Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
ŠTÚDIJNÝ PROGRAM: Softvérové inžinierstvo

Bc. Michal Jemala

Vizuálne prehľadávanie RDF dokumentov

Diplomová práca

Vedúci diplomovej práce: prof. Ing. Mária Bieliková, PhD.
december 2006

Prehlasujem, že som diplomovú prácu vypracoval samostatne na základe štúdiom nadobudnutých poznatkov a uviedol som všetku použitú literatúru.

Michal Jemala

Na tomto mieste by som chcel poďakovať
prof. Ing. Márii Bielikovej, PhD. za ochotu, cenné
rady a pripomienky, ktorými ma usmerňovala pri
vypracovávaní tejto diplomovej práce

Chcel by som taktiež poďakovať mojim rodičom
a bratom za ich podporu a trpezlivosť.

Zadanie

Sem vložiť namiesto tejto strany zadanie

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný odbor: SOFTVÉROVÉ INŽINIERSTVO

Autor: Michal Jemala
Diplomový projekt: Vizuálne prehľadávanie RDF dokumentov
Vedúci diplomového projektu: prof. Ing. Mária Bieliková, PhD.
December, 2006

Práca sa zaoberá problematikou vizualizácie rozsiahlych priestorov metadát, ktoré obohacujú informácie uvedené na webe. Metadáta sú reprezentované jazykmi ako RDF, RDFS a OWL. Cieľom je umožniť používateľom efektívne sa orientovať a prehľadávať takéto priestory. Analyzujú sa existujúce metódy, techniky a samotné vizualizačné nástroje, pričom snahou je určiť ich výhody a nevýhody. Poznatky získané z procesu analýzy, slúžia ako podklad pri návrhu vizualizačnej metódy, ktorá umožňuje prehliadať priestor inkrementálne. Metóda vizualizuje v každom momente iba určitú časť priestoru, tzv. okno, pričom obsah okna zohľadňuje ohodnotenie jednotlivých entít získané na základe interakcie s používateľmi alebo externými aplikáciami. Výhodou takejto vizualizačnej metódy je možnosť prezentovať používateľovi dostatočné množstvo detailov pri súčasnom zachovaní primeranej úrovne globálneho prehľadu. Overením navrhnutej metódy je experimentálny prototyp vizualizačného nástroja umožňujúci prehliadať štruktúru ontológie v textovom a grafickom režime.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: SOFTWARE ENGINEERING

Author: Bc. Michal Jemala
Diploma project: Visual browsing of RDF documents
Supervisor: prof. Ing. Mária Bieliková, PhD.
2006, December

This work discusses the problematic of visualization of extensive metadata spaces. Metadata are represented with languages as RDF, RDFS or OWL, which are appointed to enhance information published on the web. Point of this work is to analyze existing methods, techniques and visualization tools and find out its advantages and disadvantages. Knowledge acquired in this process represents the basis for design of a visualization method, which allows incrementally browsing of extensive metadata spaces. This method visualizes in every moment only specific part of space, named window, where the content of this window is created with according to evaluations of space entities. Entity evaluations are gained from interaction with users or external applications. The main advantage of this approach is the possibility to present to the user appropriate level of entity details and simultaneously preserve adequate level of global overview. The verification of designed method is the implementation of experimental prototype which can visualize ontologies in textual or graphical modes.

Obsah

1. Úvod

Súčasný web obsahuje nepreberné množstvo informácií, ktoré sa ďalej neustále zväčšuje. Jednou z možností ako aspoň čiastočne zvládnuť takúto situáciu, je automatizovanie jednotlivých činností podporené technológiami webu so sémantikou. Víziou webu so sémantikou je definovanie a prepojenie informácií na webe takým spôsobom, že budú môcť byť identifikovateľné, spracovávateľné a vyhodnocované strojmi. Aby sme dosiahli tento cieľ, je nutné zaviesť formalizovaný spôsob opisu informácií publikovaných na webe, pričom tento by sa mal stať štandardom. V minulosti existovali viaceré pokusy definovať jazyky umožňujúce formálny opis informácií na webe, niektoré z týchto jazykov sa stále ešte používajú (Cyc, Ontolingua, OCML, DAML, OIL), iné sú už len históriou (SHOE, Ontobroker). Aktuálne sa masívne presadzuje iniciatíva konzorcia W3[1], ktoré sa snaží o štandardizáciu v tejto oblasti. Výsledkom sú jazyky RDF, RDFS, OWL.

Metadáta zapísane pomocou týchto jazykov umožňujú špecifikovať sémantiku webových stránok, prípadne vyjadriť špecifické pojmy a vzťahy v rámci určitej konkrétnej oblasti záujmu (napr. doména pracovných ponúk). Problémom môže byť komplexnosť a rozsah takejto formálnej špecifikácie. Preto je nutné identifikovať vhodné metódy, ktoré umožnia skúmať, prehľadávať a narábať s takýmito rozsiahlymi priestormi metadát. Jednou z nich, je možnosť vizualizovať priestor metadát takým spôsobom, ktorý umožní eliminovať problém rozsiahlosti takéhoto priestoru.

Cieľom tejto práce je analyzovať problematiku vizualizácie metadát zapísaných formou RDF, RDFS a OWL. Identifikovať používané vizualizačné techniky a existujúce nástroje, ktoré takúto funkcionalitu poskytujú, vyhodnotiť ich klady a zápory a na základe nadobudnutých poznatkov navrhnúť vizualizačnú metódu, ktorá umožní efektívnejšie vizualizovať takéto metadáta a vytvorením prototypu overiť navrhnutú metódu.

Práca je rozdelená do siedmych kapitol. Kapitola 1 predstavuje úvod do problematiky. V kapitole 2 analyzujeme problematiku vizualizácie ontológií, opisujeme existujúce vizualizačné techniky, poskytujeme prehľad a porovnanie v súčasnosti najpoužívanejších vizualizačných nástrojov. V kapitole 3 formulujeme ciele práce stanovené na základe vyhodnotenia analýzy. Kapitola 4 sa venuje návrhu vizualizačnej metódy, ktorá umožňuje inkrementálne prehliadať rozsiahle priestory, pričom pri vizualizácii zohľadňuje aj interakciu používateľov resp. aplikácií s entitami vizualizovaného priestoru. Kapitola 5 obsahuje návrh vizualizátora, ktorý využíva navrhnutú metódu. V kapitole 6 popisujeme prototyp vizualizátora implementovaného s cieľom overiť navrhnutú vizualizačnú metódu. Záverečná kapitola obsahuje celkové zhodnotenie dosiahnutých výsledkov a návrh možných vylepšení a rozšírení.

[1] http://www.w3.org

Práca ďalej obsahuje päť príloh. V prílohe A sa nachádza technická dokumentácia, príloha B obsahuje inštalačný postup a používateľskú príručku k implementovanému prototypu vizualizátora. Príloha C predstavuje základné charakteristiky jazykov RDF, RDFS a OWL. V prílohe D opisujeme ontológiu pracovných ponúk, ktorú sme využívali počas analýzy a návrhu, ale najmä pri implementácii a testovaní prototypu. V prílohe E je uvedená štruktúra a popis priloženého elektronického média.

2. Vizualizácia ontológií

Pojem ontológia chápeme v kontexte filozofie ako náuku o bytí, resp. ako univerzálnu sústavu znalostí popisujúcich objekty, javy a zákonitosti sveta „tak ako je“, t.j. nezávisle od ľudského usudzovania o ňom.

V kontexte informatiky opisuje ontológia to, čo už existuje a môže byť reprezentované v informačnom alebo v znalostnom systéme. Formálne je ontológia definovaná ako *explicitná formálna špecifikácia zdieľanej konceptualizácie*[2] [18]. Táto definícia požaduje vyjadriť konceptualizáciu, t.j. systém pojmov modelujúci určitú časť sveta, prostredníctvom vyjadrovacieho prostriedku s jasne definovanou syntaxou, navyše sa požaduje, aby sa takto vyjadrená konceptualizácia dala opakovane použiť a aby bola navonok prístupná.

V literatúre sa najčastejšie spomínajú nasledovné spôsoby využitia ontológií [18]:

- podpora porozumenia medzi ľuďmi (napr. medzi expertmi a znalostnými inžiniermi),
- podpora komunikácie medzi počítačovými systémami,
- uľahčenie návrhu znalostne-orientovaných aplikácií.

Pričom vo všetkých troch úlohách sa ontológie môžu uplatniť v širokom spektre problémových oblastí a úloh. Ontológie je možné rozčleniť na základe niekoľkých kritérií. V kontexte s historickými informatickými disciplínami rozlišujeme [18]:

1. *Terminologické ontológie* predstavujú pokročilejšie synonymické slovníky, ktoré sa používajú v knihovníctve a oboroch zameraných prevažne na textové informácie.
2. *Informačné ontológie* reprezentujú rozvinutie databázových konceptuálnych schém, ktoré zaisťujú abstrakciu a vyššiu kontrolu integrity.
3. *Znalostné ontológie* sa týkajú reprezentácie znalostí v rámci umelej inteligencie.

Dôležitým hľadiskom je taktiež predmet formalizácie, resp. to, čím sa konceptualizácia reprezentovaná ontológiou zaoberá [18]:

a. *Doménové ontológie*, ktorých predmetom je špecifická vecná oblasť (napr. problematika medicíny alebo fungovania firmy, problematika určitej choroby alebo poskytovania úveru).
b. *Generické ontológie* sa usilujú o zachytenie všeobecných zákonitostí, ktoré platia medzi vecnými oblasťami (napr. problematiky času, vzájomné pozície objektov, skladby objektov).
c. *Úlohové ontológie* sa zameriavajú na procesy odvodzovania (napr.: diagnostika, zhodnocovanie, konfigurácie alebo plánovanie).

[2] Predstavuje pôvodnú definíciu T. Grubera „*explicitná špecifikácia konceptualizácie*“, doplnenú W. Borstem o pojmy *explicitná* a *zdieľaná*

d. *Aplikačné ontológie* sú najšpecifickejšie. Predstavujú súhrn modelov prevzatých a adaptovaných pre konkrétnu aplikáciu.

Základným prvkom ontológií sú *triedy*. Triedy predstavujú konkrétnu množinu objektov, ktoré sú zväčša hierarchicky usporiadané. Trieda sa v kontexte ontológií líši od triedy, tak ako ju poznáme z objektovo-orientovaného programovania. Triedy sa interpretujú skôr ako unárne relácie definované na danej doméne objektov. Rozlišujeme dva druhy tried:

a. *Definované triedy* sú špecifikované postačujúcimi a nutnými podmienkami.

b. *Primitívne triedy* sa označujú všetky ostatné triedy.

Inštancie tried sa nazývajú *indivíduá* a reprezentujú konkrétne objekty záujmu v danej doméne. Inštancie sú identifikované jednoznačným identifikátorom, t.j. pomocou URI (angl. Uniform Resource Locator).

V prípade ontológií je možné definovať relácie n-tíc objektov. Pre binárne relácie sa používa pojem *slot* alebo *vlastnosť*. Avšak na rozdiel od objektovo-orientovaných modelov, v prípade ontológie sú vlastnosti rovnocenné triedam. Možné je taktiež definovať vlastnostiam vlastnosti, t.j. akési meta-vlastnosti. Typickým prípadom je vzťah *„is a“*, ktorý umožňuje definovať hierarchiu.

2.1 Princípy vizualizácie ontológií

Vizualizácia ontológií predstavuje zobrazenie tried, inštancií a vlastností. Vo všeobecnosti pomáha vizualizácia lepšie a rýchlejšie pochopiť a následne analyzovať dáta, ktoré ontológia opisuje.

Za vizualizáciu ontológie možno považovať každé zobrazenie iné ako jej zápis v zdrojovom formáte. Prirodzeným prostriedkom na vizualizáciu ontológií je graf, kde uzly predstavujú triedy a inštancie a hrany reprezentujú vlastnosti. Avšak niektoré ontológie (napr. WordNet[3] alebo Sweto[4]) je veľmi ťažko vizualizovať priamo vo forme grafu, pretože obsahujú obrovský počet uzlov a hrán. Vizualizácia takýchto rozsiahlych štruktúr predstavuje výzvu a vyžaduje si aplikáciu špecifických vizualizačných techník, metód a princípov. Napríklad vo väčšine prípadov nie je nutné zobraziť celý graf, navyše aj z hľadiska pochopenia je lepšie poskytnúť používateľovi postupne ucelené a čo najviac súvisiace časti grafu a následne prostredníctvom navigačných, filtrovacích a vyhľadávacích techník postupne prehľadávať aj ostatné časti grafu.

[3] WordNet predstavuje sémantický lexikón anglického jazyka. V roku 2005 obsahoval približne 150 000 slov a v komprimovanej podobe mal 12MB. http://wordnet.princeton.edu.

[4] Semantic Web Technology Evaluation Ontology predstavuje ontológiu určenú na porovnávanie existujúcich nástrojov pre prácu s ontológiami. Verzia 1.3 obsahuje viac ako 800 tis. tried a viac ako 1,5mil. vlastností. http://lsdis.cs.uga.edu/projects/semdis/sweto

2.2 Vizualizačné techniky

Existuje viacero vizualizačných techník, ktoré sú viac či menej vhodné na vizualizáciu dát reprezentovaných vo forme ontológií. V nasledujúcej kapitole si rozoberieme najrozšírenejšie z nich.

