ZAKLJUČNO POROČILO
O REZULTATIH OPRAVLJENEGA RAZISKOVALNEGA DELA NA PROJEKTU V OKVIRU CILJNEGA RAZISKOVALNEGA PROGRAMA (CRP) »KONKURENČNOST SLOVENIJE 2006 – 2013«

I. Predstavitev osnovnih podatkov raziskovalnega projekta

1. Naziv težišča v okviru CRP:

Konkurenčno gospodarstvo in hitrejša rast

2. Šifra projekta:

V2-0215

3. Naslov projekta:

MetaStoritev - Semantično sklapljanje Grid storitev

3. Naslov projekta

3.1. Naslov projekta v slovenskem jeziku:

MetaStoritev - Semantično sklapljanje Grid storitev

3.2. Naslov projekta v angleškem jeziku:

MetaService - Semantic Coupling of Grid Services

4. Ključne besede projekta

4.1. Ključne besede projekta v slovenskem jeziku:

storitve grid, sklapljanje storitev, semantični opis, razvrščanje, usmerjanje, strojno učenje

4.2. Ključne besede projekta v angleškem jeziku:

grid services, service coupling, semantic description, scheduling, routing, machine learning

5. Naziv nosilne raziskovalne organizacije:

XLAB Razvoj programske opreme in svetovanje d.o.o.

5.1. Seznam sodelujočih raziskovalnih organizacij (RO):

Institut Jožef Stefan
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko

6. Sofinancer/sofinancerji:

/

7. Šifra ter ime in priimek vodje projekta:

21555 Marjan Šterk

Datum: 2.10.2008

Podpis vodje projekta:

dr. Marjan Šterk

Podpis in žig izvajalca:

XLAB d.o.o., mag. Gregor Pipan

XLAB
XLAB d.o.o. / SI-1000 Ljubljana

II. Vsebinska struktura zaključnega poročila o rezultatih raziskovalnega projekta v okviru CRP

1. Cilji projekta:

1.1. Ali so bili cilji projekta doseženi?

- [x] a) v celoti
- [ ] b) delno
- [ ] c) ne

Če b) in c), je potrebna utemeljitev.

1.2. Ali so se cilji projekta med raziskavo spremenili?

- [ ] a) da
- [x] b) ne

Če so se, je potrebna utemeljitev:

2. Vsebinsko poročilo o realizaciji predloženega programa dela[1]:

Sistem za sklapljanje storitev prek semantičnih opisov

Uvod

Kot smo zapisali med cilji projekta, je eden od **pogojev za širšo uporabo storitev Grid** poenostavitev sklapljanja posameznih storitev ne le s sintaktične plati, ki je že do neke mere pokrita s prizadevanji za zagotavljanje interoperabilnosti, marveč tudi s semantične plati. Cilj je bil zato razvoj novega sestavnega dela posrednika Grid - **metastoritve**, ki bi nudila nove, vsebinsko bogatejše storitve s **samodejnim sestavljanjem posamičnih preprostih storitev** na podlagi njihovega **razširjenega semantičnega opisa** in bi temeljila na načelu toka podatkov (angl. dataflow).

Storitve Grid so pravzaprav spletne storitve z dodatnimi opisi, ki se nanašajo predvsem na zahteve po virih in rezervacijah le-teh. Razvita rešitev je dovolj splošna, da deluje nad poljubno tehnologijo in ni omejena samo na storitve Grid. Splošnost rešitve pogojuje tudi zastavljen cilj: " *Projekt je aplikacijsko nevtralen, kar ima za posledico, da bodo lahko rezultate te raziskave izkoriščale vse aplikacije, ki so izvedene kot storitve, ter bodo imele izdelane vmesnike za meta-storitev. Nadalje projekt omogoča izdelavo kompleksnih potekov dela (ang. Workflow), kar je predvsem* **namenjeno podjetjem."**

Glavni cilji projekta so bili:

- prispevati k pristopom za semantični opis storitev,
- obogatiti nabor dostopnih storitev,
- poenostaviti in pospešil razvoj novih storitev in aplikacij.

Semantični opis storitev smo zasnovali folksonomijah, ki so znane predvsem iz principov ter algoritmov, ki delujejo pri socialnih omrežjih (Grobelnik et al, 2007). Le-ta so v zadnjem času pridobila na veljavi in razširjenosti, uporabnikom ter razvijalcem pa so zanimiva predvsem zaradi enostavnosti uporabe. Priljubljenost folksonomij je predvsem posledica dejstva, da semantika sloni na principu vreče besed (angl. bag of words), le-te pa so definirane s strani uporabnika, ne pa prek višje avtoritete, organizacije ali protokola (Hotho et al, 2006).

Metastoritve služijo kot vmesni sloj med uporabnikom ter sistemi na nižjem nivoju, saj omogočajo dostopanje do posameznih komponent tega sistema ne preko naslova temveč preko semantičnega opisa komponente. Poleg tega se semantični opisi komponent prenesejo tudi na rezultate. Možno je poizvedovanje oziroma iskanje prek semantičnih opisov, kar velja tako za storitve kot za rezultate.

Nabor storitev smo povečali predvsem na dveh področjih. Prvo je lokalizacija

[1] Potrebno je napisati vsebinsko raziskovalno poročilo, kjer mora biti na kratko predstavljen program dela z raziskovalno hipotezo in metodološko-teoretičen opis raziskovanja pri njenem preverjanju ali zavračanju vključno s pridobljenimi rezultati projekta.

spletnih podatkov, torej določanje koordinat za podatke, vezane na lokacijo, na primer portal z nepremičninami, novicami, zdravstvene ustanove, naslove podjetnikov iz rumenih strani, itd, ki jih nato prikažemo na zemljevidu. Drugo pomembno področje pa je pridobivanje semantične vsebine iz neoznačenih spletnih strani prek definicije vzorcev (angl. patterns).

Poleg dodane semantike smo razvili tudi preprost izvajalni skriptni jezik, ki razvijalcem omogoča enostavnejše izvajanje ter povezovanje storitev. Razvijalcu je tako olajšano delo s samimi storitvami in se lahko osredotoči kako uporabiti rezultate skripte oziroma sklopa storitev.

Izkaže se, da je popolno samodejno sklapljanje storitev prek semantike NP poln problem in je ekvivalentno problemu avtomatskega generiranja programov (Gu et al, 2004; Hu et al, 2007; Klusch et al, 2006). Za razvoj takšnih sistemov je EU namenila ogromna sredstva, razvoj pa je še vedno v teku. Da bi se izognili tovrstnim težavam, smo se odločili, da uporabimo princip cevovodenja rezultatov izvedenih storitev, znan predvsem iz sveta, ki temelji na operacijskih sistemih UNIX. Uporabnik določi zaporedje izvajanja storitev glede na opis, sistem sam poišče iskano storitev ter jo izvede z želenimi parametri, rezultate pa posreduje naslednjemu ukazu. Tako postane izvajanje storitev polavtomatsko.

Ko prek posebnega ukaza in semantičnega opisa storitev izvedemo, lahko njene rezultate shranimo in posredujemo naslednjim metastoritvam. Ta princip nam omogoča skriptno izvajanje storitev v sistemu. V ta namen smo razvili preprost skriptni jezik, ki zaporedno izvaja ukaze, namenjene izbiri storitev, izvajanju, shranjevanju parametrov, ostrenju rezultatov, itd. Podrobnejši seznam in opis ukazov je naveden v nadaljevanju. Oglejmo si preprost primer skripte:

```
execute delo borza delnice trenutno stanje
select delnica delo borza
store delnice
saveresults v@delnice borza.results append
```

Skripta kliče po izvedbi storitve, katere semantični opis vsebuje ključne besede "delo, borza, delnice, trenutno stanje" (vrstica 1). Gre za storitev, ki iz Delove spletne strani pobere trenutno stanje delnic. Izmed pridobljenih rezultatov nas zanimajo le tisti, katerih semantični opis vsebuje "delnica, delo, borza" (vrstica 2). Izbiro shranimo v spremenljivko "delnice" (vrstica 3). Shranimo rezultate, ki se nahajajo pod naslovom spremenljivke "delnice", v datoteko "borza.results tako, da dodamo vsebino k že obstoječi vsebini (vrstica 4).

Izvajanje storitve ni omejeno na okolje. Klic lahko poteka tako v okolje Grid kot izvaja sistemske ukaze. Še posebej je sistem primeren za semantično označevanje spletnih strani. Razvijalec lahko izlušči vsebino posameznih vzorcev (angl. patterns), torej bloke HTML strani, vsebini teh blokov pa pripiše semantiko. Tako uporabnikom omogočimo dostop do označene vsebine strani, kljub pomanjkanju označitve le-teh v osnovni različici. Takšno izluščenje in označitev

2 http://peersim.sourceforge.net/
3 http://www.cs.waikato.ac.nz/ml/weka/

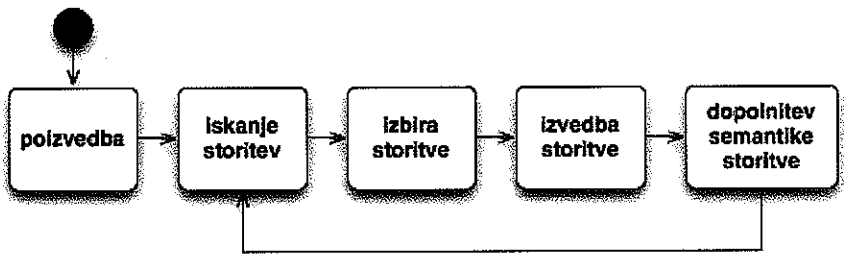
omogoča razvoj mešanja strani (angl. mashups) ter korak bliže k semantičnemu spletu, ne da bi razvijalce spletnih strani silili k podrejanju standardov ter besednjaku za označevanje. Označevanje ter določanje zanimivih kosov informacij tako prepustimo uporabnikom - trend socialnega označevanja ter povezovanja v spletu že nekaj časa strmo narašča.

Zasnova in dopolnjevanje semantičnih opisov

Običajni pristop k semantičnemu opisovanju so ontologije. Zaradi njihove striktnosti in togosti smo se raje odločili za tako imenovane ljudske semantike oziroma folksonomije, alternativo, ki se je razvila v zadnjem času. Gre za opisovanje virov s preprostimi nizi besed ali besednih zvez. Nad njimi ne bdi nobena avtoriteta, ni se potrebno podrejati nekim formalnim predpisom, ki so značilni za ontologije. Opisovanje virov je tako enostavnejše, uporabniku bolj dostopno, s tem pa se tudi poveča verjetnost, da si bo uporabnik ali razvijalec dejansko vzel čas in vir označil (Ilijašić et al, 2007). Prav tako je nad folksonomijami možno izvajati operacije dodajanja, nadgrajevanja ter odvzemanja vsebine, prav tako pa ugotavljanje enakosti, kar pri izvedenih ontologijah pogosto postane problem. Največji razmah so folksonomije dosegle z razširitvijo socialnih omrežij oz. tako imenovanega Web 2.0.

