omogući korisnicima lakše korištenje njenih funkcionalnosti.

10. Literatura

[1] Kolar M, Vukmirović I, Dalbelo Bašić B, Šnajder J. Computer Aided Document Indexing System, Journal of Computing and Information Technology CIT, Vol. 13, No. 4, December 2005.

[2] Kolar M, Šarić F, Vukmirović I. Strojno potpomognuto indeksiranje dokumenata, rad za Rektorovu nagradu, Zagreb 2006, dostupno na Internet adresi: http://flambard.zemris.fer.hr/rektor/.

[3] Extensible Markup Language (XML) Version 1.0 Specification. World Wide Web Consortium, 2004.

[4] Thesaurus Eurovoc - Volume 2: Subject-Oriented Version. Ed. 3. Annex to the index of the Official Journal of the EC. Luxembourg, Office for Official Publications of the European Communities, dostupno na Internet adresi: http://europa.eu/eurovoc.

[5] Bratanić M, ed. Pojmovnik Eurovoc, 2nd ed., HIDRA, Zagreb, 2000.

[6] Službene stranice text mining grupe, dostupno na Internet adresi: http://textmining.zemris.fer.hr/tmtdoc

[7] David M. Beazley, An Easy to Use Tool for Integrating Scripting Languages with C and C++,Presented at the 4th Annual Tcl/Tk Workshop, Monterey, CA. July 1996.

[8] Upute za korištenje SWIG alata, dostupno na Internet adresi: www.swig.org

[9] Enterprise Java Computing, Govind K. Seshadri, Cambrige University Press, 1999

[10] Upute za korištenje CMAKE alata, dostupno na Internet adresi: http://www.cmake.org

[11] Šilić A, Šarić F, Dalbelo Bašić B, Šnajder J, TMT: Object-oriented Text Classification Library

[12] Luhn HP. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. In: IBM Journal of Research and Development, 1(4), 1957.

[13] Maron ME. Automatic Indexing : An Experimental Inquiry. In: Jorunal of the ACM, Vol. 8, No. 3, 1961.

[14] Ferber R. Automated Indexing with Thesaurus Descriptors: A Co-occurrence based Approach to Multilingual Retrieval. In: Research and Advanced Technology for Digital Libraires. Proceedings of European Conference of Digital Libraries 1997.

[15] Pouliquen B, Steinberger R, Ignat C. Automatic Annotation of Multilingual Text Collections with a Conceptual Thesaurus. In: Proceedings of the Workshop Ontologies and Information Extraction Summer School The Semantic Web and Language Technology – Its Potential and Practicalities. Bucharest, Romania, 28 July – 8 August 2003.

[16] Steinberger R. Cross-lingual Keyword Assignment. In: Proceedings of the XVII Conference of the Spanish Society for Natural Language Processing, Jaen, Spain, 12-14 September 2001.

[17] Steinberger R, Pouliquen B. Cross-lingual Indexing. Final Report for the IPSC Exploratory Research Project. JRC Internal Note, October 2003.

[18] Plaunt C, Norgard B. An Association-Based Method for Automatic Indexing with a Controlled Vocabulary. In: Journal of the American Society for Information Science 49(10), 1998

[19] Ripplinger B, Schmidt P. Automatic Multilingual Indexing and Natural Language Processing. In: Proceedings of SIGIR, Athens, 2000.

[20] Lahtinen T. Automatic indexing: an approach using an index term corpus and combining linguistic and statistical methods. Academic Dissertation, University of Helsinki, Faculty of Arts, December 2000.

[21] Službene stranice Catalog and Index of French-language Health Internet resources, dostupno na Internet adresi: http://www.cismef.org/

[22] Silvester JP, Genuardi MT, Klingbiel PH, Machine-aided indexing at NASA, Information Processing & Management, 1994, vol. 30, no 5, p. 631-645.

[23] CARNet - Časopis Edupoint godište II | broj 3 | Zagreb | 20.2.2002.

[24] Dario Sušanj: Java, Znak, Zagreb 1997.

[25] Upute za korištenje PHP jezika, dostupno na Internet adresi: http://www.php.net

[26] .NET tehnologija, dostupno na Internet adresi:http://hr.wikipedia.org/wiki/.NET

[27] Malenica M. Primjena jezgrenih metoda u kategorizaciji teksta. Diplomski rad, Zagreb, rujan 2004.

[28] Vidović I. Web servis za automatsko indeksiranje dokumenata. Diplomski rad, Zagreb, rujan 2007.

[29] BURNSIDE BE, RUBIN DL, STRASBERG HR. *Automated Indexing of Mammography Reports Using Linear Least Squares Fit*. Objavljeno u: Proceedings of the 14th International Conference on Computer Assisted Radiology and Surgery, 2000.

[30] MONTEJO RÁEZ A. *Toward conceptual indexing using automatic assignment of descriptors*. Objavljeno u: Proceedings of the ACM 2002 Workshop on Personalization Techniques in Electronig Publishing, 2002.

[31] VAN BAKEL B, BOON R, MARS N, OLTMANS E. *Condorcet Final Report*. Vossius Laboratory, Department of Computer Science, University of Twente, Enschede, Netherlands, Technical Report CTIT TR-00-02.

[32] Manning C, Schutze H. Foundations of statistical natural language processing, The MIT Press 2003.

[33] Pfaffenberg B., Schafter S., White C., Karow B. HTML, XHTML, and CSS Bible, Indianapolis, 2004.

[34] Goodman D., Morrison M., JavaScript Bible, Indianapolis, 2004.