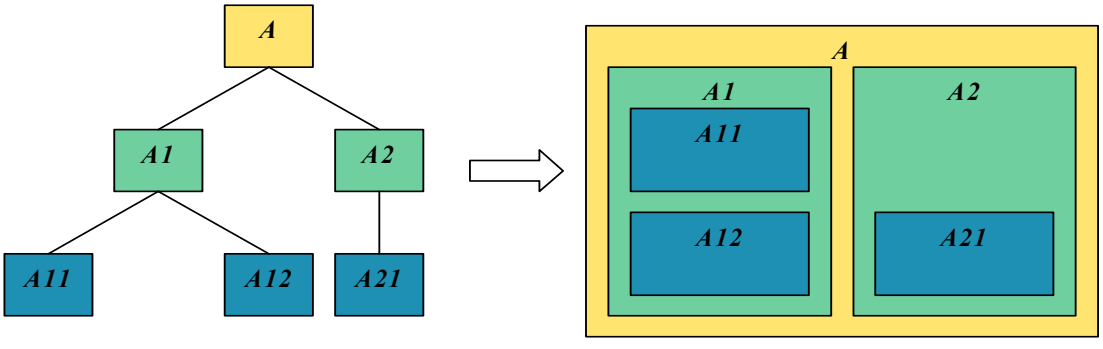
2.2.1 Stromy

Ontológie sa zobrazujú vo forme stromov, zdôraznená je teda hierarchia tried. Problém je, že takmer žiadna ontológia neobsahuje iba hierarchické vzťahy. Okrem nich, sú zväčša definované vzťahy medzi jednotlivými triedami na rôznej úrovni hierarchie. Výsledkom čoho je často neprehľadná spleť čiar, ktorá degraduje výslednú vizualizáciu. Riešením tohto problému sú najčastejšie rôzne filtrovacie mechanizmy, ktoré umožňujú z vizualizácie odstrániť nepotrebné väzby (negatívne filtrovanie).

Ďalším problémom je rozmiestňovanie jednotlivých tried, ak ontológia obsahuje veľké množstvo tried na jednej úrovni hierarchie. V tomto prípade dostávame príliš „košatý“ strom, ktorý nie je možné rozumne vizualizovať v celom rozsahu. Riešením tohto problému je, buď možnosť expandovať iba požadované vrcholy (pozitívne filtrovanie) alebo premietnuť strom na zvolenú kvadratickú plochu (guľa, elipsa, hyperbola, atď.), pričom sa využívajú vlastnosti perspektívneho zobrazenia (veľkosť uzla, a tým aj úroveň detailov, sa zmenšuje so rastúcou vzdialenosťou uzla od koreňa). Samozrejme uvedené spôsoby je možné kombinovať.

2.2.2 Vnorený graf

Vizualizačná technika vnorený graf (angl. nested graph), sa používa najmä na vizualizáciu hierarchických vzťahov. Avšak je možné zakomponovať do vizualizácie aj iné väzby. Princíp tejto techniky je uvedený na Obr. 1 b), pričom na Obr. 1 a) je uvedený prislúchajúca hierarchia. Takýmto prístupom sa čiastočne eliminuje veľký počet hrán a sprehľadňuje sa vizualizácia.



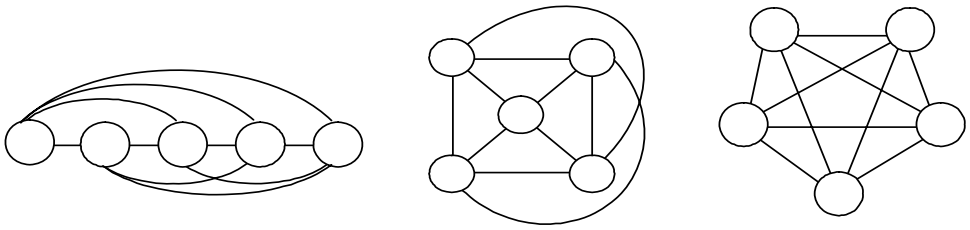
Obr. 1 Princíp vizualizačnej techniky vnorený graf

Avšak ak je v ontológii definovaná príliš hlboká hierarchia, vnorenie ju neumožní zobraziť v celom rozsahu. Najčastejšie sa takýmto spôsobom vizualizujú iba susediace úrovne hierarchie. Riešením je

pri prehľadávaní „odsekávať“ podstromy a zobrazovať aktuálny uzol ako koreň stromu, t.j. postupne sa vnárať do hierarchie. Takáto technika predstavuje efektívny spôsob navigácie daného vizualizačného nástroja, hlavne v prípade, ak je skombinovaná s globálnym pohľadom na ontológiu.

2.2.3 Grafy

V tomto prípade si z hľadiska vizualizácie všetky triedy v ontológií rovnocenné, a predstavujú uzly grafu. Hrany sú orientované a zodpovedajú jednotlivým reláciám medzi triedami. V prípade tejto vizualizačnej techniky je zvlášť dôležitý prepracovaný rozmiestňovací algoritmus, ktorý by mal zabezpečiť čo prehľadnejšie zobrazenie jednotlivých uzlov tak, aby sa hrany čo najmenej prekrývali prípadne, aby súvisiace triedy boli aj geometricky blízko. Tri rôzne prípady rozmiestnenia piatich uzlov je zobrazený na Obr. 2. V prípadoch a), b) máme len jedno skríženie, v prípade c) dokonca 5. Algoritmus rozmiestňovania výrazne ovplyvňuje prehľadnosť vizualizácie.

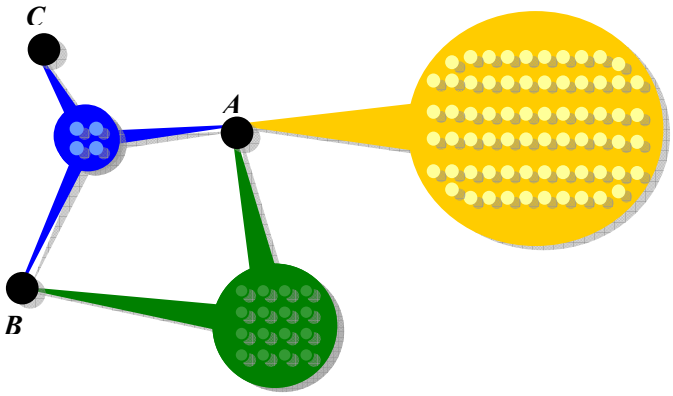


a) b) c)

Obr. 2 Príklad rozmiestnenia 5 uzlov (10 hrán, „každý s každým“)

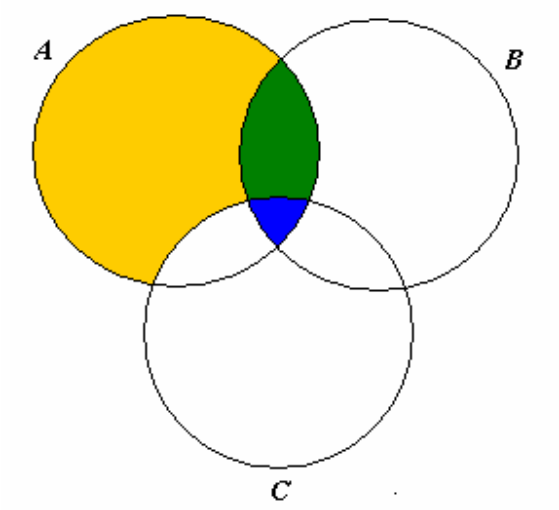
2.2.4 Zhluky

Táto vizualizačná technika vychádza z vennových diagramov [8]. Umožňuje vizualizovať prekrývanie inštancií tried v ontológiách. Prekrývaním inštancií (angl. instance overlapping) sa označuje situácia, kedy daná inštancia prislúcha viacerým triedam. Na Obr. 3 je uvedený príklad inštancií tried A, B a C.



Obr. 3 Princíp vizualizačnej techniky zhlukovania

Inštancie v žltom zhluku prislúchajú iba triede ***A***, inštancie v zelenom zhluku triedam ***A*** a zároveň ***B*** a inštancie v modrom zhluku triedam ***A*** a zároveň ***B*** a zároveň ***C***. Obr. 4 ilustruje rovnakú situáciu vizualizovanú pomocou vennových diagramov.



Obr. 4 Analogický prípad vizualizovaný pomocou vennových diagramov

2.2.5 Mapy

Mapy[5] sa využívajú rovnaký princíp ako vnorený graf, na vyjadrenie hierarchie, avšak jednotlivé uzly sú navyše ohodnotené na základe zvoleného kritéria. Toto ohodnotenie slúži na určenie plochy, ktorú bude daný uzol vo vizualizáciu zaberať. Pričom kritériá, podľa ktorých sa táto hodnota vyjadrí, sú rôzne (počet potomkov, počet nasledovníkov, počet relácií, počet inštancií, atď.). Techniku máp je vhodné použiť, ak chceme zistiť alebo explicitne vyjadriť kvantitatívne charakteristiky tried alebo inštancií v ontológii. Princíp tejto techniky ilustruje náčrt na Obr. 5.

Obr. 5 Princíp vizualizačnej techniky mapa

[5] Niektoré zdroje uvádzajú aj pojem stromové mapy (angl. tree maps), aby zdôraznili, že sa jedná o vizualizáciu hierarchických (stromových) štruktúr [5].

2.2.6 Text

Textové informácie sa používajú v každej vizualizácii, umožňujú identifikovať jednotlivé triedy, inštancie a vlastnosti. Zobrazujú doplňujúce informácie, ktoré sú resp. môžu byť súčasťou ontológie, ale z hľadiska prehľadnosti je vhodné ich zobrazovať osobitným spôsobom (napr. `rdfs:label` `rdfs:comment`). Príkladom môže byť samostatné okno alebo dialóg v programe zobrazujúci v štruktúrovanej podobe informácie o práve zvolenej triede, inštancii alebo vlastnosti.

2.3 Prehľad existujúcich nástrojov

V tejto časti sumarizujeme aktuálny stav v oblasti vizualizácie ontológií. Vybrali sme najpoužívanejšie existujúce nástroje, ktoré kategorizujeme a opisujeme ich funkcionalitu, vlastnosti a špecifiká. Na záver uvádzame porovnanie jednotlivých nástrojov s cieľom určiť najvhodnejší nástroj pre jednotlivé kategórie, ktoré sme identifikovali.

2.3.1 Kategorizácia

Existujúce nástroje umožňujúce vizualizovať ontológie možno zradiť do viacerých kategórií podľa rozličných kritérií. Keďže funkcionalita existujúcich vizualizačných nástrojov je rozmanitá, nie je možné vo veľkej väčšine prípadov zaradiť nástroje výlučne do jednej kategórie. Kategorizáciu môžeme vykonať na základe nasledovných kritérií:

a) *životný cyklus ontológie* – sem patria nástroje podporujúce *vývoj, napĺňanie a používanie ontológií.* V každom z uvedených prípadov sa kladie pri vizualizácii dôraz na iné aspekty. V prípade vývoja ontológie je nutná dostatočná úroveň detailov. Dôležité je umožniť používateľovi editovať ontológiu, t.j. jednoduchým spôsobom pridávať a mazať jednotlivé koncepty a ich vlastnosti. V prípade napĺňania ontológie, nehrá vizualizácia konceptov a ich vlastností dôležitú úlohu, väčší dôraz sa kladie na podporu automatizovaného procesu a samotná vizualizácia je voliteľná. Dôležitá je ale efektívna navigácia a zabezpečenie korektnosti vstupov. Posledný prípad, používanie ontológie, zahŕňa širokú škálu požiadaviek na funkcionalitu vizualizačného nástroja. Počnúc analyzovaním, cez vyhľadávanie až po navigáciu v ontológiách. Vizualizácia v každom z týchto prípadov umožňuje efektívnejšie realizovanie používateľských činností.

b) *typ ontológie*[6] – sem možno zaradiť nástroje od *generických*, ktoré umožňujú vizualizovať ľubovoľný typ ontológie, cez nástroje, ktoré vizualizujú doménové a úlohové ontológie (vtedy vizualizácia môže zahŕňať aj črty špecifické iba v danej doméne, resp. pre danú úlohu), až po

[6] Ide o typ ontológie z pohľadu predmetu formalizácie. Bližšie pozri [18]

špecifické, ktoré umožňujú vizualizovať aplikačné ontológie, v ktorých vizualizácia zohľadňuje konkrétne požiadavky danej aplikácie [15]

c) *vizualizačná technika* – používané vizualizačné techniky sú podrobne opísané v kapitole 2.2. Treba poznamenať, že použitá vizualizačná technika súvisí s etapou v životnom cykle ontológie. Použitie niektorých techník je výhodnejšie v etape vývoja (strom, graf), iné zasa pri analýze, v etape používania ontológií (mapy, zhluky).

2.3.2 Charakteristika vybraných nástrojov určených na vizualizáciu ontológií

Kapitola obsahuje stručný opis nástrojov určených na vizualizáciu a prácu s ontológiami. Jednotlivé nástroje sú opisované z pohľadu nasledovných základných funkcií, ktoré boli identifikované:

1. vizualizačná technika,
2. navigácia v ontológii,
3. filtrovanie konceptov a ich vlastností,
4. vyhľadávanie v ontológii,
5. zobrazovanie detailov (doplňujúce textové informácie),
6. rozmiestňovacie algoritmy,
7. export vizualizácie a podpora dokumentovania.