Čeprav smo pri razvoju uporabljali folksonomije, pa je sistem zasnovan tako, da je moč namesto izbranega semantičnega načina uporabiti tudi druge principe, na primer ontologije.

Sistem je zasnovan na principu povratne zanke, prikazane na sliki 1. Storitve so opisane s semantiko, po kateri lahko uporabniki podajajo poizvedbe. Storitve po izvajanju (lahko) pridobijo rezultate. Semantični opisi rezultatov so, kot smo omenili, dedovani s strani storitev, vendar pa se tudi vrednosti teh rezultatov prenašajo pod semantični opis storitev. Poleg tega lahko razvijalec storitve semantični opis rezultatov tudi dodatno razširi. Tako se opis storitve dopolnjuje z vsakim izvajanjem. Princip je koristen predvsem za ostrenje iskanja oziroma rangiranja storitev pri poizvedbi po semantiki, saj se storitve, ki imajo podoben semantični opis (na primer pridobivanje podatkov o borzni kotaciji delnic) izostrijo (delnice Krke, Petrola, projekcija delnice, itd.).



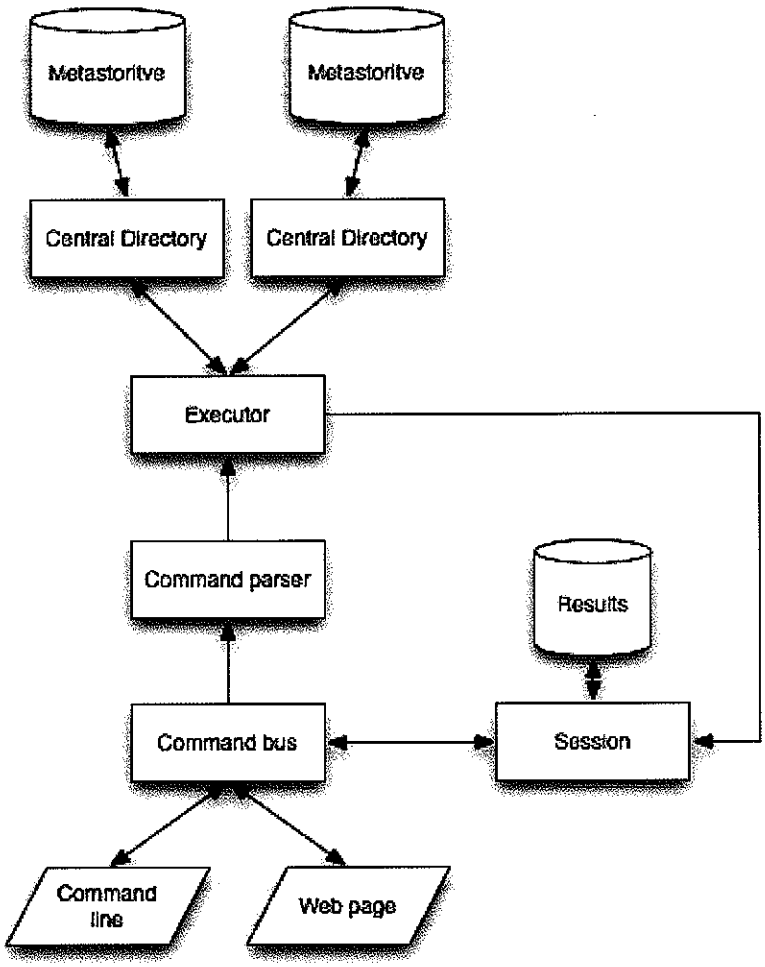
Slika 1: Princip povratne zanke pri dopolnjevanju semantičnega opisa storitve.

Arhitektura sistema

V uvodu smo omenili, da izvajanje storitev poteka po sledečem postopku:

- poizvedba po storitvi glede na semantični opis,

- poizvedba po v seji shranjenih rezultatih glede na semantični opis oz. shranjeno lokacijo,
- izvedba storitve (lahko s parametri ali brez),
- vrnjeni rezultati, dopolnitev semantičnega opisa storitve.



Slika 2: Visokonivojska arhitektura sistema.

S pomočjo slike 2, ki kaže arhitekturo sistema, opišimo potek izvajanja skripte:

1. Uporabnik sproži skriptni ukaz (lahko preko uporabniške vrstice ali pa prek spletne strani), ki se prenese v sistem. Ukaz je vezan na trenutno sejo uporabnika.
2. Ukaz se prenese v razčlenjevalnik ukazov (Command Parser). Ukaz se razbije na poimenovane žetone (angl. token), tako da je moč razlikovati med vrednostmi, ki se nanašajo na semantiko storitve, vrednostmi, ki se nanašajo na semantiko rezultatov, objekte, ki predstavljajo rezultate, itd.
3. Pravilno oblikovan in razčlenjen ukaz se posreduje izvrševalcu (Executor). Ta prek žetona s semantičnim opisom in centralnih imenikov opravi poizvedbo po storitvi in prek izbrane hevristike določi najustreznejšo med njimi. Izbrani storitvi posreduje žetone s parametri in pokliče metodo za njeno izvajanje. Storitev nato pripravi žeton z rezultati in ga vrne izvrševalcu. Pri pridobivanju rezultatov se tudi avtomatsko dopolni

semantični opis storitve.

4. Izvrševalec shrani rezultate v začasno spremenljivko, ki je vezana na sejo. Ti rezultati so na voljo naslednjemu ukazu, lahko se uporabijo kot parametri ali pa se shranijo v spremenljivko.

Podrobnejši opis komponent se nahaja v nadaljevanju.

Dodajanje storitve v sistem

Naloge razvijalca storitve so:

- implementirati metodo, ki poskrbi za izvajanje storitve,
- dodati semantični opis storitve,
- opcijsko lahko dopolni semantiko vrnjenih rezultatov (vezano na potek izvajanja storitve),
- registrirati storitev v enem izmed centralnih imenikov,
- v primeru novega centralnega imenika je potrebno tudi le-tega registrirati pri izvrševalcu.

Semantični opis storitve se doda v imenik kot seznam besed, npr. *{"gnuplot", "generate", "pdf"}*. Za vse ostalo poskrbi ogrodje samo.

Uporabnik dostopa do sistema ali prek ukazne vrstice (s pomočjo skriptnih ukazov) ali pa prek spletne strani. Prvi način je namenjen predvsem razvijalcem spletnih strani, saj zahteva poznavanje ukazov, ki jih ima na voljo skriptni jezik. Končnemu, tehnično nepodkovanemu uporabniku sestavljeno storitev predstavimo na uporabniku prijazen način prek spletne strani.

Semantika

V uvodni predstavitvi smo poudarili razloge za izbiro semantike, ki temelji na označevanju z besednimi zvezami (folksonomije) in ne na neki ontologiji. Semantika je tako predstavljena z vrečo besed (angl. bag of words) oziroma besednih zvez. Iskanje prek semantike je možno preko naslednjih operacij:

- vsebuje semantični opis,
- je podmnožica semantičnega opisa,
- enakost semantičnega opisa.

Operacija vsebovanosti je pomembna pri poizvedbah, kjer poizvedujemo po storitvah, ki jih še ne poznamo. Preko osnovnih zaznamkov pridemo do širše skupine storitev, ki so na voljo z nekim opisom, potem pa z dodajanjem opisa ostrimo to izbiro (na nek način je postopek podoben ostrenju rezultatov pri spletnih iskalnikih).

Operacija preverjanja vsebovanosti podmnožice se uporablja predvsem pri cevovodenju rezultatov z novimi storitvami. Semantika rezultata se preveri s semantiko storitve. V primeru, da je presek semantike rezultata ter storitve neprazna množica besed, je storitev kandidat za uporabo rezultata.

Rezultat

Rezultat je sestavljen iz vrednosti ter semantičnega opisa. Semantični opis se deduje od storitve, ki ga je pridobila, lahko pa je dopolnjen tudi s semantiko metode, ki je rezultat vrnila.

Metastoritev

Metastoritev služi kot ovoj (angl. wrapper) za izvajanje operacij, bodisi sistemskega ukaza ali pa poljubnega programskega bloka. Kot smo že omenili pri opisu sistema, je možno z opisanim ogrodjem označevati spletne strani, torej informacijo, ki ni označena. Predvsem je to enostavno za strani, katerih izvorna koda je sestavljena iz generiranih blokov, npr. strani z novicami. Te vzorce je moč razbrati ter jim dodati semantiko. Razvijalec mora definirati regularni izraz za želene vzorce ter mu določiti semantiko, ogrodje pa poskrbi za označevanje vsebine in pridobivanje rezultatov.

Primer bloka kode, ki poskrbi, da iz spletne strani hiše Delo pridobimo podatke o posamezni delnici:

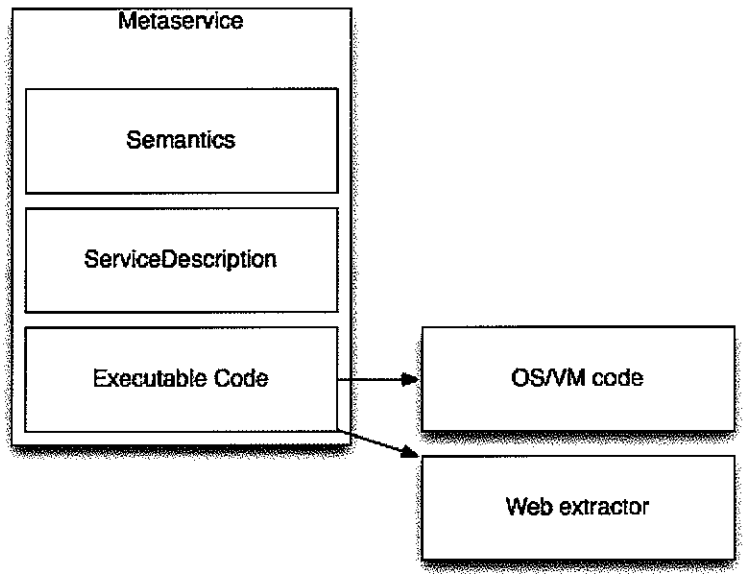
```
Pattern prow = Pattern.compile("(?s)<tr(.*?)</tr>");
Semantics srow = new Semantics();
srow.addKeyword("delnica");
Extractor e = new Extractor(prow, srow);
Result row = e.extract(content);
```