Protégé

Protégé[7] je voľne dostupná platforma, ktorá poskytuje podporu pre prácu s ontológiami. Základom Protégé je bohatá množina štruktúr a akcií, ktoré podporujú vytváranie, vizualizáciu a editovanie ontológií. Protégé taktiež umožňuje definovať formuláre na vkladanie inštancií tried. Vytvorené ontológie je možné exportovať do najrôznejších formátov ako je napr. OWL, RDF, RDF Schema, XML, pričom je podporovaný aj import z týchto formátov.

Architektúra Protégé je založená na zásuvných moduloch a tak je veľmi jednoduché rozšíriť jeho funkcionalitu. My sa v ďalšom zameriame na zásuvné moduly ktoré, podporujú vizualizáciu ontológií.

OntoViz

OntoViz predstavuje zásuvný modul do Protégé. Využíva voľne dostupný vizualizačný softvér *GraphViz*[8] (vyvinutý firmou AT&T), ktorý musí byť nainštalovaný ešte pred samotným použitím zásuvného modulu. Ako vizualizačná technika je použitý *graf*, pričom uzly reprezentujú triedy a hrany vlastnosti medzi nimi. Keďže sa jedná o zásuvný modul, *navigáciu* zabezpečuje pohľad

[7] http://protege.stanford.edu/

[8] http://www.graphviz.org/

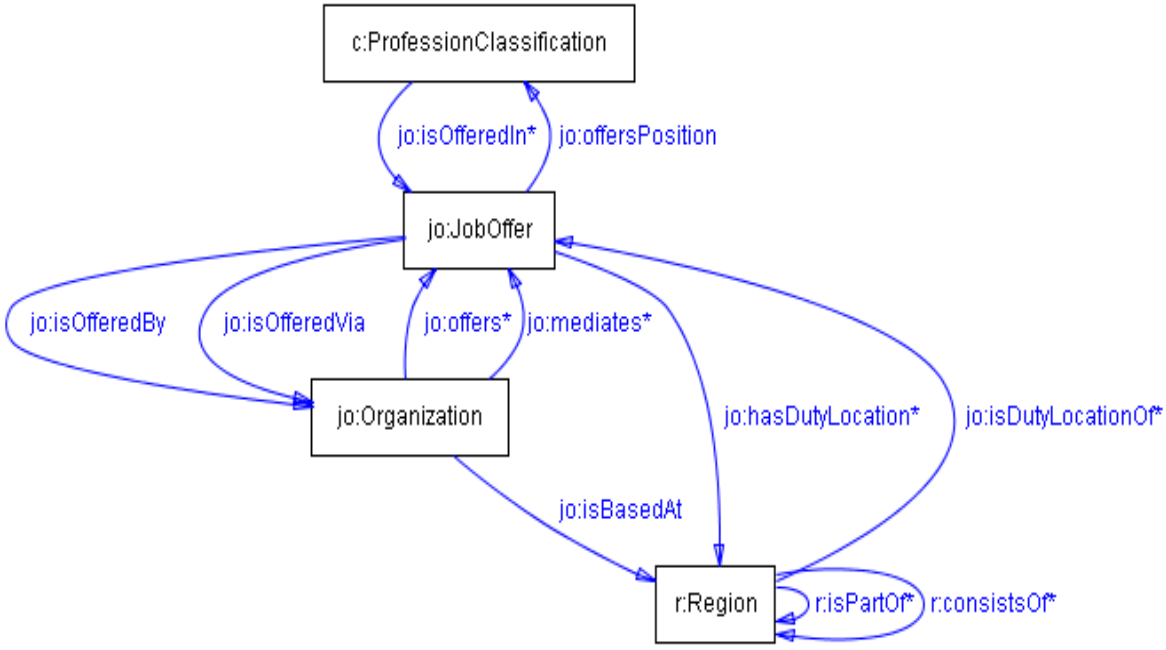
„Class hierarchy“, zobrazujúci v Protégé hierarchiu tried vo forme stromového grafického komponentu. Navigácia je podporená zvýrazňovaním aktuálne zvolenej položky v tomto komponente a jej prislúchajúceho uzla v samotnom grafe. Je možné zvoliť, ktoré triedy sa majú zobrazovať a následne pomocou operátorov uvedených v Tab. 1 dopĺňať graf a vizualizovať tak aj „okolie“ zvolených tried.

Tab. 1 Operátory umožňujúce dopĺňať zobrazovaný graf o ďalšie triedy a vlastnosti

Atribút	Popis
sub (subclass)	zobrazenie všetkých podtried zvolenej triedy
sup (superclass)	zobrazenie všetkých nadtried zvolenej triedy
slx (slot extension)	zobrazenie všetkých takých tried, pre ktoré platí, že existuje vlastnosť medzi týmito triedami a zvolenou triedou, pričom zvolená trieda je z oboru definície tejto vlastnosti
isx (inverse slot extension)	zobrazenie všetkých takých tried, pre ktoré platí, že existuje vlastnosť medzi týmito triedami a zvolenou triedou, pričom zvolená trieda je z oboru hodnôt tejto vlastnosti
slt (slots)	zobrazenie vlastností zvolenej triedy (priamo v uzle prislúchajúcom zvolenej triede)
sle (slot edges)	zobrazenie vlastností zvolenej triedy vo forme orientovaných hrán (len tých, pre ktoré je zvolená trieda z oboru definície a v grafe sa už nachádza trieda z jej oboru hodnôt)
ins (instances)	zobrazenie všetkých inštancií zvolenej triedy
sys (system frames)	zobrazenie systémový tried a vlastností

OntoViz teda umožňuje zadať, ktoré triedy sa majú zobrazovať a následnou aplikáciou operátorov postupne dopĺňať graf, ďalej umožňuje explicitne definovať, ktoré vlastnosti sa majú zobrazovať, prípadne ich rozlíšiť rôznymi farbami. Za veľkú nevýhodu považujeme fakt, že OntoViz neumožňuje filtrovanie tried a len vo veľmi obmedzenej miere umožňuje definovať hĺbku hierarchie. Táto skutočnosť má za následok v mnohých prípadoch veľmi rozsiahly graf. (napr. aplikáciou operátora „sub“ môžeme dostať razom veľmi rozsiahly graf). Ďalšou nevýhodou je, že s vygenerovaným grafom nie je možné následne manipulovať. Pôvodná verzia umožňuje len export grafu do formátu GIF.[9] Príklad vizualizácie ontológie pomocou nástroja OntoViz je uvedený na Obr. 6, pričom je zobrazená časť ontológie pracovných ponúk, vyjadrujúca vzťahy medzi triedami reprezentujúcimi pracovnú ponuku, organizáciu, región a klasifikáciu profesie (pozri prílohu D).

[9] Existuje modifikácia, ktorú urobil Ing. Jaroslav Kuruc z FIIT STU, umožňujúca export grafu do formátu SVG, ktorý je možné následne editovať externe.



Obr. 6 Príklad vizualizácie časti ontológie vytvorenej pomocou zásuvného modulu OntoViz[10]

RDF-Gravity

RDF-Gravity je určený pre vizualizáciu RDF a OWL súborov. Využíva jednak voľne dostupnú knižnicu Jung[11] a taktiež rámec Jena[12], ktorý umožňuje nástroju RDF-Gravity okrem jednoduchej manipulácie s ontológiami zapísanými v jazykoch RDF a OWL, taktiež aj formulovanie vyhľadávacích dopytov. RDF-Gravity využíva vizualizačnú techniku *graf*. Aplikovaním globálnych a lokálnych *filtrov*, umožňuje vizualizovať v danom momente len požadovanú časť ontológie.

Globálne filtre umožňujú z celkovej vizualizácie vyňať nepodstatné, resp. implicitne zrejmé uzly a sprehľadniť tak aktuálny a aj všetky nasledujúce grafy (napr. nezobrazovať vlastnosti `rdfs:label`, `rdf:first`, `rdf:rest` atď.).

Lokálne filtre umožňujú v zásade rovnakú funkcionalitu, avšak umožňujú filtrovať iba predikáty. Avšak následne je možné z aktuálneho grafu vyňať všetky uzly, ktoré nie sú spojené žiadnou hranou s nejakým iným uzlom. Okrem toho poskytuje RDF-Gravity sadu voliteľných filtrov umožňujúcich filtrovať literály, inštancie, anonymné uzly, koncepty, vlastnosti, URI zdroje, a tiež uzly, ktoré nie sú spojené žiadnou hranou s nejakým iným uzlom.

---

[10] Obrázok je prevzatý z prezentácie M. Barlu a M. Tvarožka uvedenej v rámci seminára PeWe. Bližšie pozri http://www.fiit.stuba.sk/research/pewe

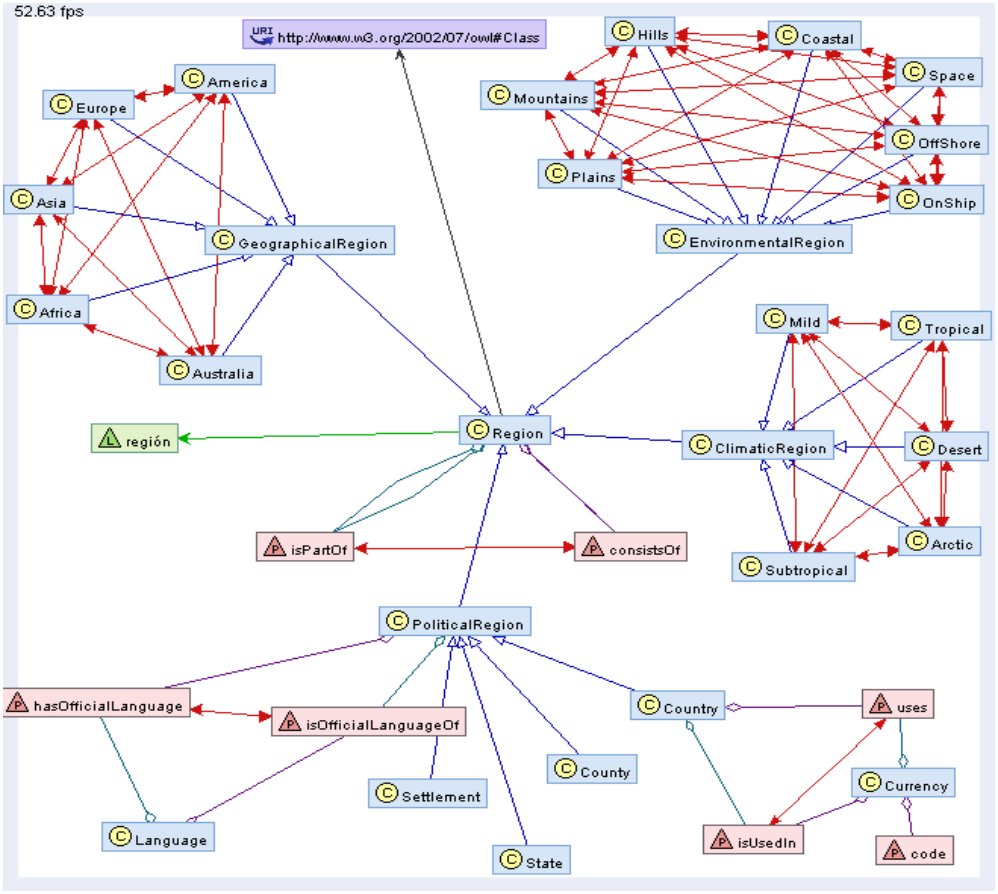
[11] http://jung.sourceforge.net, JUNG (Java Universal Network/Graph Framework) poskytuje prostriedky pre modelovanie, analýzu a vizualizáciu dát vo forme grafov

[12] http://jena.sourceforge.net

RDF-Gravity umožňuje používateľom *prehľadávať* ontológie jednak pomocou full-textového vyhľadávača aplikovaného na zdroje, na vlastnosti alebo na obe súčasne. A taktiež umožňuje používateľovi formulovať dopyty v jazyku RDQL[13]. Výsledky vyhľadávania získané jedným alebo druhým spôsobom, je možné začleniť resp. vyňať z aktuálnej vizualizácie.

Na rozdiel od nástroja OntoViz, umožňuje RDF-Gravity manipuláciu s uzlami a hranami v grafe. Aplikáciou rozmiestňovacieho algoritmu, ktorý sa nazýva *scrambling*, umožňuje RDF-Gravity automaticky rozmiestniť uzly v grafe tak, aby bol tento čo najprehľadnejší.

Vyvolaním kontextového menu príslušného uzla je umožnená používateľovi *navigácia* v grafe. Pre daný uzol umožňuje zobraziť zoznam všetkých vchádzajúcich a vychádzajúcich hrán, pričom zvolením niektorej z nich, sa pridá hrane prislúchajúci uzol do aktuálnej vizualizácie. Treba poznamenať, že navigácia je pre uzly s vysokým stupňom značne neprehľadná. Okrem toho je možné pridať do grafu všetky inštancie príslušného uzla a taktiež zobraziť mu prislúchajúci komentár.



Obr. 7 Príklad vizualizácie časti ontológie vytvorenej pomocou RDF-Gravity

[13] http://www.w3.org/Submission/RDQL

Ďalšou veľmi dôležitou vlastnosťou tohto nástroja je, že umožňuje zahrnúť do vizualizácie ontológie z viacerých súborov. RDF-Gravity neumožňuje export vizualizácie do externého formátu, čo považujeme za jeho veľkú nevýhodu. Príklad vizualizácie vytvorenej pomocou RDF-Gravity je uvedený na Obr. 7, pričom je zobrazená časť ontológie pracovných ponúk, opisujúca triedu Region, členenie regiónu a vzájomné vzťahy.

OntoRama

Nástroj *OntoRama* predstavuje, podľa jeho tvorcov, prototyp všeobecného ontologického prehliadača [7]. Vstupným formátom pre súbory, ktorý dokáže nástroj OntoRama vizualizovať, je RDF/XML, ktoré parsuje pomocou rámca Jena. Koncepty v ontológii sa vizualizujú ako vrcholy a ich vzájomné vlastnosti ako hrany grafu. *OntoRama* vizualizuje koncepty sformované do hierarchie. Keďže niektoré koncepty majú v ontológiách viacero nadtried (viacnásobná dedičnosť), môžu sa vytvárať cykly. OntoRama rieši tento problém tak, že vytvára kópie takýchto konceptov a tak umožňuje zachovať stromovú štruktúru. Kvôli udržaniu prehľadnosti sú všetky takéto koncepty označené červenou farbou.

Špecifikom tohto nástroja je samotná vizualizačná technika, ktorá sa nazýva *hyperbolický strom.* Hyperbolický strom predstavuje strom premietnutý na plochu hyperboly, pričom zmenou jej parametrov sa dosahuje zväčšovanie, resp. zmenšovanie počtu zobrazovaných konceptov (t.j. zoom). Výhodou takého prístupu, oproti predchádzajúcim dvom je fakt, že samotná vizualizácia je v 3D a správnym rozložením vrcholov je ju možné výrazne sprehľadniť. Takýto prístup sa ukáže výhodný hlavne v prípadoch, kedy je nutné vizualizovať veľké množstvo konceptov, ktoré majú veľký počet vzájomných relácií.

Ontológiu je možné pomocou nástroja OntoRama editovať, t.j. pridávať a mazať koncepty, a to jednak v navigačnej časti a taktiež priamo v samotnej vizualizáci. OntoRama umožňuje definovať jednotlivým vlastnostiam špecifický znak a farbu, ktoré slúžia následne ako identifikátor príslušnej vlastnosti vo vizualizáci. Takýto prístup sprehľadňuje vizualizáciu, pretože sa v nej nezobrazuje zbytočne veľa textu. Navyše na vybrané vlastnosti je možné aplikovať *filter* a vyňať ich tak z vizualizácie.

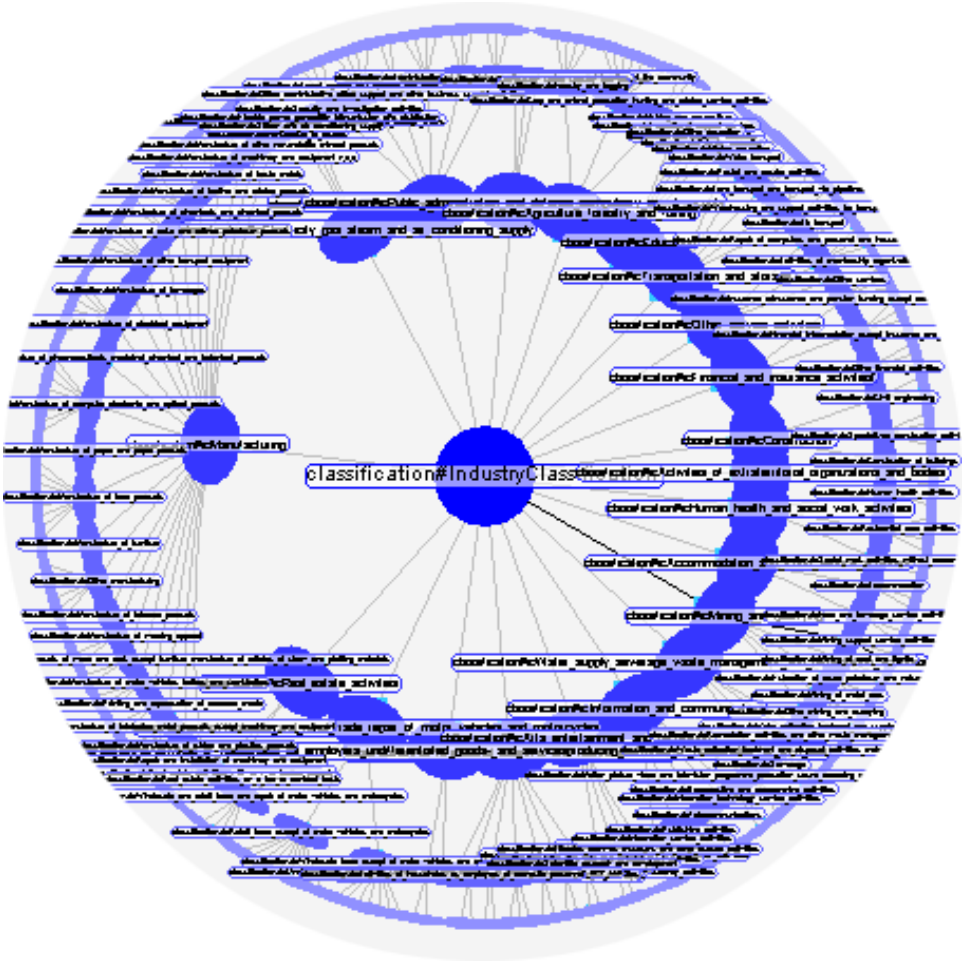
Navigácia v ontológii je zabezpečená stromovým komponentom, umiestneným v pravej časti okna. Navigácia je podporená aj zaradením histórie vykonaných krokov.

Pri rozsiahlych ontológiách s množstvom konceptov, je veľmi vhodné použiť *vyhľadávací mechanizmus*, ktorý OntoRama poskytuje. Avšak ten zďaleka neposkytuje takú funkcionalitu ako v prípade RDF-Gravity.

Nie je podporované ani fulltextové vyhľadávanie, ani formulovanie dopytov v RDQL alebo inom dopytovacom jazyku. Je nutné zadať plne kvalifikované meno hľadaného konceptu. Keďže nie

všetky ontológie formujú jednu jedinú hierarchiu, je vhodné zaradiť aj možnosť zobrazovať viacero stromov. OntoRama takúto možnosť poskytuje.

Príklad vizualizácie ontológie je uvedený na Obr. 8, pričom je zobrazená časť ontológie pracovných ponúk, reprezentujúca klasifikáciu priemyselných odvetví, ktorá je veľmi rozsiahla a mala by sa hodiť práve pre vizualizáciu v OntoRame.



Obr. 8 Príklad vizualizácie časti ontológie vytvorenej pomocou OntoRama

Výsledok vizualizácie je avšak napriek všetkým kladným vlastnostiam tohto nástroja, značne neprehľadný. Je zrejmé, že *OntoRama* sa nehodí na dokumentačné účely, preto nie je prekvapujúce, že neposkytuje ani možnosť exportovať vizualizáciu do nejakého grafického formátu.

Jambalaya

Jambalaya predstavuje zásuvný modul do Protégé. Využíva doménovo nezávislú vizualizačnú techniku SHriMP[14], ktorá umožňuje prehliadať komplexný informačný priestor pomocou viacerých pohľadov [17]. Prednastavený pohľad v Jamabalya zahŕňa dva princípy:

1. ontológia sa vizualizuje ako graf,
2. hierarchické väzby sú vyjadrené pomocou vnorenia, t.j. ide o vizualizačnú techniku *vnorený graf*.

Triedy a inštancie sú zobrazené ako uzly a vlastnosti (okrem tých vyjadrujúcich hierarchiu) ako orientované hrany medzi nimi. *Navigácia* je umožnená jednak pomocou pohľadu „Class hierarchy“, ktorý je súčasťou Protégé, pričom selekcia príslušnej triedy v tomto pohľade, vyvolá animované priblíženie triedy aj v samotnej vizualizáci.

Druhou možnosťou je navigácia priamo vo vnorenom grafe. Pomocou približovania a vzďaľovania (angl. zoom) sa postupne zobrazuje len aktuálna úroveň, t.j. jej vrchol a prislúchajúce priame podvrcholy. Okrem toho je na navigáciu možné využívať kontextové menu prislúchajúce hranám, ktoré umožňuje zobraziť hrane prislúchajúci zdroj resp. cieľ (podľa orientácie hrany). Prehľadná navigácia je taktiež podporená históriou krokov a špeciálnym pohľadom na celú vizualizáciu (angl. thumbnail view).

Zaujímavou vlastnosťou zásuvného modulu Jambalaya je, že kombinuje vizualizáciu s dialógmi Protégé. Formuláre umožňujúce *editovať* triedy a inštancie sú zobrazené v rámci uzla, ktorý predstavuje list stromu. Okrem prednastaveného pohľadu vnorený graf, umožňuje Jambalaya vizualizovať ontológiu aj vo forme klasického stromu. Z vizualizácie ja možné *odfiltrovať* zvolené triedy, inštancie a hrany. Jednotlivé filtre sa zadávajú v dialógoch „Node filter“ a „Arc filter“ veľmi podobným spôsobom ako v prípade RDF-Gravity (zaškrtnutím prislúchajúceho checkboxu). Rozdielom v tomto prípade je, že Jambalaya zoskupuje triedy a vlastnosti do skupín. Výber a aplikácia filtrov je tak omnoho pohodlnejšia.

V prípade uzlov sú definované nasledovné skupiny: definovaná trieda, primitívna trieda, RDFS trieda, enumerácia, logická operácia, reštrikcia a indivíduum. V prípade hrán sú definovaná tieto skupiny: hierarchia tried a inštancií, domain/range, zdedené vlastnosti, logické vlastnosti, vlastnosti indivíduí a reštrikcie na triedach. Okrem toho je možné definovať vlastnú skupinu a pridať do nej žiadané vlastnosti.

K prehľadnosti a lepšej orientácii prispieva aj možnosť zadať pre jednotlivé skupiny tried a vlastnosti rozličné farby a v prípade tried aj rozličné tvary zobrazovaných uzlov. Jambalaya podporuje *fulltextové vyhľadávanie*, avšak momentálne iba v prípade tried, nie je možné vyhľadávať

[14] SHriMP = Simple Hierarchical Multi-Prespective, http://www.thechiselgroup.org/shrimp

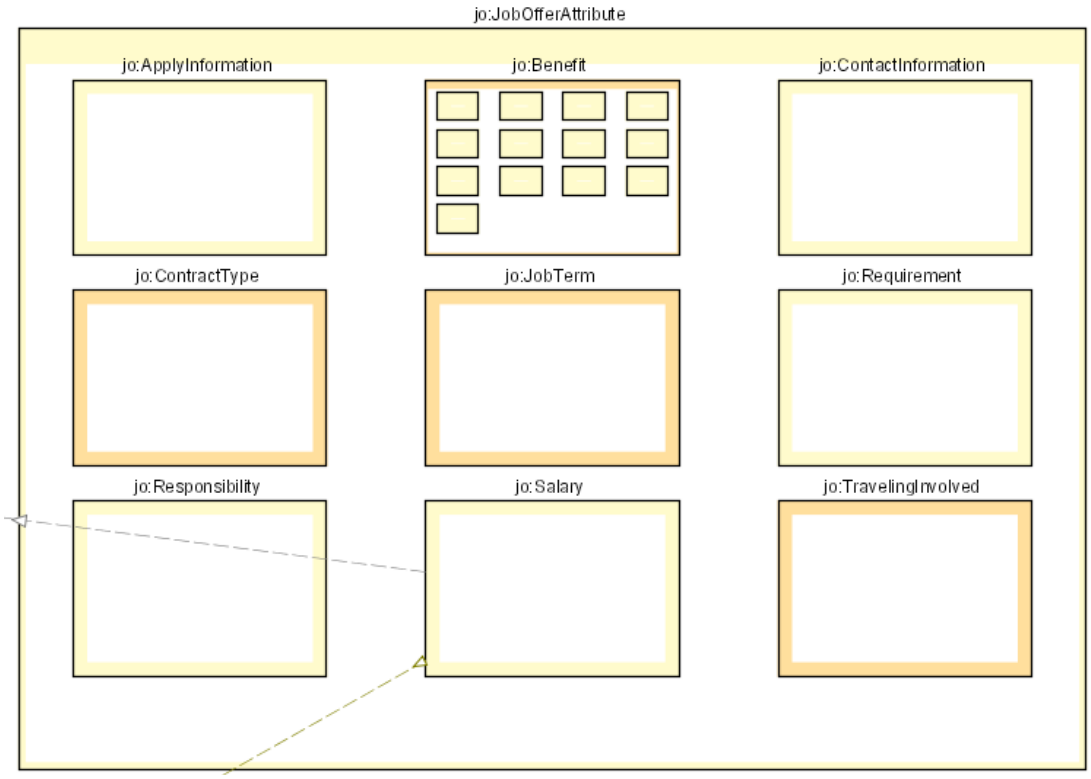
vlastnosti. Je možné aplikovať regulárne výrazy a zúžiť prehľadávanie iba na požadovanú skupinu uzlov.