Seveda s tem dobimo samo notranjo obliko vrstice tabele, nič pa ne vemo o posameznih vrednostih. Kompleksnejši primer dodajanja semantike je prikazan z naslednjim blokom kode. Za vsak rezultat, ki ga dobimo prek vzorca iz prejšnjega primera, preverimo ali je prave oblike in če je, nad njim izvedemo novo ekstrakcijo ter označimo posamezne komponente (kratica, ime, sprememba itd). Prek objekta "ContextResult" shranjujemo novo informacijo direktno v že obstoječi rezultat. Tako postane rezultat gnezden (vsebuje druge rezultate), s čimer podeduje vso semantiko vsebovanih rezultatov.

```
Pattern pcol = Pattern.compile("(?s)<td(.*?)</td>");
Semantics scol = new Semantics();
scol.addKeyword("vrstica");
Extractor ecol = new Extractor(pcol, scol);
for(String rowval : row.values) {
      Result cols = ecol.extract(rowval);
      if(cols.values.size() > 4) {
            ContextedResult cr = new ContextedResult(); Semantics s0 =
            new Semantics();
            s0.addKeyword("kratica");
            cr.addContext(0, s0);
            Semantics s1 = new Semantics();
            s1.addKeyword("ime");
            cr.addContext(1, s1);
            Semantics s2 = new Semantics();
            s2.addKeyword("sprememba"); s2.addKeyword("procent");
            cr.addContext(2, s2);
            Semantics s3 = new Semantics();
            s3.addKeyword("vrednost");
            cr.addContext(3, s3);
            Result nr = cr.putInContext(cols);
            nr.addSemantics(srow);
            nr.purifyValues("(?s)>(.*)</");
      }
}
```

Poleg semantičnega opisa ter bloka za izvajanje imajo metastoritve tudi dodaten

semantičen opis. Le-ta je vezan na semantiko ter vrednost rezultatov, ki jih storitev vrača uporabniku. Hranjenje te informacije je nujno za vzpostavljanje povratne zanke ter s tem povezanim ostrenjem iskanja rezultatov, oziroma specializacije storitev. Trenutna izvedba hrani popolno informacijo kot dolgoročni spomin. Ta rešitev zadošča za prototip ogrodja, ki smo ga razvili, v nadaljevanju pa bo potrebno poleg dolgoročnega spomina dodati tudi hevristiko za kratkoročni spomin, ki bo hranil neko količino sprotne informacije, in v dolgoročni spomin prenašal samo informacijo z "dovolj" poizvedbami oz. zahtevami.



Slika 3: Komponente metastoritve.

Centralni imenik ter izvrševalec

Centralni imenik hrani vse storitve, ki se nahajajo na posameznem vozlišču v omrežju. Storitve je moč v imenik dodajati ali odvzemati v realnem času.

Centralne imenike je potrebno registrirati v izvrševalca. Le-ta vedno posreduje poizvedbe vsem centralnim imenikom. Izvrševalec je sestavljen iz več komponent, med drugim izvajalnika ukazov in podatkov o seji. Slednji vsebujejo pridobljene rezultate zadnjega ukaza in pomnilnik za spremenljivke.

Rezultati zadnjega ukaza so na voljo naslednjim operacijam. V primeru, da rezultate shranimo v pomnilnik, je potrebno rezultatom pripisati "naslov", preko katerega jih bo moč dostopati. Skupine ustvarimo s posebnim ukazom, do njih pa pridemo prek podmnožice semantičnega opisa.
Izvajanje ukazov poteka zaporedno in sinhrono. Sinhronost je potrebna zaradi podatkovnih nevarnosti, odprava le-teh in uvedba asinhronega izvajanja pa bi bistveno povečala kompleksnost izvedbe.

Skriptni ukazi

Za izvajanje storitev imamo dva ukaza:

- **execute** "opis"
- **pexecute** "opis" **with** "parametri"

Prvi ukaz izvede storitev glede na njen semantični opis, med tem ko je drugi namenjen za izvajanje storitev, ki potrebujejo tudi parametre. Izvajanje trenutno deluje po principu "za vse", se pravi, izvedejo se vse storitve, ki ustrezajo semantičnemu opisu.

Za izbiranje izmed rezultatov, ki jih vrne izvajanje storitve imamo na voljo naslednje ukaze:

- **select** "opis"
- **group** "spremenljivka" **by** "opis"
- **selectgroup** "vrednost"
- **subset** "opis" **from** "spremenljivka"
- **contains** "vrednost" **from** "opis"

Z ukazom "select" izmed rezultatov, ki so trenutno shranjeni v seji kot zadnje vrnjeni rezultati storitve, izberemo podmnožico takšnih, ki vsebujejo vsaj izbran semantični opis. Z "group" naredimo skupine, ki so deljene po enakih vrednostih izbranih semantičnih opisov. Ukaz "selectgroup" izbere tisto skupino rezultatov, katere skupina vsebuje izbrano "vrednost". Ukaz "subset" iz spremenljivke pobere tiste rezultate, katerih semantični opis je nadmnožica podanega opisa. Ukaz "contains" izbere rezultate, ki vsebujejo tako želeni semantični opis kot tudi želeno vrednost.

Za shranjevanje rezultatov imamo na voljo naslednje ukaze:

- **store** "ime"
- **saveresults** "spremenljivka" "datoteka" "način"
- **loadresults** "datoteka"

Ukaz "store" shrani rezultate, ki so trenutno na vrhu sklada pod spremenljivko "ime". Ukaz "saveresults" pa shrani vse rezultate shranjene pod neko spremenljivko v datoteko. Način je lahko ali ustvarjanje nove datoteke ali pa dodajanje.

Pomožni ukazi:

- **sortbydate**

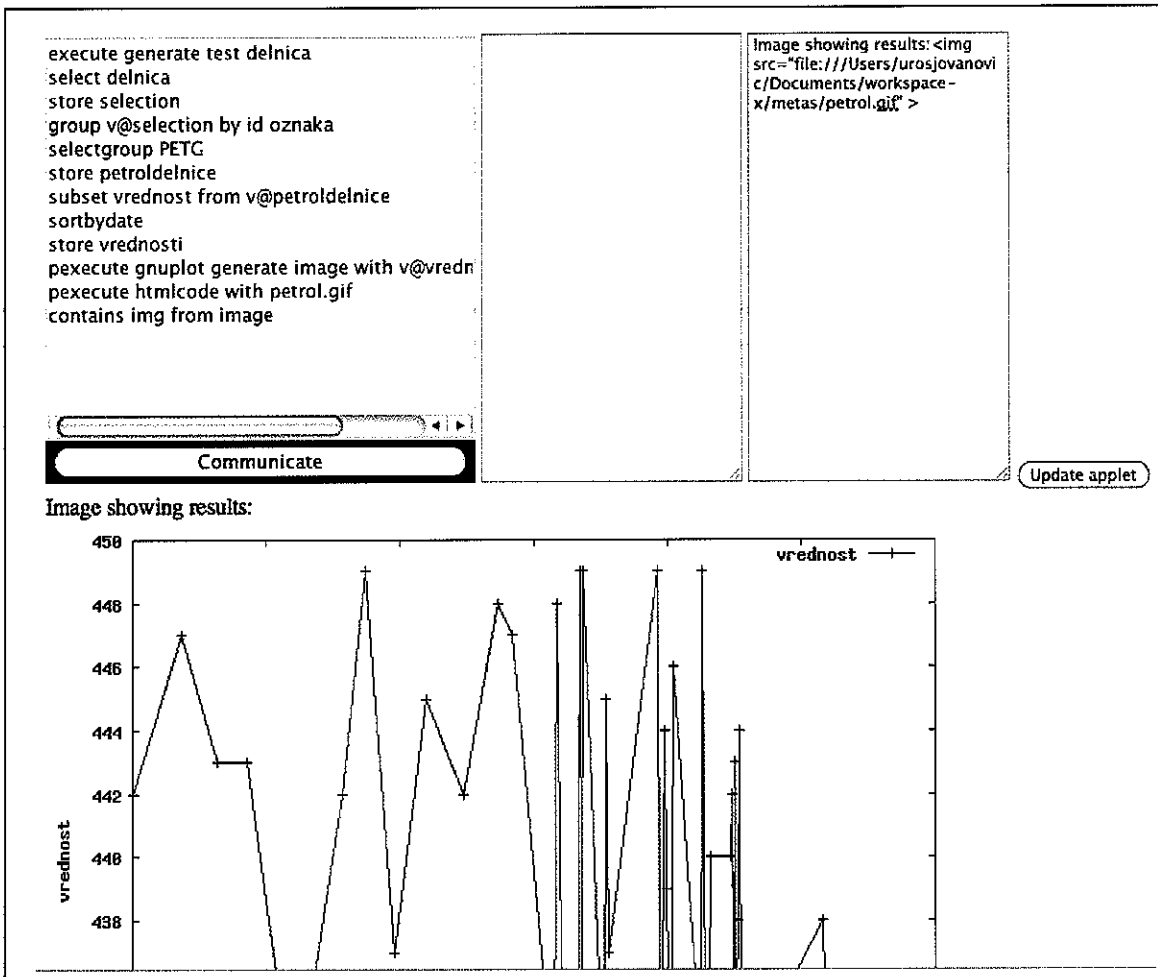
Ukaz "sortbydate" uredi rezultate po času nastanka, od najstarejšega do najnovejšega. Ukaz je koristen v primerih, ko je potrebno rezultate prikazati v pravilnem časovnem zaporedju.

Spletna stran za oddaljeno izvajanje ukazov

Za lažji razvoj in izvajanje skript brez potrebe po nameščanju ogrodja na lastno infrastrukturo smo razvili prototip spletne strani, ki bo omogočala izvajanje skript. Prototipa je prikazan na sliki 4. Vsebuje polje že izvršenih ukazov (zgodovino ukazov), polje za vnos ukazov ter tekstovno polje za prikaz rezultatov. Na dnu strani je HTML element, namenjen sprotnemu osveževanju in prikazu rezultatov. Razvidno je, da zadnja ukaza skripte

```
pexecute htmlcode with petrol.gif
contains img from image
```

najprej ustvarita sliko, nato pa njen naslov vrneta v pravilno oblikovani HTML obliki na spletno stran, ki sliko uspešno prikaže.



Slika 4: Prototip spletne strani za oddaljeno izvajanje ukazov.

Primeri uporabe

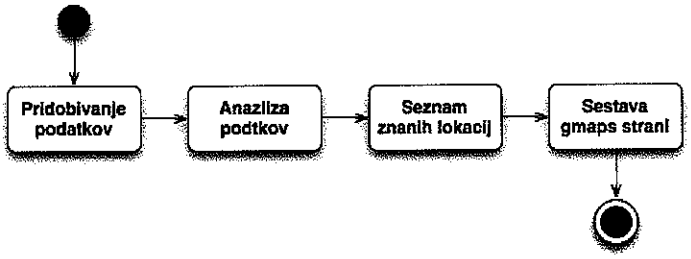
Lokaliziranje podatkov

Prvi primer uporabe se nanaša na lokaliziranje podatkov, pridobljenih na spletu, ter njihov prikaz s na zemljevidu, na primer Google maps. Postopek je prikazan na sliki 5. Najprej zajamemo podatke z izbranih spletnih strani s pomočjo ekstrakcije vzorcev, kot smo opisali v predhodnih razdelkih. Rezultate nato podamo storitvi, ki jih analizira. Pri analizi gre predvsem za razpoznavo krajevnih imen: mest, vasi, ulic, itd. Razpoznane kraje nato podamo naslednji storitvi, ki pa jim določi lokacijo. Sedaj imamo na voljo troje podatkov:

- vsebino sporočila,
- kraj sporočila,
- lokacijo na zemljevidu.