Rozmiestňovanie uzlov v rámci vnorenia je možné meniť. Jamabalya implementuje rozmiestňovacie algoritmy uvedené v Tab. 2.

Tab. 2 Rozmiestňovacie algoritmy implementované v Jambalaya

Algoritmus	**Opis algoritmu**
mriežka	jednotlivé uzly sú umiestnené na mriežke pričom ich poradie je možné špecifikovať podľa abecedy, typu, počtu potomkov alebo počtu relácií
radiálne rozmiestnenie	uzly sú umiestnené do kruhu
spring rozmiestnenie	sémanticky príbuzné uzly, t.j. také ktoré, sú v spoločnej relácií, sú aj geometricky príbuzné
strom	vertikálne a horizontálne orientovaný strom
mapa	potomkovia vyplnia celú plochu prislúchajúcu rodičovi, pričom veľkosť daného potomka sa určí napríklad podľa počtu jeho priamych potomkov, podľa počtu všetkých jeho nasledovníkov alebo podľa počtu relácií

Keďže Jambalaya poskytuje možnosť *ukladať si aktuálne pohľady* (angl. snapshots), umožňuje tak rýchlo prepínať medzi predmetnými vizualizáciami bez zdĺhavej navigácie. Zaznamenané pohľady ukladajú do postupnosti pohľadov (angl. filmstrip), ktorá sa dá uložiť do súboru.



Obr. 9 Príklad vizualizácie časti ontológie vytvorenej pomocou zásuvného modulu Jambalaya

Príklad vizualizácie ontológie je uvedený na Obr. 9, pričom je zobrazená časť ontológie pracovných ponúk, reprezentujúca atribúty pracovnej ponuky.

Zásuvný modul Jambalaya predstavuje prepracovaný nástroj určený na vizualizáciu znalostí uložených vo forme ontológie. Agreguje viacero veľmi zaujímavých techník a metód, ktoré výrazne podporujú efektívnosť výslednej vizualizácie. Avšak podobne ako OntoRama, ani Jambalaya sa nehodí na dokumentačné účely.

ClusterMap Viewer

Predstavuje techniku vyvinutú firmou Aduna[15], určenú pre vizualizáciu klasifikovaných objektov. Jej hlavným účelom je prehľadne vizualizovať prekrývanie jednotlivých klasifikácií. V prípade ontológií, predstavujú objekty inštancie tried a klasifikácie samotné triedy. K dispozícii je jednak knižnica[16] napísaná v Jave a jednak skúšobná verzia samostatnej aplikácie (*ClusterMap Viewer*), ktorá využíva túto knižnicu.

Navigácia a *filtrovanie* sú v nástroji ClusterMap Viewer, vyriešené stromovým komponentom, ktorý obsahuje hierarchiu tried. Pri každej triede je zároveň vyjadrený počet inštancií a tiež filtrovacie pole, ktorého označením zahrnieme do vizualizácie príslušnú triedu. Navigácia je taktiež podporená históriou krokov. Samotná vizualizácia pozostáva zo zhlukov inštancií a bodov priradených jednotlivým triedam. Zhluky sú prepojené s bodmi práve vtedy, ak obsahujú inštancie patriace do tej triedy, ktorá prislúcha tomuto bodu. Ak je zhluk spojený s viacerými triedami, obsahuje iba tie inštancie, ktoré prislúchajú všetkým týmto triedam súčasne. Z tohto pohľadu predstavuje technika zhlukovania ekvivalent Vennových diagramov. Pridávaním tried do vizualizácie sa zhluky automaticky preskupujú. Používa sa rozmiestňovací algoritmus spring, ktorý minimalizuje počet prelínajúcich sa hrán.

Nevýhodou nástroja ClusterMap Viewer je fakt, že priamo neposkytuje podporu pre zdrojové súbory vo forme RDF/XML alebo OWL. Má vlastný veľmi jednoduchý XML formát vstupného súboru. Avšak poskytuje možnosť vizualizovať Sesame[17] úložisko. Technika zhlukovania má široké použitie, v [2] sa uvádza takéto rozdelenie jej využitia:

Analýza:

- rozdelenie objektov danej klasifikácie, spolu so zobrazením prekrývania
- pohľad na dáta z viacerých uhlov
- porovnávanie rôznych dátových množín v rovnakej doméne

[15] http://aduna.biz

[16] http://aduna.biz/downloads/clustermap/2005.1/clustermap-2005.1.zip

[17] http://www.openrdf.org

- porovnávanie zmien v čase v jednej dátovej množine
- vyhodnocovanie správnosti automatickej klasifikácie

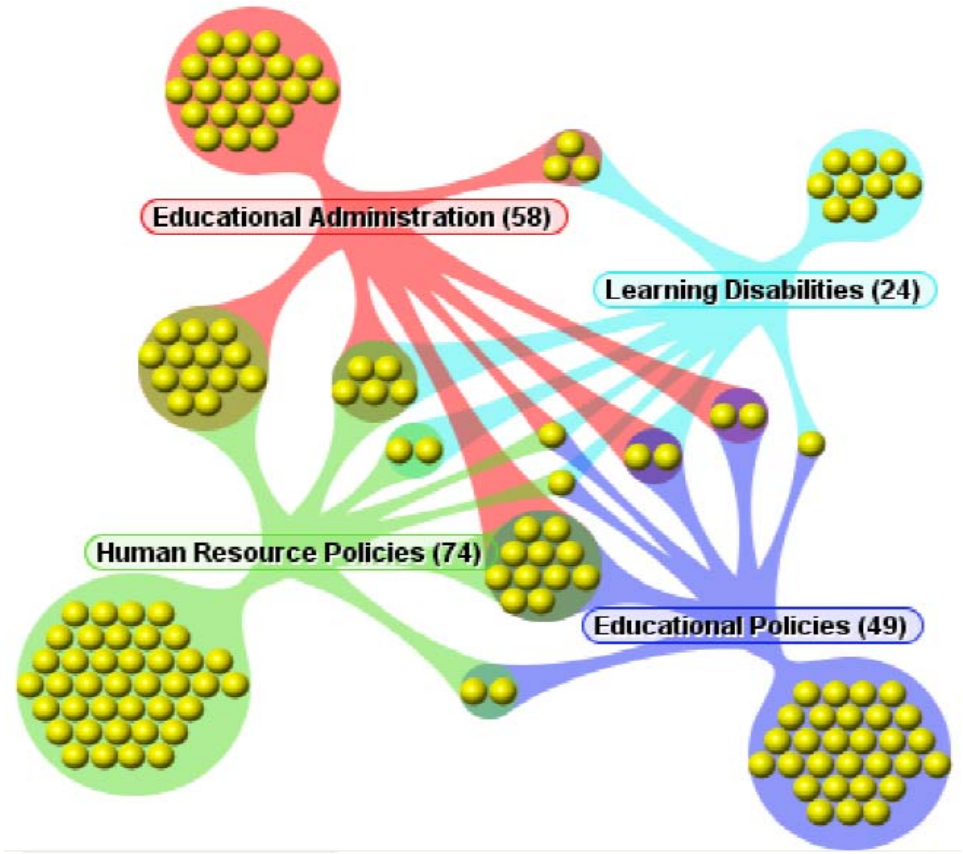
Vyhľadávanie:

- formulovanie dopytov len doplňaním ďalších kľúčových slov
- logické operácie (AND, OR) sú implicitné

Navigácia:

- interaktívna mapa stránky
- sémantický zoom

Na Obr. 10 je zobrazená časť klasifikácie vizualizovanej pomocou ClusterMap Viewer, ktorá nepredstavuje vizualizáciu ontológie.



Obr. 10 Vizualizácia vytvorená pomocou ClusterMap Viewer

Sesame

Sesame[18] je voľne dostupný rámec vytvorený v programovacom jazyku Java pre uchovávanie, dopytovanie a odvodzovanie nad informáciami zapísanými v RDF. Môže byť použitý ako databáza alebo ako knižnica pre ďalšie aplikácie, ktoré potrebujú pracovať interne s ontológiami zapísanými v RDF a RDF Schema [4].

Na rozdiel od predchádzajúcich nástrojov, nástroj Sesame vizualizuje ontológie vo forme textu, Umožňuje zobraziť výroky vo forme trojíc, pričom elementy výroku (subjekt, predikát, objekt) sú zobrazené ako linky. Je teda možné postupne sa v nich preklikávať a prehliadať tak ontológiu. Sesame poskytuje fulltextové vyhľadávanie, ale aj možnosť zadávať dopyty v jazykoch SeRQL-S, SeRQL-C a RDQL.

Na Obr. 11 je znázornený príklad vizualizácie časti ontológie pracovných ponúk pomocou rámca Sesame. Vizualizovaná je trieda JobOffer, pričom vizualizácia je rozdelená do troch tabuliek. Prvá obsahuje výroky, kde trieda JobOffer predstavuje subjekt výroku, druhá predikát a tretia objekt.

Showing statements for: **http://nazou.fiit.stuba.sk/nazou/ontologies/v0.6.16/offer-job#JobOffer**

☑ Use resource labels in overview

Statements with this value as subject:

subject	predicate	object
-	type	Resource
-	type	Class
-	type	http://www.w3.org/2002/07/owl#Class
-	subClassOf	Resource
-	subClassOf	Job offer
-	subClassOf	http://nazou.fiit.stuba.sk/nazou/ontologies/v0.6.16/offer#Offer
-	label	"Job offer"@sk
-	label	"pracovná ponuka"@sk

Statements with this value as predicate:

subject	predicate	object
-- no statements found --		

Statements with this value as object:

subject	predicate	object
je miestom vykonávania práce	range	-
offers	range	-
mediates	range	-
http://nazou.fiit.stuba.sk/nazou/ontologies/v0.6.16/offer-job-inst#S007_lekavy_01068	type	-

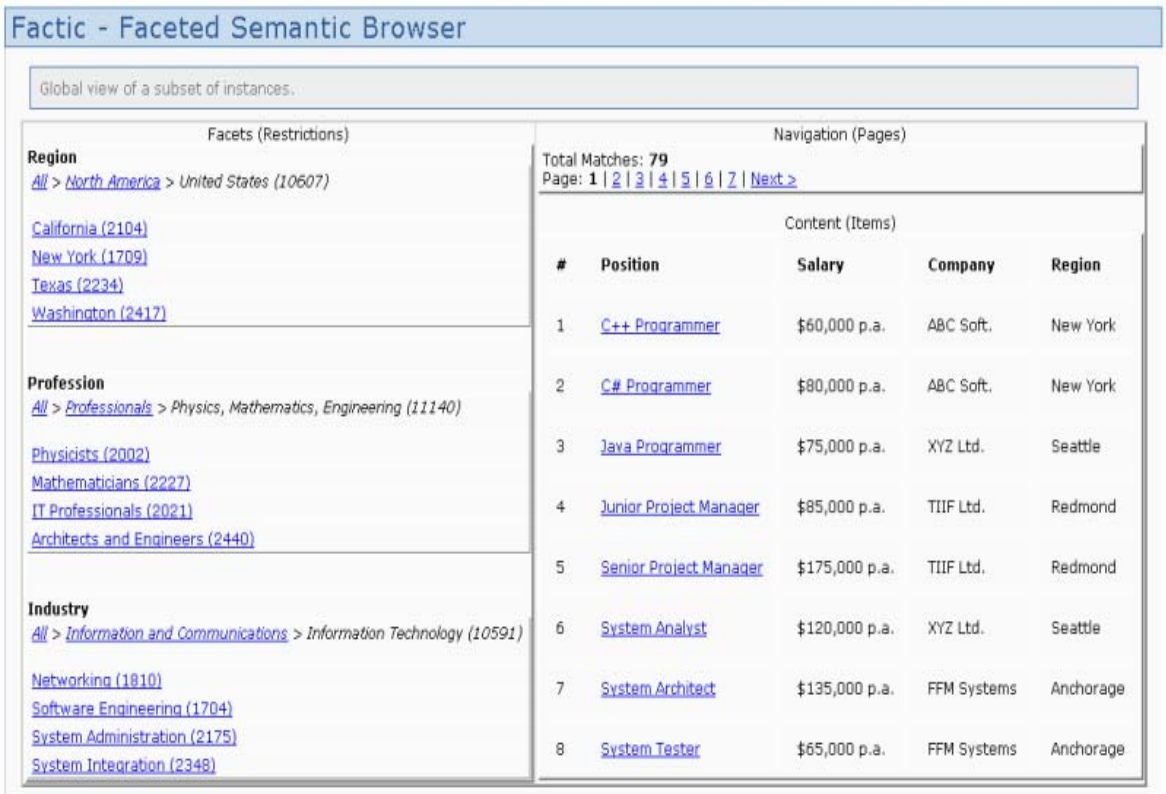
Obr. 11 Vizualizácia vytvorená pomocou rámca Sesame

[18] www.openrdf.com

Factic

Nástroj *Factic* je implementácia fazetového prehliadača, ktorý umožňuje prehliadať rozsiahle informačné priestory. Pri prehliadaní sa využívajú definované fazety a im prislúchajúcich reštrikcie. Fazeta predstavuje všeobecné kritérium, na základe ktorého sa redukuje vizualizovaný priestor. Reštrikcie predstavujú konkrétne hodnoty danej fazety. Factic umožňuje konštrukciu faziet jednak explicitne definovanou kombináciou reštrikcií pomocou logických operátorov AND, OR a NOT alebo implicitným definovaním kritérií ktoré musí daná fazeta spĺňať. Factic je implementovaný v rámci projektu NAZOU[19] a umožňuje prehliadať inštancie pracovných ponúk uložených v ontologickom úložisku. Samotná vizualizáciu má textový charakter. Rozhranie Facticu tvoria nasledovné hlavné časti (pozri Obr. 12):

- zoznam aktuálne aplikovaných faziet,
- zoznam všetkých faziet a im prislúchajúcich reštrikcií,
- zoznam inštancií, ktoré vyhovujú aktuálne aplikovaným fazetám.



Obr. 12 Vizualizácia inštancií ontológie pracovných ponúk pomocou fazetového prehliadača Factic

[19] Nástroje pre Získavanie, Organizovanie a Udržovanie znalostí v prostredí heterogénnych zdrojov, pre podrobnejší opis pozri kapitolu 5.1.2 Požiadavky kladené na vizualizátor

Veľkou výhodou nástroja Factic je adaptívne správanie sa používateľského rozhrania. Prispôsobovanie sa deje v týchto smeroch:

- *Adaptácia faziet*. Zahŕňa ich usporiadanie, uprednostňovanie, filtrovanie, dodatočné anotovanie.
- *Dynamické generovanie faziet*. Vytváranie faziet a im prislúchajúcich reštrikcií na základe metadát uložených v modely používateľa a v doménovom modely.

Adaptácia výsledkov vyhľadávania. Zahŕňa ich usporiadanie, filtrovanie, dodatočné anotovanie.

2.3.3 Porovnanie existujúcich nástrojov

Každý z uvedených nástrojov poskytuje rozdielnu funkcionalitu, aj z pohľadu kvantity aj kvality.

Tab. **3** poskytuje prehľadné porovnanie týchto nástrojov, pričom s výnimkou vizualizačnej techniky, sa na škále od 1 (☆ - nedostatočná úroveň) po 5 (☆☆☆☆☆ - výborná úroveň) vyhodnocovali všetky podstatné poskytované funkcie. Znak '–' znamená, že daná funkcia nie je nástrojom vôbec podporovaná.

Tab. 3 Porovnanie vybraných vizualizačných nástrojov

Nástroj / Vlastnosť	OntoViz	RDF Gravity	OntoRama	Jambalaya	Cluster Map	Sesame
Vizualizačná technika	graf	graf	hyperbo-lický strom	vnorený graf	graf zhlukov	text
Navigácia	☆☆	☆	☆☆☆	☆☆☆☆☆	☆☆	☆☆☆☆
Filtrovanie	☆☆	☆☆☆	☆☆	☆☆☆	☆☆	☆☆
Vyhľadávanie	–	☆☆☆☆	☆☆	☆☆☆	–	☆☆☆☆☆
Rozmiestňovanie	☆☆	☆☆☆	☆☆☆☆	☆☆☆☆☆	☆☆☆☆	–
Zobrazenie detailov	☆	☆	☆☆☆	☆☆☆☆	☆	☆☆
Vstupný súbor	☆☆☆	☆☆☆☆	☆☆☆	☆☆☆	☆☆	☆☆☆
Export vizualizácie	☆☆☆	–	–	–	☆☆☆☆	–
Dokumentácia	☆☆☆	☆☆☆	☆	☆	☆☆☆	–
Interaktívnosť	☆	☆☆☆☆	☆☆☆	☆☆☆☆	☆☆☆	☆☆☆☆
Editácia	–	–	☆☆	☆☆☆☆	–	–
Súčet	***17 x*** ☆	***23 x*** ☆	***23 x*** ☆	***32 x*** ☆	***21 x*** ☆	***20x*** ☆

Vyhodnotením Tab. 3 dostávame, že najlepšie v porovnávaní uspel zásuvný modul Jambalaya, ktorý predstavuje prepracovaný nástroj, poskytujúci bohatú funkcionalitu a je zvlášť vhodný na prehliadanie a interaktívnu prácu s ontológiami.

2.4 Zhodnotenie súčasného stavu vo vizualizácii ontológií

V predchádzajúcej časti sme opísali vybrané existujúce nástroje určené na vizualizáciu ontológií. Každý jeden z nich využíva na vizualizáciu inú techniku alebo metódu. Na základe vyhodnotenia analyzovaných nástrojov môžme formulovať nasledujúce závery:

- jednou z najpoužívanejších vizualizačných techník je technika graf,
- vizualizačné nástroje, ktoré umožňovali interaktívne zasahovať do vizualizácie patrili medzi najlepšie vyhodnotené,
- filtrovanie zobrazovaných prvkov výrazne ovplyvňuje kvalitu vizualizácie,
- najčastejšie sa vyskytovali rozmiestňovacie algoritmy strom a radiálne rozmiestnenie,
- vyhľadávanie na základe kľúčových slov je vhodné doplniť aj vyhľadávaním na základe komplexnejších dopytov vo zvolenom jazyku,
- vizualizačná technika ovplyvňuje možnosť exportu vizualizácie do externého grafického formátu,
- vyššie hodnotenie mali nástroje, ktoré zobrazovali v danom momente iba určitú časť celého priestoru a následne navigačnými technikami umožnili prehliadať postupne aj ostatné časti,
- nástroje neposkytujú podporu pre používateľov, ktorí nie sú dostatočne znalí vo vizualizovanom informačnom priestore,
- z hľadiska lepšej orientácie v priestore vynikajú nástroje, ktoré vo poskytujú viacero pohľadov na vizualizáciu súčasne.

Napriek relatívne kladným výsledkom hodnotenia, nie je možné prehlásiť, že najlepšie ohodnotený nástroj Jambalaya je ideálny a nepotrebuje už žiadne ďalšie modifikácie alebo vylepšenia. Napríklad z hľadiska používateľského komfortu je oveľa jednoduchšie použiť webové rozhranie ontologického úložiska Sesame alebo vytvoriť dokumentačné výstupy ako v prípade RDF-Gravity. Najzávažnejší problém avšak tkvie v princípe ako sú dáta prezentované používateľovi. V žiadnom z uvedených nástrojov nie je možnosť odporučiť používateľovi časť vizualizácie, ktorou by mal v prehliadaní začať resp. ktorú si nemusí vôbec všímať pretože nie je dostatočne dôležitá. Všetky z uvedených nástrojov vychádzajú pri vizualizácii iba zo syntaxe jazyka, v ktorom sú metadáta reprezentované. Napríklad spomínaný nástroj Jambalaya využíva reláciu „is a“ a preto je na začiatku vizualizácie používateľovi prezentovaná najvšeobecnejšia trieda Thing a jej priamych potomkov.

Predstavme si ale vizualizátor, ktorý by prezentoval používateľovi časti informačného priestoru na základe sémantiky jednotlivých entít, ktoré ho tvoria. Napríklad chceme prehliadať informačný priestor predstavujúci model určitej domény, vizualizátor v ňom uprednostňuje pri prezentácii tie entity tohto modelu, pre ktoré sú v danom kontexte odhadne najvyššia miera vhodnosti na základe ich významu alebo skutočnosti ako prehliadali ontológiu iní používatelia. Ďalej vizualizátor odporučí používateľovi nasledujúci krok, ktorým sa vo vizualizácii posunie ďalej, pričom stále berie do úvahy nielen syntax ale aj uvedenú mieru vhodnosti príslušnej entity a jej okolia.

3. Ciele práce

S ohľadom na výsledky analýzy a sformulované závery definujeme ciele tejto práce. Prvoradým cieľom je navrhnúť vizualizačnú metódu, ktorá spĺňa tieto kritéria:

1. *Inkrementálne vizualizovanie* – vizualizovať v každom momentne iba určitú časť celého priestoru s cieľom poskytnúť používateľovi dostatočnú úroveň detailov o zobrazovaných entitách vizualizovaného priestoru,
2. *Komplementárne pohľady* – poskytnúť viacero pohľadov na vizualizovaný priestor,
3. *Podpora navigácie* – podporovať používateľa pri navigácii v priestore ontológie, pričom sa zohľadnia nielen syntaktické špecifiká, ale v čo najväčšej miere aj sémantika vizualizovaných entít.

Následne, s cieľom overiť navrhnutú metódu, vytvoríme prototyp vizualizačného softvéru, ktorý spĺňa nasledovné kritéria:

1. *Podporované formáty* – snahou je čo najväčšia nezávislosť od zdrojového formátu, v ktorom je ontológia zapísaná, ale minimálne je nevyhnutné podporovať v súčasnosti najrozšírenejšie formáty RDF, RDFS a OWL,
2. *Cieľový používateľ* – nástroj sa zmeriava na typ používateľa ontológie, ktorý sa chce oboznámiť s jej štruktúrou a nie editovať ju alebo prehliadať jej inštancie,
3. *Intuitívne používateľské rozhranie* – rozhranie vizualizátora by malo poskytovať cieľovému používateľovi dostatočný komfort.

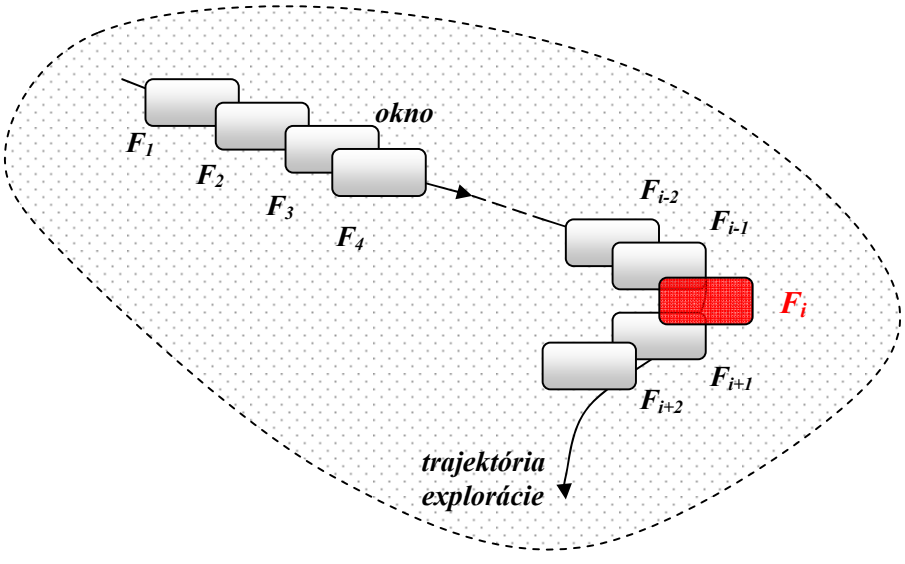
4. Inkrementálne vizuálne prehľadávanie s ohodnocovaním na základe významu

Kapitola obsahuje návrh metódy vizualizácie ontológií, ktorá využíva techniku inkrementálneho prehľadávania priestoru a dopĺňa ju o možnosť stanoviť každej vizualizovanej entite aj kvantitatívne ohodnotenie, na základe ktorého je možné prispôsobovať vizualizáciu používateľom tak, aby sa v nej dokázali jednoduchšie orientovať a navigovať.

4.1 Princíp inkrementálneho prehľadávania

Efektívna vizualizácia rozsiahlych informačných priestorov si vyžaduje prepracovaný systém interaktívnej navigácie. V prípade vizualizácie rozsiahlych grafov priama aplikácia rozmiestňovacích algoritmov neposkytuje potrebné výsledky, resp. úplne zlyháva. Preto sa vyvinuli viaceré techniky, ktoré sa snažia tento problém prekonať [19]. Od všeobecne známych techník ako sú približovanie a vzďaľovanie (angl. zoom), cez techniky, ktoré sa snažia pri zobrazení detailov zachovať aj informáciu o aktuálnom kontexte (angl. fisheye) a techniky zhlukovania významovo podobných informácií (angl. clustering), až po inkrementálne prehľadávania vizualizovaného priestoru (angl. incremental browsing).

Princíp inkrementálneho prehľadávania spočíva v zobrazení malej časti rozsiahleho grafu, pričom ostatné časti sú zobrazené až v prípade potreby. Vizualizačný systém zobrazí časť grafu, tzv. okno, prehľadávanie potom znamená posun okna po zvolenej trajektórii (pozri Obr. 13).



Obr. 13 Princíp inkrementálneho prehliadania rozsiahlych grafov [10]

Nech e_j^i je *j*-ta entita okna F_i, nasledovné okno F_{i+1} sa vygeneruje pridaním susedných entít entity e_j^i a v prípade potreby aj odobratím entít pôvodného okna F_i na základe zvoleného algoritmu A tak aby,

vizualizácia spĺňala stanovené kritéria, formálne $F_{i+1} = \varepsilon\left(e_j^i\right) \cup F_i \setminus f_i(\mathrm{A})$, pričom F_i predstavuje pôvodné okno, F_{i+1} nasledujúce okno, $\varepsilon\left(e_j^i\right)$ okolie entity e_j z *i*-teho okna a $f_i(\mathrm{A})$ množinu entít *i*-teho okna skonštruovanú na základe algoritmu *A*. Trajektória prehliadania potom predstavuje zoznam entít, ktorých okolie sa v danom okne vizualizovalo.