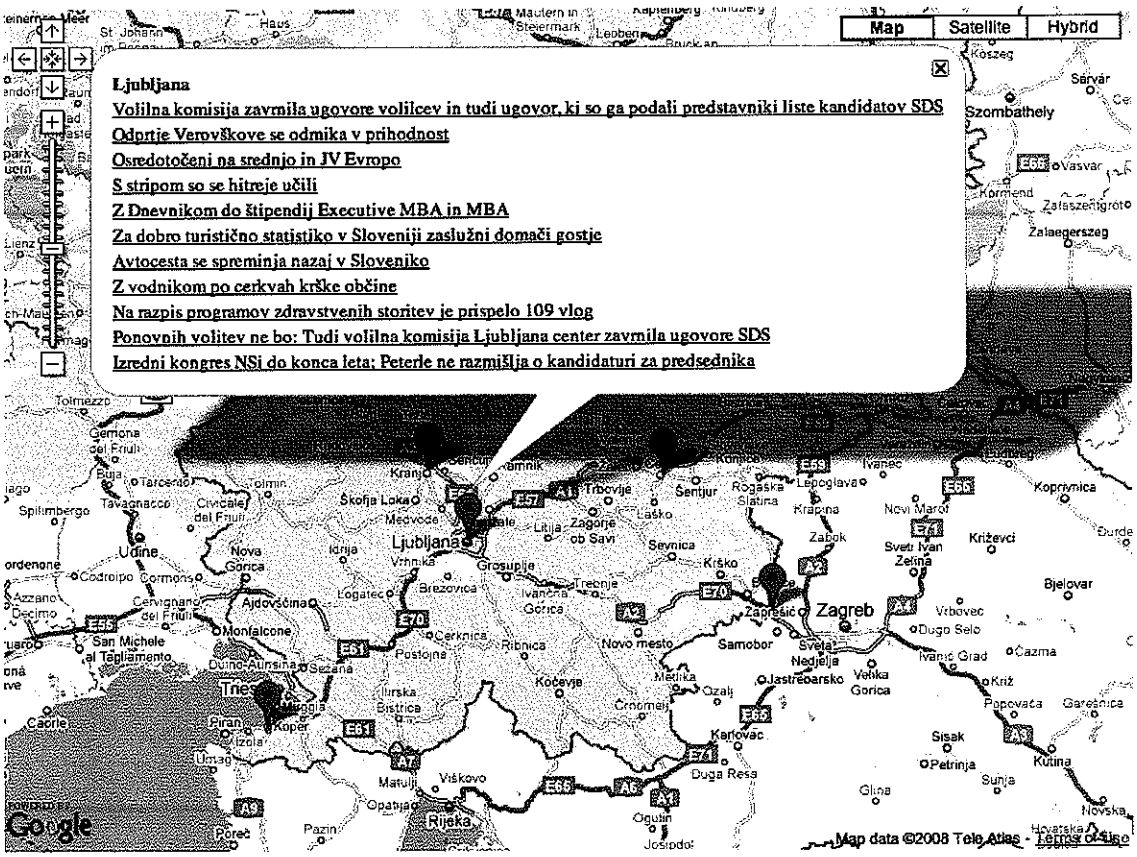
Te trojke posredujemo generatorju lokaliziranih podatkov. Generatorjev je lahko seveda več, odvisni pa so od načina prikaza podatkov. Najbolj splošna oblika izvoza je v formatu XML. V našem primeru smo uporabili izvoz v HTML, prikaz pa poteka s pomočjo spletne aplikacije Google maps. Google maps namreč omogoča, da na želeno lokacijo "pripnemo" zaznamek z vsebino. Na slikah je ta zaznamek razviden kot balon, ki izhaja iz izvorne lokacije. S klikom na zaznamek

se nam odpre njegova vsebina (v belem oblačku), torej spletne povezave na spletne strani, ki hranijo analizirano vsebino.



Slika 5: Postopek označevanja podatkov.

Posebno skrb smo posvetili prikazu večje količine podatkov. V primeru manjše natančnosti prikaza ozemlja (pogled od daleč), zaznamke združimo in jih prikažemo kot en sam zaznamek. V primeru, da uporabnik izbere nek zaznamek (desni gumb), se spremeni natančnost prikaza zemljevida tako, da vsebuje le zaznamke, ki jih je vseboval izbrani združeni zaznamek. V primeru, da je teh novih zaznamkov ponovno preveč za prikaz, so lahko nekateri zaznamki ponovno združeni.
Oglejmo si najprej preprost primer lokaliziranih novic (slika 6). Podatki so pridobljeni s spletne strani medijske hiše Dnevnik (http://www.dnevnik.si), omejili pa smo se le na kraje Ljubljana, Maribor, Celje, Koper, Kranj in Novo mesto. Slika prikazuje seznam novic, ki se nanašajo na ali pa izvirajo iz mesta Ljubljana.



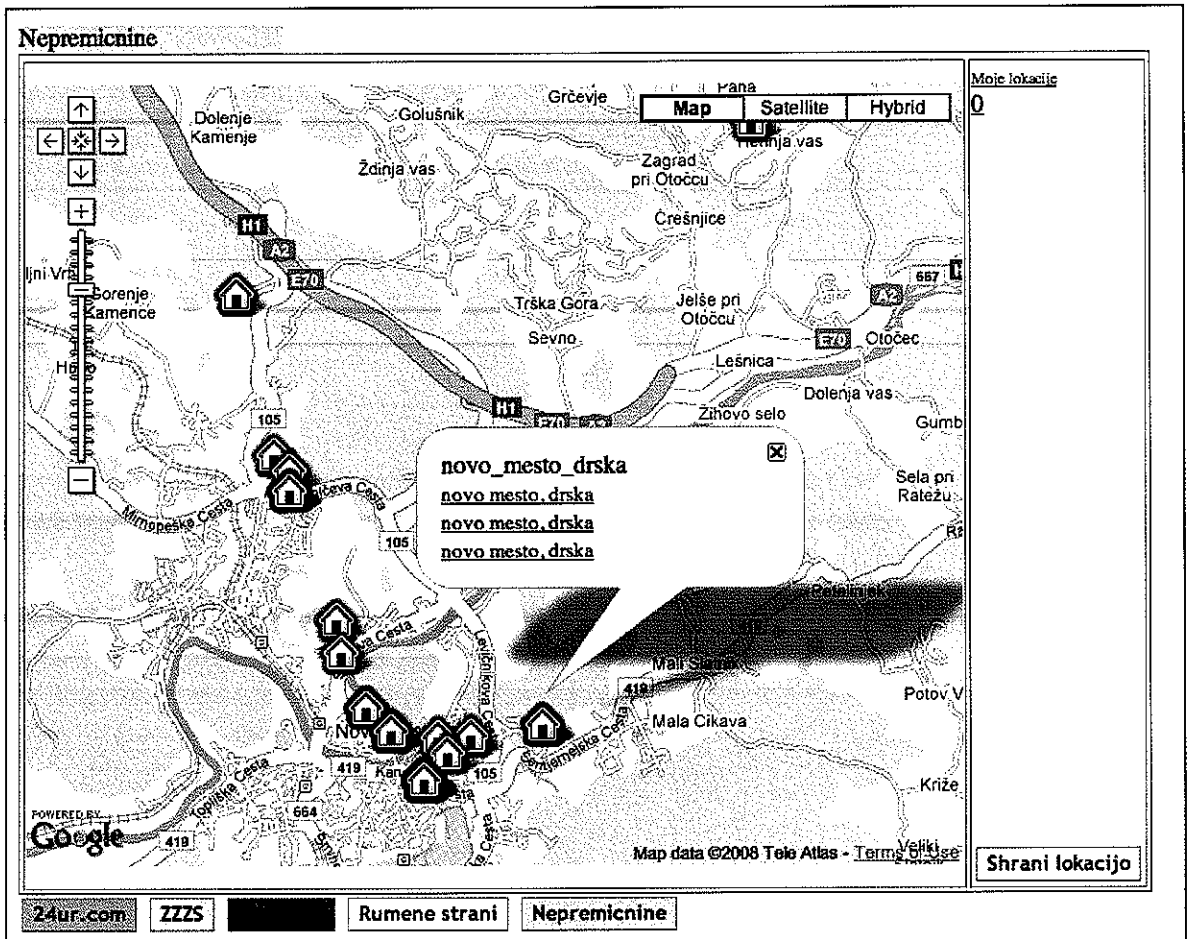
Slika 6: Prikaz novic Dnevnika za Ljubljano.

V naslednjem koraku smo razvili kompleksnejši vmesnik za prikaz lokaliziranih podatkov. Izbrali smo naslednje tematike:

- nepremičnine,
- novice s strani 24ur.com,
- lokacije poslovalnic ZZZS,
- lokacije poslovalnic družbe Mercator,
- rumene strani.

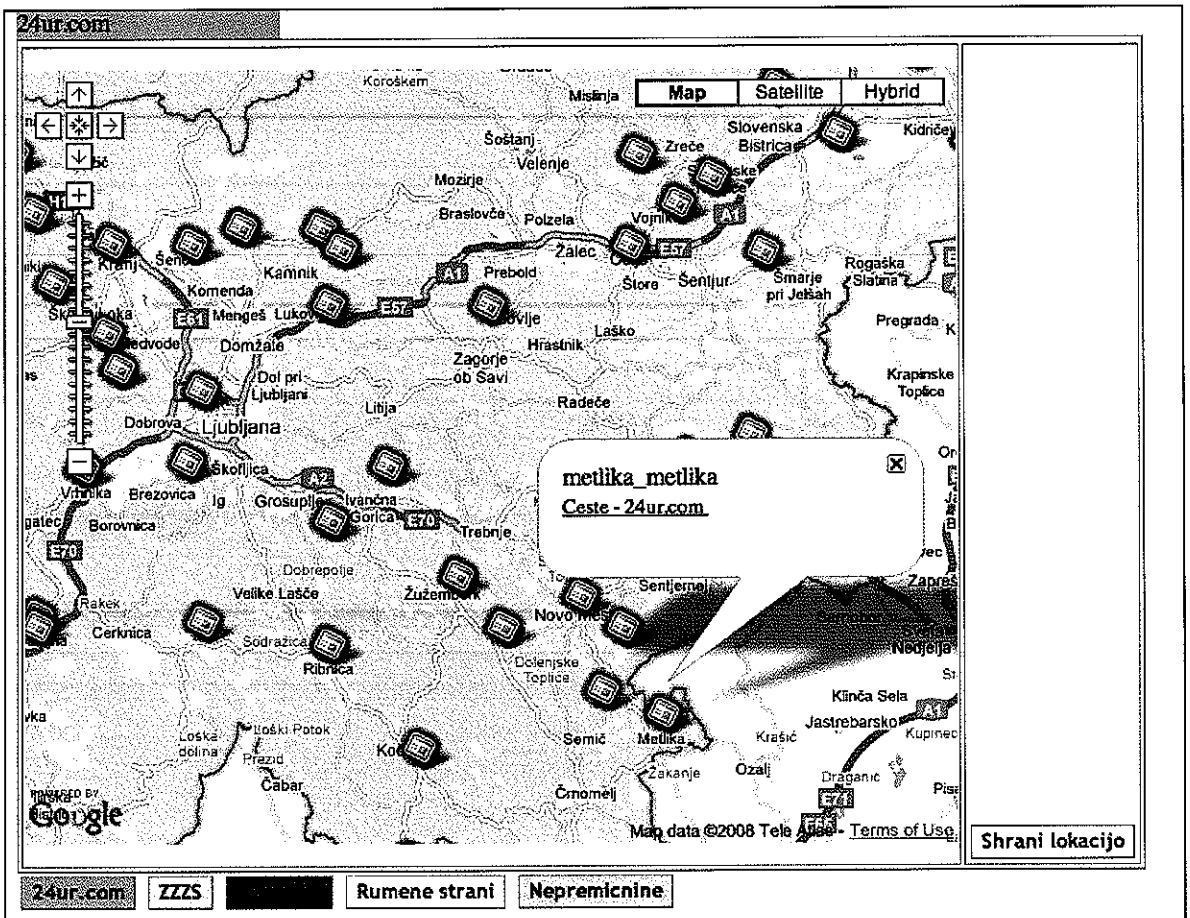
Podatki za nepremičnine so pridobljeni s spletne strani http://nepremicnine.net . ZZZS ima na svoji spletni strani (http://www.zzzs.si) seznam poslovalnic, prav tako Mercator (http://www.mercator.si). Podatki o novicah so pridobljeni s spletne strani 24ur.com (http://24ur.com), bolj natančno iz njihovega RSS kanala za novice iz Slovenije. Rumene strani (http://www.rumenestrani.com) vsebujejo ogromno podatkov o podjetnikih in poslovalnicah širom Slovenije. Poudariti je potrebno, da smo naveden spletne strani uporabili izključno kot primere uporabe sistema. Avtorske pravice za vse podatke, pridobljene s teh strani, so v lasti navedenih pravnih oseb.