Dôležitými aspektmi techniky inkrementálneho prehľadávania sú:

1. stratégia generovania nových okien, t.j. voľba trajektórie,
2. spôsob vizualizácie pôvodných dát po vytvorení nového okna.

Voľba trajektórie

Voľba trajektórie je obyčajne plne v réžii používateľa, ktorý sa po preskúmaní aktuálnej vizualizácie rozhodne, ktorým smerom sa ďalej uberať.

Vizualizácia pôvodných dát

V princípe existujú dva spôsoby ako vizualizovať pôvodné dáta:

a. Do vizualizácie dopĺňame nové dáta bez toho, aby sme modifikovali pôvodné. Po určitom čase avšak narazíme na dva problémy. Prvým je, že vizualizácia prestane byť prehľadná v dôsledku nadmerného množstva vizualizovanej informácie. Tento problém sa dá čiastočne riešiť aplikáciou vhodného rozmiestňovacieho algoritmu pri každej zmene okna [13]. Druhým problémom je obmedzená veľkosť okna, ktorá je priamo úmerná rozlíšeniu a veľkosti displeja. Z tohto dôvodu nie je možné a ani vhodné neustále pridávať dáta do vizualizácie.

b. Odstránenie časti informácie z pôvodnej vizualizácie. V tomto prípade je potrebné rozhodnúť, ktoré dáta z vizualizácie odstrániť. Existuje viacero algoritmov, ktoré túto úlohu riešia. Okrem toho je vhodné zabezpečiť históriu predchádzajúcich vizualizácií v prípade, ak by sa chcel používateľ vrátiť k niektorej z nej.

Takýto prístup k vizualizácii rozsiahlych informačných priestorov prináša viacero výhod:

- vizualizácia obsahuje iba obmedzené množstvo informácie požadovanej v danom okamihu,
- zobrazované informácie majú dostatočnú úroveň detailov,
- vysoký stupeň interaktívnosti vizualizácie.

Okrem spomínaných výhod má však táto vizualizačná technika aj svoje nevýhody:

- vizualizácia neposkytuje globálny pohľad na zobrazované dáta, aj keď existujú metódy ako sa vysporiadať aj s týmto problémom napr. kombináciou globálneho a lokálneho pohľadu na vizualizovaný priestor [14]
- pre efektívnu vizualizáciu je potrebná znalosť informačného priestoru.

4.2 Aplikácia inkrementálneho prehľadávania pri vizualizácii ontológií

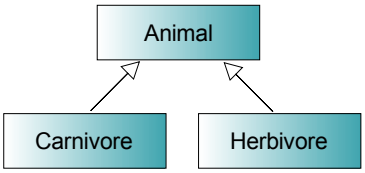
Princíp inkrementálneho prehľadávania sa dá aplikovať aj pri vizualizácii ontológií. Ontológie je možné vizualizovať ako graf, ktorý už v prípade menších ontológií predstavuje pomerne rozsiahly informačný priestor. Použitie tejto techniky je preto veľmi vhodné. V nasledujúcom najskôr objasníme spôsob inkrementálneho prehľadávania ontológie založený na postupnom rozvíjaní uzlov, reprezentujúcich triedy v ontológii a zhodnotíme jeho výhody a nevýhody.

Princíp inkrementálneho prehľadávania objasníme na ontológii, ktorá stručne opisuje svet africkej divočiny [1]. Základ ontológie predstavujú triedy *Animal* a *Plant*. Vizualizáciu budeme vytvárať od triedy Animal:

Krok 1: Vizualizácia triedy Animal

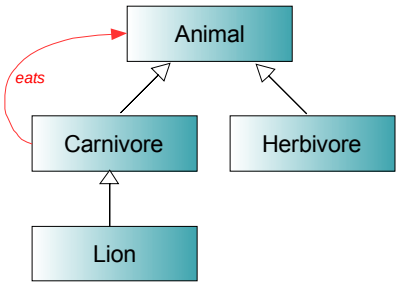
Predstavuje štartovací bod vizualizácie. Zobrazí triedu Animal, pričom všetky vzťahy na ostané triedy (vlastnosti typu ObjectProperties) sú zatiaľ schované.

Krok 2: Vizualizácia okolia triedy Animal



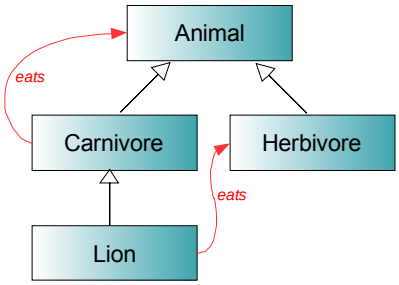
Rozvinutím triedy Animal, získame triedy *Carnivore* a *Herbivore*, ktoré predstavujú špecializácie triedy Animal.

Krok 3: Vizualizácia okolia triedy Carnivore



Rozvinutím triedy Carnivore získame triedu *Lion*, ktorá reprezentuje konkrétneho predstaviteľa triedy mäsožravcov a tiež vzťah mäsožravcov a ostatných zvierat (vlastnosť *eats*).

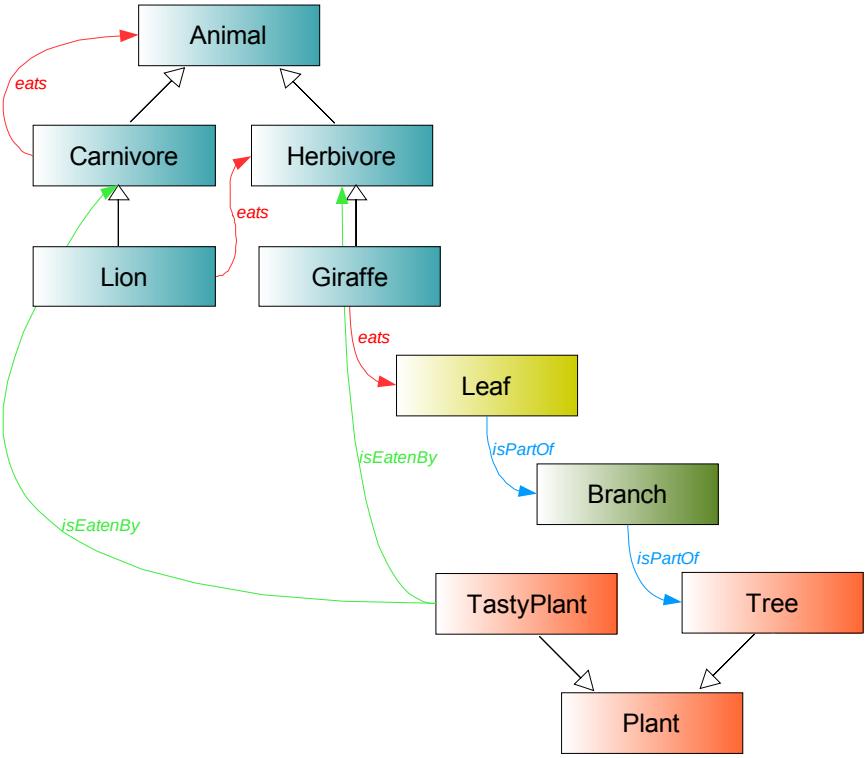
Kroky 4: Vizualizácie okolia triedy Lion



Rozvinutím triedy Lion už nezískame ďalšiu špecializáciu, iba vzťah triedy Lion a triedy Herbivore.

Krok 5-10: Ďalej postupujeme rovnakým spôsobom pričom postupne rozvíjame triedy, ktorých okolie chceme vizualizovať.

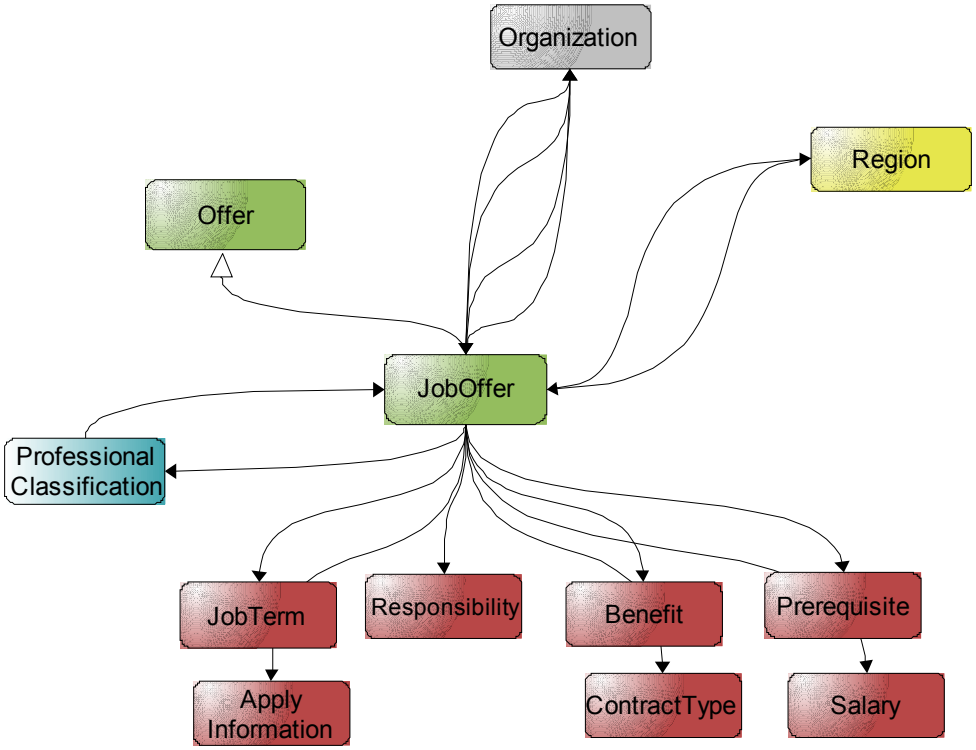
Krok 11: Výsledok vizualizácie ontológie aplikáciou inkrementálneho prehľadávania.



Takýmto spôsobom postupne prejdeme cez všetky triedy ontológie, pričom v každom okamihu máme prehľad o predchádzajúcej vizualizovanej časti. V tomto prípade nie je nutné aplikovať redukciu zobrazovanej informácie, pretože sa jedná o pomerne jednoduchý informačný priestor. Avšak v prípade komplexnejších ontológií môže nastať situácia, kedy rozvinutie danej triedy spôsobí explóziu ďalších tried (pozri Obr. 14). Tento problém je možné riešiť aplikáciou redukcie predchádzajúcej vizualizovanej informácie v kombinácii s aplikáciou rozličných filtrov, na základe ktorých je možné redukovať aj novo vizualizovanú informáciu. Cieľom filtrov je zobraziť v danom kontexte len významovo najdôležitejšie triedy. Samotný filter je možné realizovať využitím

ohodnocovacieho modulu, ktorý v danom kontexte určí dôležitosť tried a zobrazí iba tie najvýznamnejšie z nich. Okrem toho je možné preddefinovať filtre, ktoré umožnia filtrovať napríklad:

- *inštancie triedy* – vhodné najmä v situáciách, kedy nás pri vizualizácii inštancie tried nezaujímajú alebo využívame efektívnejší nástroj na ich zobrazenie,
- *potomkov alebo predkov* – často krát je výhodnejšie vizualizovať hierarchiu samostatne,
- *dátové typy* – informáciu o dátových typoch je vhodnejšie zahrnúť do iného pohľadu (napr. tabuľka vyvolaná z kontextového menu triedy),
- *triedy z danej ontológie* – vhodné v prípade, že chceme vizualizovať iba triedy prislúchajúce danej ontológii.



Obr. 14 Príklad rozvinutia triedy, ktorá ma väčší počet nasledovníkov

4.3 Ohodnocovanie

Stanovenie miery dôležitosti danej triedy, t.j. kvantitatívne ohodnotenie jej relevancie, umožní používateľovi lepšie sa orientovať v ontológii. Na základe tohto ohodnotenia, je možné zostaviť usporiadaný zoznam tried nasledovníkov, priamočiaro aplikovať filtre, zvýrazniť dôležitejšie triedy a tým nasmerovať používateľa vo vizualizácii. Efektívny ohodnocovací algoritmus predstavuje prostriedok, ktorým možno výrazne prispieť k jednoduchej a priamočiarej navigácii v ontológiách. Môžeme rozlíšiť viaceré techniky a metódy ohodnocovania.

4.3.1 Techniky stanovenia miery dôležitosti

A. Globálne stanovenie dôležitosti:

- v tomto prípade je cieľom stanovenie miery dôležitosti každej triedy, prípadne vlastnosti v rámci celej ontológie,
- proces sa vykoná v rámci predspracovania ontológie, t.j. po fáze načítania ontológie, čo môže spôsobiť pomerne dlhú inicializáciu vizualizácie,
- v kombinácii so statickým ohodnocovaním (pozri nižšie) je možné takýmto spôsobom stanoviť iniciálne hodnoty.

B. Kontextom riadené stanovenie dôležitosti:

- v tomto prípade sa ohodnocovací modul aplikuje až v prípade potreby a navyše sa ohodnotia iba triedy v danom okolí,
- postupnou aplikáciou ohodnocovacieho modulu na danú množinu tried sa po čase ohodnotí celá ontológia,
- keďže sa v danom momente ohodnocuje iba obmedzená množina tried, nebude zrejme negatívne ovplyvnená ani odozva systému na požiadavky používateľov.