Vmesnik je prikazan na slikah 7, 8 in 9. Med posameznimi sklopi podatkov prehajamo z izbiro tematike (obarvani gumbi na dnu strani). Trenutno izbrana tematika je prikazana na vrhu strani. Prav tako se glede na tematiko med sabo zaznamujejo zaznamki: nepremičnine so prikazane s hišo, poslovalnice ZZZS z rdečim križem, poslovalnice Mercatorja z nakupovalnim vozičkom, itd. Večjo količino zaznamkov prikažemo z združenim modrim zaznamkom na način, opisan v primeru novic medijske hiše Dnevnik.

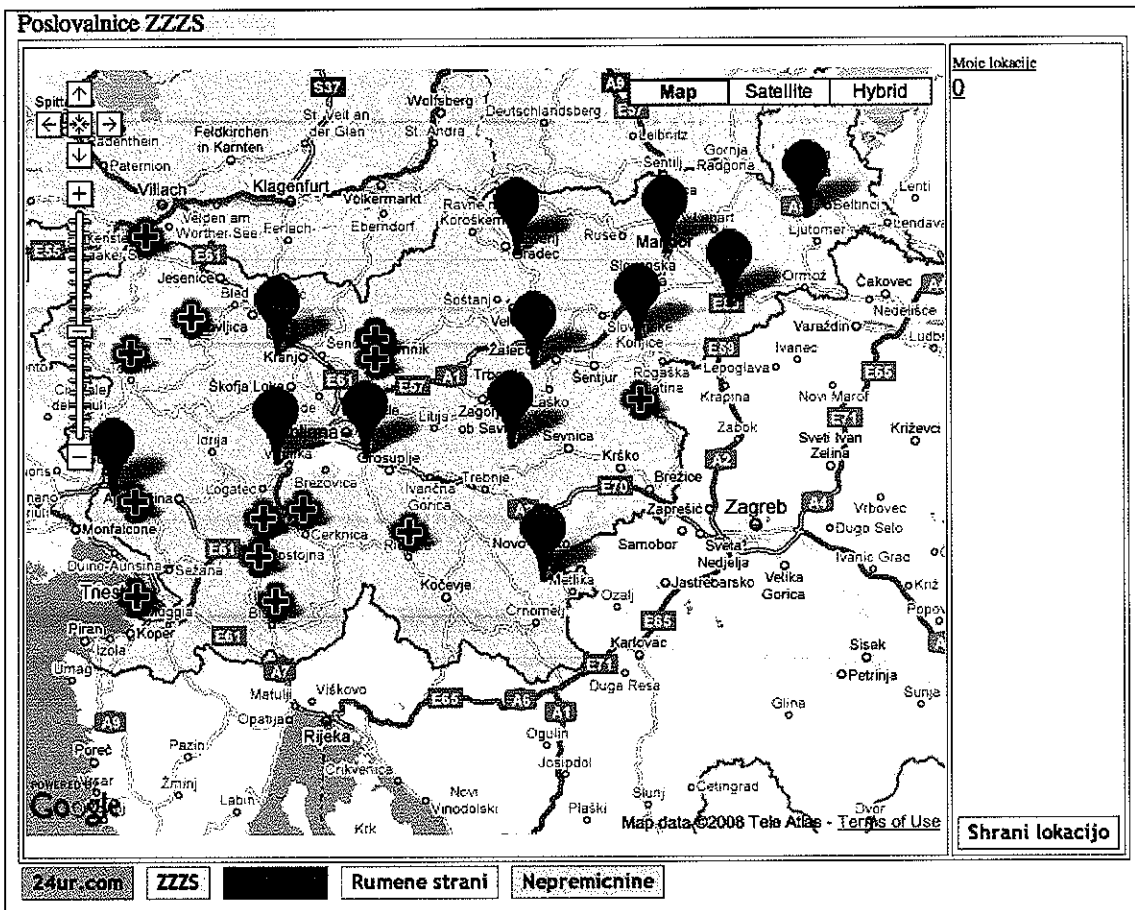


Slika 7: Prikaz lokacij nepremičnin za center Novega mesta.

Na desni strani vmesnika se nahaja poseben seznam, ki služi za zaznamke obiskanih lokacij. Na primer, da uporabnik išče nepremičnino v več regijah, Ljubljani, Kopru ter Novem mestu. Ko ugotovi lokacijo zanimive nepremičnine v Ljubljani, klikne na gumb "Shrani lokacijo". V seznam se shranijo trenutni parametri pogleda. Uporabnik se lahko mirno premakne s pogledom v Koper, shrani zanimive lokacije, ter nadaljuje iskanje v Novem mestu. Nato s klikom na shranjeni zaznamek prehaja med izbranimi lokacijami. Na sliki 7 je trenutni pogled na nepremičnine, ki se prodajajo v novomeškem starem jedru, shranjen pod lokacijo z zaznamkom 0. V primeru menjav vsebin (nepremičnine, ZZZS, itd.) ostanejo zaznamki lokacij enaki. Tako lahko uporabnik, ki se zanima za neko nepremičnino, preveri bližino ustanov ZZZS (bolnice, lekarne ali zdravstvenega doma), bližino nakupovalnega centra oz. prodajalno živil, preko novic 24ur.com pa lahko spremlja kulturno dogajanje v okolici.
Slika 9 prikazuje lokacijo poslovalnic ZZZS. Gre za zdravstvene ustanove, ki so naštete na spletni strani ZZZS. Zaznamki z rdečim križem kažejo na posamezno ustanovo, modri baloni pa združene zaznamke. S klikom na zaznamek z rdečim križem dobi uporabnik povezavo na opis ustanove s spletne strani ZZZS. Ta vsebuje opis ustanove, naslov ter kontaktne informacije.
Slika 8 prikazuje novice v ljubljanski, celjski ter novomeški regiji. V primeru Metlike kaže na novice, ki se nanašajo na tamkajšnje stanje na cestah.



Slika 8: Prikaz novic strani 24ur.com za jugovzhodni del Slovenije.



Slika 9: Poslovalnice ZZZS po Sloveniji.

Sestavljanje storitev s skriptnim jezikom

Prejšnji primer je kazal sestavljeno storitev, predstavljeno uporabniku kot spletno stran. Oglejmo si še bolj zapleten sistem skripte, ki nariše graf gibanja vrednosti delnice družbe Krka:

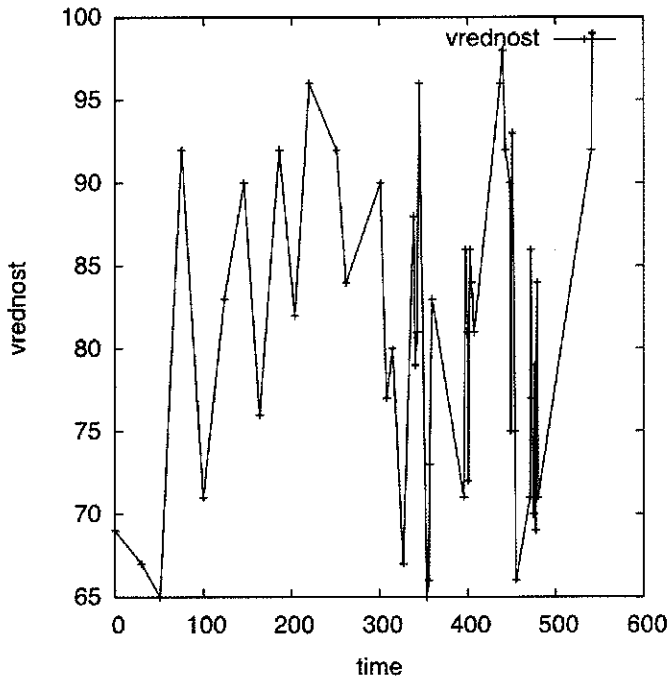
```
execute delo borza delnice trenutno stanje
select delnica
store selection
group v@selection by id oznaka
selectgroup KRKG
store krkadelnice
group v@krkadelnice by origin servername
pexecute echo with v@krkadelnice
subset vrednost from v@krkadelnice
sortbydate
store vrednosti
pexecute echo with v@vrednosti
pexecute gnuplot pdf with v@vrednosti krka
```

Najprej poženemo storitev, ki iz spletne strani Dela (http://www.delo.si, slika 10) prebere ter naloži trenutne vrednosti delnic. Nato izmed rezultatov izberemo samo "delnice", ter shranimo izbiro v spremenljivko "selection". Te rezultate uredimo po oznaki. Izberemo skupino, katere oznaka vsebuje vrednost "KRKG", ter shranimo skupino pod spremenljivko "krkadelnice". Sledita dva ukaza, ki sta namenjena samo preverjanju rezultatov (testni izpis). Prvi uredi rezultate v

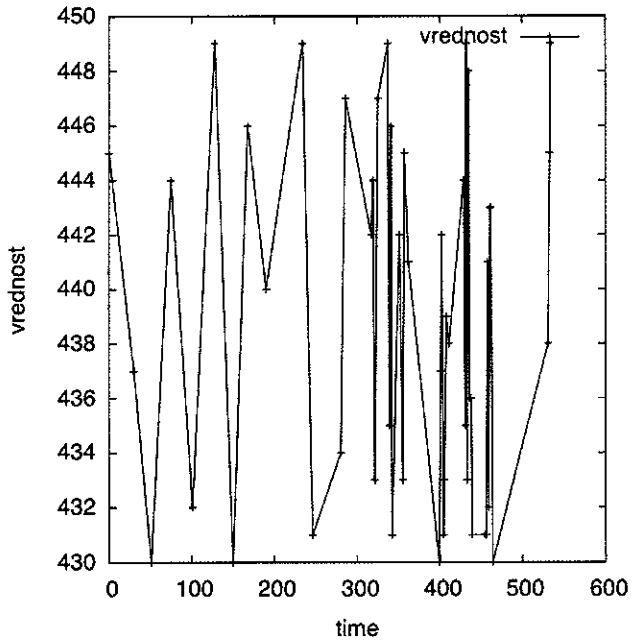
skupine glede na izvor, oz. lokacije storitve, ki je vrnila rezultat. Drugi ukaz pokliče storitev "echo", ki prikaže vsebino. Nadaljujemo s tem, da iz skupine Krkinih delnic izločimo samo podsklope rezultatov, ki vsebujejo opis "vrednost". Te rezultate uredimo po času nastanka (ker so lahko rezultati iz različnih virov premešani). Na koncu pokličemo storitev, ki iz urejenih vrednosti Krkinih delnic naredi graf, rezultat pa shrani v format PDF (sliki 11 in 12). Končni graf je prikazan na sliki 11, podoben graf za delnice Petrola pa na sliki 12.

IFHR	CENTER NALOŽBE	▲ 2.96	7.3	0.21	7.3	7.3	7.3	7.3	7.3	7.12	1,61	220
ITBG	ISTRABENZ	▼ -2.99	69.85	-2.15	70	69.85	69.85	70	70.9	67.3	10,69	153
KBMR	NOVA KREDITNA BANKA MARIBOR	▼ -0.4	19.93	-0.08	20	19.71	19.7979	20	20	19.79	9,37	470
KDHP	KD HOLDING	▼ -1.81	35.35	-0.65	35.35	35.35	35.35	35.35	36	35.35	0,85	24
KDIR	KD ID	▲ 0.67	7.51	0.05	7.6	7.5	7.5111	7.5	7.97	7.51	12,70	1691
KRKG	KRKA	▼ -0.77	80.04	-0.62	81	79.55	79.8347	80.8	79.95	79.6	372,37	4652
KSFR	KS NALOŽBE	▼ -3.57	2.7	-0.1	2.7	2.7	2.7	2.7	2.8	2.7	3,82	1413
LKPG	LUKA KOPER	▼ -3.38	48.28	-1.69	49.99	47.5	48.2568	49.5	48.89	48.01	24,38	505
MAHR	MAKSIMA HOLDING	0	6.2	0	6.2	6.2	6.2	6.2	6.4	6.2	1,59	257
MELR	MERCATOR	▼ -2.98	207.05	-6.35	208.2	206.3	206.3	207.5	207	206.3	63,77	308
MIPG	MIP	▲ 4.58	2.51	0.11	2.51	2.51	2.51	2.51	3.99	2.5	0.28	110

Slika 10: Izsek spletne strani Dela s podatki o delnicah.



Slika 11: Gibanje tečaja Krke



Slika 12: Gibanje tečaja Petrola

Usmerjanje in razvrščanje opravil

Zgoraj opisani del projekta rešuje problem iskanja storitev in njihovega povezovanja v sestavljeno storitev. Če naj sestavljena storitev deluje čim hitreje in čimbolj učinkovito, mora to veljati tudi za posamične storitve, ki jo sestavljajo. Pri storitvah, dostopnih na enem samem mestu (na primer na enem spletnem strežniku), imamo zvezane roke. Pač pa se mnoge posamične storitve lahko izvajajo kot opravila v sistemu Grid, pri katerih pa je razvrščanje ključnega pomena za učinkovito izvajanje. V sklopu prijave raziskovalnega projekta smo zato zapisali, da bomo raziskali tudi uporabo preprostih metod umetne inteligence pri usmerjanju in razvrščanju opravil po zmogljivostih sistema Grid

Uvod

Pregled področja usmerjanja ter razvrščanja v porazdeljenih sistemih pokaže (Gaweda & Wilk, 2006) , da se v Grid sistemih – poleg dejanske izvedbe orodja – veliko posvečajo tudi usmerjanju in razvrščanju opravil (routing & scheduling). Prva močna ločnica med različnimi sistemi GRID je v pogledu, ki ga ima usmerjevalnik oz. razvrščevalnik na sistem. Če je sistem lokalen, gre bolj za razvrščanje opravil, ki so že prispela na izbrano vozlišče (po fazi odkrivanja zmogljivosti – ang. Resource Discovery). Diametralen pristop, kjer ima sistem za usmerjanje in razvrščanje vpogled v globalno sliko sistema GRID pa običajno pomeni, da gre bolj za usmerjanje opravil v vozlišča in je tako bistveno bližje fazi odkrivanja zmogljivosti. Iz opisane razlike med pristopoma lahko sklepamo tudi na različne podatke, ki jih imata na voljo. Medtem ko imamo pri razvrščanju natančnejši vpogled v delovanje vozlišča, je pri usmerjanju ta vpogled dokaj majhen – sistem gledamo kot celoto in tudi podatki (parametri) sistema, ki so na voljo, so globalne narave. Oba pristopa pa imata možnost uvajanja kriterijske funkcije, ki jo optimizirata. Običajno je to čas izvajanja poslov, čedalje pogosteje pa prihaja do merjenja kakovosti storitve (QoS – Quality of Service). Pri slednji se

meri predvsem zadovoljstvo uporabnika, ki je podal časovni okvir, v katerem se naj se opravilo izvede in je torej v tesni povezavi s časom izvajanja poslov.

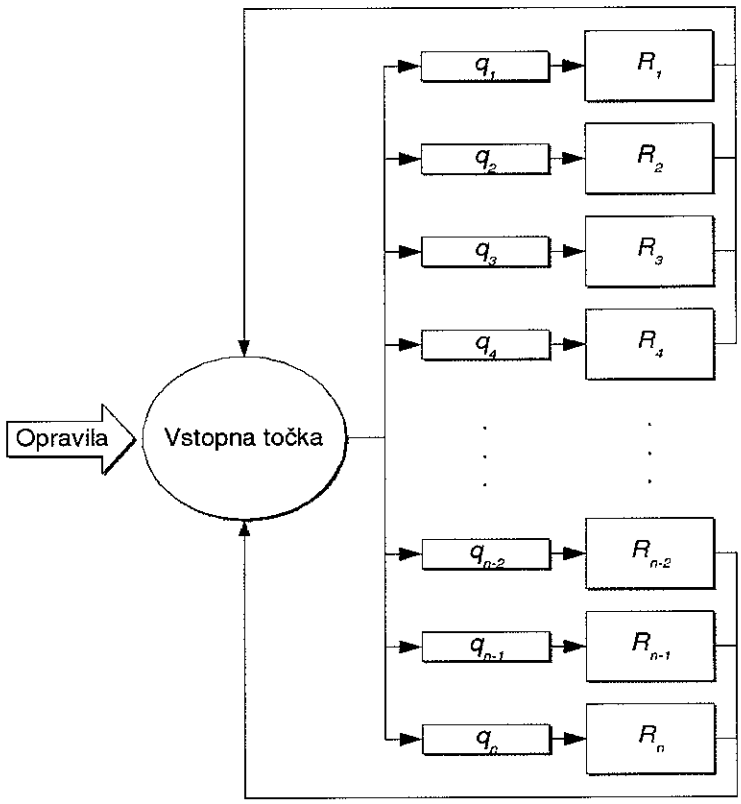
Trenutne rešitve na področju usmerjanja opravil delimo tudi glede na to, ali so statične ali dinamične. Med prve prištevamo rešitve, ki so algoritmične narave in se torej vedno odzovejo enako, med druge pa rešitve, ki temeljijo na umetni inteligenci, strojnem učenju ali statistiki. Med prvimi sta najpreprostejši naključno usmerjanje poslov ter usmerjanje po vrsti (ang., round robin).

Rešitve, ki temeljijo na statistiki, umetni inteligenci oz. strojnem učenju (Di Caro & Dorigo, 1998; Clark s sod., 2003; Itao s sod., 2001; Chen s sod., 2004) pogosto uporabljajo vzpodbujevano učenje (reinforcement learning). Najbolj pogosto uporabljena rešitev je pristop, imenovan Q-Learning (Boyan & Littman, 1994), kjer se z odzivom, ki ga dobijo pri določenem usmerjanju poslov, sčasoma naučijo primernega usmerjanja. Poleg pristopov z vzpodbujevanim učenjem, se pogosto uporabljajo tudi pristopi, ki temeljijo na genetskih algoritmih, ki pa običajno izračunajo strategijo že vnaprej. Eden od zanimivejših pristopov k problematiki je združevanje pristopov k planiranju (umetna inteligenca) ter pristopov k razvrščanju (optimizacija) (Smith & Johnson, 2000). Prednost takšnega hibridnega pristopa je upoštevanje dolgoročnega planiranja v kombinaciji s kratkoročnim razvrščanjem, kar daje dobre rezultate za primer planiranja proizvodnje.

V nadaljevanju bomo opisali arhitekturo sistema, postopek dela, nato pa predstavili naše izboljšave.

Arhitektura sistema

Slika 13 prikazuje visokonivojsko arhitekturo tipičnega sistema GRID. Vidimo, da je v takšnem sistemu le ena vstopna točka, v katero vstopajo opravila. Povezava med vstopno točko in zmogljivostmi, ki so na voljo (označena z R_1, ... R_n) je dvosmerna, saj vozlišča sporočajo rezultate v vstopno točko (natančneje, modul, ki je odgovoren za zbiranje rezultatov). Vidimo tudi, da so med vstopno točko in zmogljivostmi čakalne vrste (označene s q_1, ..., q_n) – izbira lokacije čakalnih vrst je poljubna in odvisna od implementacije sistema, saj imamo lahko namesto večjega števila čakalnih vrst tudi eno samo čakalno vrsto.



Slika 13: Arhitektura sistema

Nadaljujmo z natančnejšim opisom vstopne točke v sistem. Le-ta zajema naslednje sklope:

Podatkovna shramba

V tem modulu hranimo podatke o opravilih. Podatki so združeni, saj imamo na začetku zgolj opravilo, ki ga je potrebno izvesti, kasneje, ko pa je opravilo izvedeno, pa pridobimo še podatke o izvedbi. Podatki so sestavljeni iz naslednjih atributov:

- izvor opravila,
- pričakovani čas izvajanja oz. zahtevane zmogljivosti za opravilo,
- dejanski čas izvajanja opravila,
- vozlišče, na katerem se je opravilo izvajalo,
- stanje čakalne vrste vozlišča, ki je opravilo izvajalo,
- stanje čakalnih vrst ostalih vozlišč,
- dodatne atribute, ki so bili pripisani opravilu,
- oceno uspešnosti izvajanja opravila.

Ocena uspešnosti izvajanja je običajno izpeljana iz atributov dejanskega časa izvajanja in pričakovanega časa izračuna rezultatov. Tudi to oceno lahko poljubno definiramo, saj jo vstavimo v sistem kot kriterijsko funkcijo optimizacije.

Razvrščevalnik/Usmerjevalnik

Modul zajema algoritem, ki je lahko statičen in vnaprej znan, ter kot takšen vedno ponudi enako vozlišče za izvajanje naslednjega opravila, ali pa je zgrajen z metodami umetne inteligence, strojnega učenja oz. statistike. Modul lahko, ne glede na tip algoritma, uporablja podatke iz podatkovne shrambe za izbor naslednjega opravila.

Modul za strojno učenje

V primeru uporabe algoritmov strojnega učenja vstopna točka vsebuje tudi modul za strojno učenje. V tem modulu so zbrani algoritmi strojnega učenja, ki jih uporabljamo za učenje usmerjevalnika in razvrščevalnikov. Vhod v izbrani algoritem so podatki iz podatkovne shrambe, izhod pa model, s katerim napovedujemo vozlišče, ki bo, glede na podano kriterijsko funkcijo, imelo največjo verjetnost za uspešno izvajanje opravila.

Uporaba sistema

V zgoraj opisani sistem prihajajo opravila preko vstopne točke, ki jim dodeli, na podlagi različnih parametrov, vozlišče, na katerem se naj to opravilo izvaja. Opravilo se potem prestavi v čakalno vrsto izbranega vozlišča, ki po izvajanju tega opravila sporoči rezultate izvajanja nazaj v vstopno točko.
Če povzamemo korake:

1. Opravilo prispe v sistem.
2. Usmerjevalnik na podlagi določenega algoritma in stanja sistema izbere vozlišče, na katerem se bo opravilo izvajalo.
3. Opravilo se prestavi v čakalno vrsto izbranega vozlišča.
4. Po izvajanju opravila, se podatki o izvajanju vrnejo nazaj v vstopno točko.
5. Podatki o izvajanju se zabeležijo v podatkovno shrambo.

Glede na naš namen, da osnovno razvrščanje izboljšamo z uporabo preprostih algoritmov umetne inteligence in strojnega učenja, je potrebno k osnovnemu poteku izvajanja opravila dodati še naslednje korake:

- inicializacija sistema,
- učenje modela,
- ocenjevanje modela,
- napovedovanje z modelom,
- inkrementalno učenje modela.

Inicializacija sistema običajno zajema učenje modela s testnimi podatki. Le tako dobimo model, ki upošteva osnovne lastnosti sistema in se na njem naučimo dobrih parametrov, ki jih je moč uporabiti pri nadaljnjih učenjih sistema.

Učenje modela je učenje, z uporabo podatkovne shrambe, napovednega modela za napovedovanje vozlišča, ki bo z največjo verjetnostjo opravilo določeno opravilo.

Ocenjevanje modela se običajno opravlja periodično, s pomočjo znanih podatkov o opravili, ki smo jih shranili v podatkovno shrambo. Z ocenjevanjem modela ugotovimo, kako dobro izpolnjujemo želeno kriterijsko funkcijo.

Pomembno je poudariti zaporedje uporabe korakov takšnega sistema:

1. Inicializacija sistema.
2. Učenje modela.
3. Ocenjevanje modela.
4. Opravilo prispe v sistem.
5. Usmerjevalnik na podlagi določenega algoritma in stanja sistema izbere vozlišče, na katerem se bo opravilo izvajalo.
6. Usmerjevalnik uporabi napovedni model, ki lahko spremeni izbrano vozlišče
7. Opravilo se prestavi v čakalno vrsto izbranega vozlišča.
8. Po izvajanju opravila, se podatki o izvajanju vrnejo nazaj v vstopno točko.
9. Podatki o izvajanju se zabeležijo v podatkovno shrambo.
10. Inkrementalno učenje modela.

Vidimo torej, da se pri uporabi strojnega učenja dodajo samo določeni koraki k osnovnemu algoritmu, ki predstavljajo popravke osnovnega algoritma za usmerjanje.

Testna arhitektura

Zgoraj prikazano in opisano arhitekturo sistema smo implementirali v simulacijskem okolju PeerSim[2], za metode strojnega učenja pa smo uporabili programski paket WEKA[3]. Pri našem delu smo uporabljali predvsem dva algoritma strojnega učenja: naivni Bayes (angl. naive Bayes) in lokalno uteženo učenje (angl. locally weighted learning). Kot osnovni algoritem usmerjanja smo uporabili naključno razvrščanje ter razvrščanje po metodi round-robin. Za izbrani metodi strojnega učenja smo se odločili predvsem zaradi njune hitrosti napovedovanja ter robustnosti, saj majhne spremembe v učnih podatkih ne povzročijo velikih sprememb napovednega modela.

Vozlišča smo modelirali tako, da smo jim določili hitrost (kjer 1 pomeni normalno hitrost, <1 počasno vozlišče, >1 pa hitro vozlišče) ter verjetnost, da prenehajo delovati. Delo s tako sestavljenimi komponentami je omogočalo enostavno preverjanje hipotez o dodani vrednosti modula za strojno učenje, hkrati pa tudi pokazalo nekaj možnih težav.

V naslednjem razdelku predstavljamo eksperiment, ki je del članka, poslanega na konferenco "The IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN 2009)". V tem članku smo se ukvarjali predvsem s pregledovanjem lastnosti naše različice usmerjevalnika proti klasični različici.

Problem, ki smo ga reševali, je bil računanje minimalnega k-centra. Edini podatek o vsakem vhodnem opravilu je bil pričakovani čas izvajanja. Sistem je vseboval vstopno točko ter 8 računskih vozlišč, ki so bila različnih hitrosti: R_7-1.16, R_2-1.09, R_6-0.84, R_3-0.83, R_5-0.74, R_1-0.72, R_8-0.64 ter R_4-0.63, kjer števila pri oznaki vozlišča označujejo hitrost vozlišča (v skladu s prej vpeljano označbo hitrosti, sta tako najhitrejši vozlišči 7 in 2, ki jima sledita vozlišči 6 in 3, ostala vozlišča pa so bistveno počasnejša).

Učni podatki so bili sestavljeni iz naslednjih atributov:

- ocena časa izvajanja opravila,
- rok, do katerega se naj opravilo izvede,
- dolžina čakalne vrste v trenutku odločanja o ciljnem vozlišču,
- vozlišče, na katerem se je opravilo izvajalo,
- časovna ustreznost izvedbe (označena z dvema vrednostima, ki označujeta uspešno (v roku) ter neuspešno (prekoračen rok) izvedbo opravila).

Glede na specifično naravo problema smo opravila opisali zgolj s pričakovanim časom izvajanja, brez rokov izvedbe. Določili smo tudi istočasni začetek vseh opravil, saj so le-ta so med seboj neodvisna.

Fazo inicializacije smo izvedli s pomočjo sintetičnih testov. Generirali smo 1000 opravil, ki so imela določene čase začetka izvajanja ter rok izvedbe. Tako smo se naučili lastnosti sistema, kar v našem primeru pomeni predvsem hitrosti vozlišč.

Opravila za računanje minimalnega k-centra so bila urejena v 8 testnih množic. Ker smo za vsako od testnih množic izvedli poskus z in brez uporabe metode strojnega učenja, je bil končno število poskusov 16.

Uspešnost algoritmov smo merili z dvema atributoma: maksimalni ter kumulativni čas izvajanja. Prvi opisuje celotni čas do konca izvajanja zadnjega opravila, drugi pa čas dejanskega računanja (naj pripomnimo, da). Izboljšano metodo smo primerjali z osnovno round-robin s pomočjo relativnega prihranka pri času izvajanja $p_{relativni}$:

$$p_{relativni} = \frac{t_{round-robin} - t_{izboljšani}}{t_{round-robin}}.$$

Testna množica	Relativni prihranek pri maksimalnem času izvajanja	Relativni prihranek pri kumulativnem času izvajanja
Množica 1	0.321	0.309
Množica 2	0.001	0.282
Množica 3	0.249	0.332
Množica 4	0.229	0.286
Množica 5	0.396	0.267
Množica 6	0.001	0.284
Množica 7	0.410	0.283
Množica 8	0.000	0.281

Tabela 1: Primerjava izboljšane metode z osnovno.

V Tabeli 1 so prikazani rezultati primerjave obeh različic usmerjanja. Vidimo, da je moč doseči časovne prihranke tako pri kumulativnem kot maksimalnem času izvajanja. Podrobnejši pregled podatkov je pokazal, da izboljšani algoritem (Naive

Bayes) hitro prične izbirati vozlišča, ki so hitrejša, medtem ko osnovni algoritem round-robin ne loči med vozlišči – izboljšani algoritem je namreč več izbiral vozlišči 7 in 2.

Pri teh rezultatih poudarjamo, da je šlo za idealne pogoje simulacije, saj v realnem okolju ne moremo pričakovati, da bodo vozlišča ves čas na voljo. Poleg tega se lahko lastnosti vozlišč tudi spreminjajo, kar je v posebnem primeru, ki ga opisujemo, nemogoče zaznati. Na koncu tudi poudarjamo, da smo uporabili dober približek časa izvajanja (ki se ga da sicer pridobiti tudi iz drugih atributov).

Kljub naštetemu vsemu eksperiment prikazuje, da je uporaba preprostih metod strojnega učenja smiselna pri problemu usmerjanja opravil v sistemih GRID. Dejanski prihranki pri času so odvisni od uporabljenih algoritmov ter same arhitekture sistema. Na koncu naj poudarimo tudi, da uporaba metod strojnega učenja kot pomoči pri osnovnem usmerjanju ni brez nevarnosti, saj lahko zelo hitro pride do pretiranega prilagajanja (ang., over-fitting), ki ga je potrebno upoštevati pri delu.

Nadaljnje delo

Ker gre za razvojni projekt, kjer smo se osredotočili na uporabo semantike pri storitvah, je potrebno ogrodje za profesionalno delo razširiti predvsem z vidika robustnosti ter porabe računalniških virov. Ker gre za porazdeljen sistem, bo nadaljnje delo usmerjeno predvsem v obravnavo napak pri sporočilih med storitvami, centralnim imenikom ter izvrševalcem. Ena izmed smernic je uporabiti robustno infrastrukturo projekta XtreemOS za komunikacijo med porazdeljenimi objekti. (XtreemOS je projekt 6. okvirnega programa EU, ki razvija podporo sistemom grid na nivoju operacijskega sistema in na katerem podjetje XLAB sodeluje pri razvoju porazdeljene infrastrukture za izvajanje in demonstracijske aplikacije.

Nadaljnje delo je potrebno opraviti tudi pri razvoju hevristike za dopolnjevanje semantike prek povratne zanke. Trenutna rešitev namreč ni optimalna glede izrabe računalniških virov, predvsem pomnilnika.

V razdelku o primerih uporabe smo že nakazali uporabo ogrodja za lokalizacijo podatkov. Ker se naše podjetje aktivno ukvarja s prikazom lokacijskih podatkov, bomo ogrodju dodali storitve, ki bodo namesto izvoza podatkov v obliko HTML ter uporabe spletne aplikacije Google maps podatke izvažale v podatkovno bazo, dostopno prek protokola WFS (Web (geographical) Feature Service), ter jih prikazali v tridimenzionalnem prikazovalniku terena.

Spletno stran za oddaljeno izvajanje je potrebno dodelati, in sicer tako glede funkcionalnosti kot tudi videza in varnosti izvajanja.

Literatura

1. Hotho, A. et. al: Trend Detection in Folksonomies, Proc. First International Conference on Semantics and Digital Media Technology (SAMT), 56-70, 2006.

2. Hotho, A. et. al: Information Retrieval in Folksonomies: Search and Ranking, The Semantic Web: Research and Applications, 411-426, 2006.

3. Bouillet, E. et. al: A Folksonomy-Based Model of Web Services for Discovery and Automatic Composition, IEEE SCC (1), IEEE Computer Society, 389-396, 2008.

4. I. Ilijašić Mišić, B. Kovačič, T. Mohorić, D. Mladenić, B. Fortuna, M. Grobelnik: User study of ontology generation tool. 29th International conference on information technology interfaces, 529-533, Cavtat/Dubrovnik, 2007.

5. M. Grobelnik, D. Mladenić, B. Fortuna: From social network to light-weight ontology. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07), Hyderabad, India, 2007.

6. Google Maps API specification, http://code.google.com/apis/maps/ .

7. Gu, X. et. al: QoS-Assured Service Composition in Managed Service Overlay Networks, International Conference on Distributed Computing Systems (ICDCS), 2004.

8. Hu, Z. et. al: A Distributed Algorithm for DAG-Form Service Composition Over MANET, Wireless Communications, Networking and Mobile Computing (WiCOM), 1664-1667, 2007.

9. Klusch, M. et.al: Automated semantic web service discovery with OWLS-MX, International Conference of Autonomous Agents and Multiagent Systems (AAMAS), 915-922, 2006.

10. Boyan, J. A., Littman, M. L. (1994). Packet routing in dynamically changing networks: A reinforcement learning approach. Advances in Neural Information Processing Systems, str.. 671–678). Morgan Kaufmann Publishers, Inc.

11. Chen M., Zheng A., Lloyd J., Jordan M., Brewer E. (2004) Failure diagnosis using decision trees. Proceedings of International Conference on Autonomic Computing.

12. Clark, D. D., Partridge, C., Ramming, J. C., & Wroclawski, J. (2003). A knowledge plane for the internet. Proceedings of ACM SIGCOMM.

13. Di Caro, G., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 9, 317–365.

14. Itao, T., Suda, T., & Aoyama, T. (2001). Jack-in-the-net: Adaptive networking architecture for service emergence. Proceedings of the Asian-Pacific Conference on Communications.

15. Gaweda I., Wilk C. (2006) Grid Brokers And MetaSchedulers Market Overviews. Gridwise Tech, Poland, http://www.gridwisetech.com/metaschedulers .

3. Izkoriščanje dobljenih rezultatov:

3.1. Kakšen je potencialni pomen[4] rezultatov vašega raziskovalnega projekta za:

- ☒ a) odkritje novih znanstvenih spoznanj;
- ☒ b) izpopolnitev oziroma razširitev metodološkega instrumentarija;
- ☐ c) razvoj svojega temeljnega raziskovanja;
- ☐ d) razvoj drugih temeljnih znanosti;
- ☒ e) razvoj novih tehnologij in drugih razvojnih raziskav.

3.2. Označite s katerimi družbeno-ekonomskimi cilji (po metodologiji OECD-ja) sovpadajo rezultati vašega raziskovalnega projekta:

☐ a) razvoj kmetijstva, gozdarstva in ribolova - Vključuje RR, ki je v osnovi namenjen razvoju in podpori teh dejavnosti;

☒ b) pospeševanje industrijskega razvoja - vključuje RR, ki v osnovi podpira razvoj industrije, vključno s proizvodnjo, gradbeništvom, prodajo na debelo in drobno, restavracijami in hoteli, bančništvom, zavarovalnicami in drugimi gospodarskimi dejavnostmi;

☐ c) proizvodnja in racionalna izraba energije - vključuje RR-dejavnosti, ki so v funkciji dobave, proizvodnje, hranjenja in distribucije vseh oblik energije. V to skupino je treba vključiti tudi RR vodnih virov in nuklearne energije;

☐ d) razvoj infrastrukture - Ta skupina vključuje dve podskupini:

- transport in telekomunikacije - Vključen je RR, ki je usmerjen v izboljšavo in povečanje varnosti prometnih sistemov, vključno z varnostjo v prometu;
- prostorsko planiranje mest in podeželja - Vključen je RR, ki se nanaša na skupno načrtovanje mest in podeželja, boljše pogoje bivanja in izboljšave v okolju;

☐ e) nadzor in skrb za okolje - Vključuje RR, ki je usmerjen v ohranjevanje fizičnega okolja. Zajema onesnaževanje zraka, voda, zemlje in spodnjih slojev, onesnaženje zaradi hrupa, odlaganja trdnih odpadkov in sevanja. Razdeljen je v dve skupini:

☐ f) zdravstveno varstvo (z izjemo onesnaževanja) - Vključuje RR - programe, ki so usmerjeni v varstvo in izboljšanje človekovega zdravja;

☐ g) družbeni razvoj in storitve - Vključuje RR, ki se nanaša na družbene in kulturne probleme;

☐ h) splošni napredek znanja - Ta skupina zajema RR, ki prispeva k splošnemu napredku znanja in ga ne moremo pripisati določenim ciljem;

☐ i) obramba - Vključuje RR, ki se v osnovi izvaja v vojaške namene, ne glede na njegovo vsebino, ali na možnost posredne civilne uporabe. Vključuje tudi varstvo (obrambo) pred naravnimi nesrečami.

---

[4] Označite lahko več odgovorov.

3.3. Kateri so **neposredni rezultati** vašega raziskovalnega projekta glede na zgoraj označen potencialni pomen in razvojne cilje?

Razvito ogrodje za sklapljanje storitev je tehnologija, neposredno uporabna za hiter razvoj sestavljenih storitev, za katere nato razvijemo še spletni uporabniški vmesnik in jih ponudimo končnim uporabnikom. V tehničnem delu poročila smo navedli številne zanimive primere uporabe, ki jih je moč takoj uporabiti v gospodarstvu.

3.4. Kakšni so lahko **dolgoročni rezultati** vašega raziskovalnega projekta glede na zgoraj označen potencialni pomen in razvojne cilje?

Pri razvoju sistema za sklapljanje storitev smo prišli do temeljnega spoznanja, da namesto prezahtevnega popolno samodejnega sklapljanja storitev do uporabnih rezultatov pridemo s preprostejšim principom cevovodenja.

Poskusi z inteligentnim razvrščanjem opravil so dokazali, da že z uporabo razmeroma preprostih metod umetne inteligence lahko bistveno pohitrimo izvajanje množice opravil v sistemu Grid. Hkrati smo razvili metode za primerjavo razvrščevalnikov, uporabne tudi v drugih podobnih poskusih.

3.5. Kje obstaja verjetnost, da bodo vaša znanstvena spoznanja deležna zaznavnega odziva?

- ☐ a) v domačih znanstvenih krogih;
- ☒ b) v mednarodnih znanstvenih krogih;
- ☒ c) pri domačih uporabnikih;
- ☒ d) pri mednarodnih uporabnikih.

3.6. Kdo (poleg sofinancerjev) že izraža interes po vaših spoznanjih oziroma rezultatih?

Poleg partnerjev v projektih 6. in 7. okvirnega programa izražajo interes naslednja domača in tuja podjetja, s katerimi člani projektne skupine tudi sicer sodelujejo: SIMT, Hermes Softlab, Amebis, Cycorp, British Telecom, Microsoft, New York Times.

3.7. Število diplomantov, magistrov in doktorjev, ki so zaključili študij z vključenostjo v raziskovalni projekt?

Študij so zaključili trije magistri (Jaka Močnik, Uroš Jovanovič, Miha Vuk) in en diplomant (Blaž Fortuna).

4. Sodelovanje s tujimi partnerji:

4.1. Navedite število in obliko formalnega raziskovalnega sodelovanja s tujimi raziskovalnimi inštitucijami.

XLAB sodeluje na dveh evropskih projektih na področju sistemov Grid: XtreemOS (6. okvirni program) in SLA@SOI (7. OP).

IJS sodeluje na več evropskih projektih na področju omrežnih ontologij, kolaborativnih aspektih in evoluciji ontologij, geolokacijskih ontologijah in semantičnega spleta: NEON, SWING, TAO, ACTIVE.

4.2. Kakšni so rezultati tovrstnega sodelovanja?

Primeri uporabe iz projekta SLA@SOI bodo služili nadaljnji dodelavi in testiranju ogrodja za sklapljanje storitev, pa tudi diseminaciji rezultatov. Po drugi plati projekt XtreemOS teče že dve leti, zato smo tam pridobljene izkušnje uporabili pri razvoju razvrščevalnika in usmerjevalnika opravil kot tudi metodologije za njegovo testiranje.

V projektu NEON smo razvili pristop za napovedovanje strukturnih sprememb ontologije in povezovanjem konceptov ontologije z uporabo metod strojnega učenja, razvili pristop za uporabo velikih ontologij kot konteksta, ki omogoča učinkovito izvedbo osnovnih operacij na ontologiji, in razvili sistem za vizualizacijo ontologij v kontekstu podanega ozadja (landscape).

Na projektu SWING smo razvili OntoBridge, sistem za polavtomatsko označevanje ontologij z uporabo metod strojnega učenja.

Rezultati našega dela na projektu TAO zajemajo razvoj pristopa za analizo programske opreme (software-mining) z luščenjem znanja iz izvorne kode in dokumentacije na osnovi metod analize besedil in analize povezav (text mining and link analysis), implementacijo sistema za renderiranje in algoritma za risanje grafov, uporabno za vizualizacijo semantičnega prostora, ki je lahko v bistveno pomoč razvijalcu sestavljenih storitev.

5. Bibliografski rezultati[5] :

Za vodjo projekta in ostale raziskovalce v projektni skupini priložite bibliografske izpise za obdobje zadnjih treh let iz COBISS-a) oz. za medicinske vede iz Inštituta za biomedicinsko informatiko. Na bibliografskih izpisih označite tista dela, ki so nastala v okviru pričujočega projekta.

[5] Bibliografijo raziskovalcev si lahko natisnete sami iz spletne strani:http:/www.izum.si/

6. Druge reference[6] vodje projekta in ostalih raziskovalcev, ki izhajajo iz raziskovalnega projekta:

[6] Navedite tudi druge raziskovalne rezultate iz obdobja financiranja vašega projekta, ki niso zajeti v bibliografske izpise, zlasti pa tiste, ki se nanašajo na prenos znanja in tehnologije.
Navedite tudi podatke o vseh javnih in drugih predstavitvah projekta in njegovih rezultatov vključno s predstavitvami, ki so bile organizirane izključno za naročnika/naročnike projekta.