4.3.2 Metódy ohodnocovania

A. Statické ohodnocovanie

- vytvorené na základe analýzy štruktúry ontológie, pričom je možné sledovať napríklad nasledovné charakteristiky: počet potomkov a nasledovníkov, počet inštancií, typ uzla alebo hrany, stupeň uzla.

Tab. 4 Výsledky statického ohodnotenia vybraných tried ontológie pracovných ponúk

Trieda / Charakter.	JobOffer	Region	Organization	Classification
Potomkovia	-	25	-	10
Predkovia	1	-	-	-
Predchodci	3	2	1	-
Nasledovníky	12	1	3	-
Vlastnosti dát. typu	4	-	-	-
Vlastnosti obj. typu	13	3	3	-
Inštancie	101	372	85	897
Stupeň uzla	132	411	107	1590

- nie je triviálne vyhodnotiť takto získané dáta. Existuje viacero možných protichodných záverov, ktoré možno zaujať po vyhodnotení týchto dát. Napríklad pre dáta uvedené v Tab. 4 možno tvrdiť:
  - najvýznamnejšia je trieda *Classification* pretože má najväčší stupeň, avšak viac ako polovica z tohto množstva je kvôli veľkému počtu inštancií, okrem toho stupeň uzla ovplyvňujú aj opisy a komentáre priradené danej triede, pričom ich počet môže byť ľubovoľný,
  - najvýznamnejšia je trieda *Region*, pretože predstavuje vrchol rozsiahlej hierarchie,
  - najvýznamnejšia je trieda JobOffer, pretože má definovaných najviac vlastností objektového a dátového typu,
- je zrejmé že statické ohodnocovanie nie je možné využiť ako nosnú metódu vyhodnotenia dôležitosti tried, avšak môže slúžiť ako pomocná metóda, na základe ktorej je možné určiť iniciálne hodnoty, resp. rozhodovať v prípade rovnakých ohodnotení získaných inou metódou.

B. Dynamické ohodnocovanie

- v tomto prípade sa miera dôležitosti určuje dynamicky na základe externých podnetov
- ohodnotenia sú vytvorené a modifikované na základe interakcie s používateľom, resp. s aplikáciami:
  a. *sledovanie používateľov vizualizátora* – počas toho ako používateľ prehľadáva ontológiu sú zaznamenávané jeho preferencie pri výbere tried z aktuálneho zoznamu a následne sa na pozadí modifikujú hodnoty priradené jednotlivých triedam.
  b. *sledovanie aplikácií, ktoré vizualizovanú ontológiu využívajú* – v tomto prípade sa sleduje prístup externých aplikácií do úložiska, v ktorom sú ontológie uložené. Cieľom záujmu sú informácie o tom, ktoré entity ontológie sú predmetom požiadaviek, aká akcia sa požaduje (čítanie, modifikácia), a ktorá aplikácia požiadavku poslala. Vyhodnotením získaných dát je možné modifikovať ohodnotenia jednotlivých tried uvedených v ohodnocovacej ontológii.
- dynamické ohodnocovanie kladie zvýšené nároky na infraštruktúru systému, ktorá implementuje navrhovanú metódy,
- je nutné zabezpečiť efektívny spôsob ako získavať a vyhodnocovať externé podnety, ideálnym riešením je využitie externého logovacieho nástroja, ktorý uchováva dáta v štruktúrovanej forme.

4.4 Formálne vyjadrenie navrhovanej metódy

Nasledovný pseudokód obsahuje formálne vyjadrenie navrhovanej metódy:

$$\text{EXPLORE:} \quad \begin{aligned} & e^i \leftarrow getSelectedEntity(F_i); \\ & \varepsilon_{e^i} \leftarrow getVicinity(DS, e^i, GP); \\ & F_{i+1} \leftarrow F_i \cup \varepsilon_{e^i}; \\ & F_{i+1} \leftarrow applyFilters(F_{i+1}, FS); \end{aligned}$$

4.4.1 Opis algoritmu

Krok1: *Voľba pivota*

- používateľ si zvolí v aktuálnom okne entitu, pre ktorú sa bude generovať okolie, tzv. pivot (e^i)

Krok2: *Generovanie okolia pivota*

- funkcia *getVicinity* vygeneruje okolie aktuálneho pivota, pričom parametrami metódy sú ontologické úložisko (*DS*), zvolená entita (e^i), parametre generovania okolia (*GP*)

Krok3: *Vytvorenie nového okna*

- zjednotenie prvkov pôvodného okna a vygenerovaného okolia

Krok 4: *Aplikovanie filtrov*

- funkcia *applyFilters* aplikuje na nové okno sadu definovaných filtrov

Inicializačný mód

Pre inicializáciu metódy sa používa mierne modifikovaný algoritmus:

$$\text{INIT:} \quad \begin{aligned} & e^0 \leftarrow getMostSignificant\,\mathrm{Re}\,source(EDS); \\ & \varepsilon_{e^0} \leftarrow getVicinity(DS, e^0, GP); \\ & F_1 \leftarrow applyFilters(F_1, FS); \end{aligned}$$

Krok 1: *Voľba pivota*

- funkcia *getMostSignificantResource* zvolí za pivota entitu s najvyšším ohodnotením v úložisku, ktoré obsahuje ohodnotenia pre jednotlivé entity vizualizovaného priestoru (*EDS*)

Krok 2: *Generovanie okolia*

- vygenerovanie okolia rovnakým postupom ako v predchádzajúcom prípade

Krok 3: A*plikácia filtrov*

- aplikácia filtrov rovnako ako v predchádzajúcom prípade

4.5 Výhody a nevýhody navrhovanej metódy

Navrhovaná metóda vizualizácie ontológií ma dve významné výhody:

- *dostatočná úroveň detailov* – vzhľadom na to, že sa v jednom okamihu nevizualizuje celý informačný priestor, je možné obohatiť vizualizáciu o doplňujúce informácie o vizualizovaných entitách ontológie,
- *podpora používateľa pri navigácii* – aplikáciou významového ohodnocovania vizualizovaných entít je možné automatizovať proces tvorby trajektórie, t.j. je možné efektívne podporiť používateľa pri navigácií. Týmto je možné eliminovať problém, kedy používateľ, ktorý nepozná vizualizovanú ontológiu(e), „zablúdi“ do oblastí, ktoré nie sú v danom momente dostatočné dôležité.

Hlavnou nevýhodou navrhnutej metódy je:

- *strata globálneho prehľadu* – principiálne je veľmi ťažké zachovať v jednej vizualizácii globálny prehľad pri súčasnom zobrazení detailov jednotlivých vizualizovaných entít. Tento problém je avšak možné riešiť zavedením dodatočnej vizualizácie, ktorá zobrazuje v každom globálnu pozíciu, ktorá zodpovedá aktuálnemu stavu okna v rámci hlavnej vizualizácie.

5. Návrh vizualizátora ontológií

V tejto kapitole opíšeme návrh vizualizačného nástroja, určeného na vizualizáciu štruktúry ontológií, ktorý využíva navrhnutú metódu inkrementálneho vizuálneho prehľadávania. V úvode kapitoly definujeme ohraničenia a špecifikujeme požiadavky kladené na navrhovaný nástroj. Jadro kapitoly tvorí návrh architektúry vizualizačného systému, opis jeho modulov a stručný opis základných prvkov používateľského rozhrania.

5.1 Ohraničenia a požiadavky kladené na vizualizátor

5.1.1 Ohraničenia

Na základe analýzy sme stanovili nasledovné ohraničenia na vytváraný vizualizačný nástroj:

1. **Cieľový používateľ** - návrhár, resp. programátor aplikácií, ktoré využívajú technológie webu so sémantikou, ontológie a dáta štruktúrované vo forme súborov RDF a OWL. Pričom je zdôraznená potreba získať rýchlo prehľad o štruktúre dát.
2. **Vizualizácia štruktúry ontológie** - navrhovaný vizualizačný systém je určený na vizualizáciu štruktúry dát (tried a vlastností ontológie) a nie na zobrazenie konkrétnych inštancií dátových štruktúr.
3. **Obmedzenie rozsahu zobrazovaných dát** - používateľovi sa prezentujú iba jednotlivé časti vizualizovaného priestoru a aplikáciou navrhnutej metódy sa môže postupne presúvať z jednej časti na inú.
4. **Uprednostňovanie a filtrovanie dát** - dátové štruktúry majú v rámci celého priestoru rôznu úroveň dôležitosti, snahou našej metódy vizualizácie je, zvýrazniť dôležité časti a naopak potlačiť vo vizualizáci menej významné časti. Cieľom je pomôcť používateľovi efektívne sa navigovať v celom priestore. Na ohodnotenie dôležitosti sú navrhnuté viaceré heuristiky.
5. **Použitie vizualizátora** - navrhnutý vizualizačný nástroj je možné využívať iba použitím webového prehliadača pričom samotné spracovanie a prístup k dátam sa deje na strane servera.

5.1.2 Požiadavky kladené na vizualizátor

Nefunkcionálne požiadavky a kontext systému

Navrhovaný vizualizačný nástroj sa plánuje zakomponovať medzi aplikácie aktuálne existujúce a naďalej vyvíjané v rámci projektu NAZOU[20] (Nástroje pre Získavanie, Organizovanie a Udržovanie znalostí v prostredí heterogénnych zdrojov), ktorého cieľom je rozvoj metód a vývoj nových nástrojov podporujúcich získavanie, organizovanie a prezentovanie informácií a znalostí obsiahnutých v rôznorodých informačných priestoroch. Typickým príkladom takéhoto informačného priestoru je World Wide Web obsahujúcich málo štruktúrované, resp. neštruktúrované dáta, informácie a znalosti. Projekt NAZOU sa sústreďuje na doménu pracovných ponúk. Pri testovaní a experimentálnom overovaní navrhovaného systému sa využívajú ontológie, ktoré túto doménu opisujú (pozri prílohu D).

Z kontextu projektu NAZOU vyplývajú aj požiadavky na architektúru a použité technológie. Vizualizačný systém je implementovaný v jazyku Java a nasadený v rámci prezentačného rámca Cocoon[21].

Funkcionálne požiadavky

V nasledovnej časti sú uvedené základné prípady použitia vizualizačného systému. V prevažnej miere sa jedná o prípady použitia na najvyššej úrovni abstrakcie, ktoré zahŕňajú viaceré konkrétne prípady použitia. Pre každý prípad použitia sú definované nasledovné atribúty:

Názov	názov prípadu použitia v plnej forme
Identifikátor	kód prípadu použitia
Priorita	definuje dôležitosť daného prípadu použitia. Rozlišujú sa 3 úrovne (1=najvyššia, 2=stredná, 3=najnižšia)
Frekvencia	Definuje početnosť výskytu pri práci so systémom. Rozlišujú sa 3 úrovne (veľmi často, často, zriedka)
Opis	charakteristika prípadu použitia
Zahrnuté prípady použitia	zoznam prípadov použitia na nižšej úrovni abstrakcie, ktoré patria opisovanému prípadu použitia

[20] http://nazou.fiit.stuba.sk

[21] http://cocoon.apache.org

Názov:	*Prehliadanie štruktúry ontológie*
Identifikátor:	UC01
Frekvencia:	veľmi často
Priorita:	1 = najvyššia
Opis:	Systém musí umožniť prehliadanie štruktúry ontológie, pričom je požadovaná dostatočná úroveň detailov pri zachovaní kontextovej informácie. Z toho vyplýva, že vizualizácie bude musieť poskytovať viacero pohľadov na zobrazované dáta.
Zahrnuté prípady použitia:	a. Zobrazenie detailu triedy b. Zobrazenie zoznamu priamych nasledovníkov a predchodcov c. Zobrazenie zoznamu potomkom a predkov

Názov:	*Usporiadanie a filtrovanie vizualizovaných informácií*
Identifikátor:	UC02
Frekvencia:	veľmi často
Priorita:	1 = najvyššia
Opis:	Používateľovi sa dáta prezentujú v usporiadanej podobe, kde dôležitejšie časti ontológie sú uprednostňované a menej dôležité potláčané resp. filtrované. Táto požiadavka si vyžaduje predspracovanie prezentovaných dát.
Zahrnuté prípady použitia:	a. Zostupne a vzostupné usporiadanie zoznamov predchodcov, nasledovníkov, potomkov a predkov podľa mena b. Zostupne a vzostupné usporiadanie zoznamov predchodcov, nasledovníkov, potomkov a predkov podľa ohodnotenia c. Filtrovanie uzlov v grafickom prehliadači

Názov:	*Voľba vizualizovanej množiny dát*
Identifikátor:	UC03
Frekvencia:	často
Priorita:	1 = najvyššia
Opis:	Systém musí umožniť zvoliť množinu, resp. viacero množín dát, nad ktorou sa vizualizácia vykonáva, t.j. poskytnúť používateľovi možnosť zvoliť si množinu ontológií, ktoré chce vizualizovať.
Zahrnuté prípady použitia:	a. Voľba úložiska b. Voľba ontológií

Názov:	*Správa ohodnotení*
Identifikátor:	UC04
Frekvencia:	veľmi často
Priorita:	1 = najvyššia
Opis:	Vizualizátor potrebuje k pri realizovaní ostatných funkčných prvkov získať informáciu o ohodnoteniach prislúchajúcich k daným entitám vizualizovaného priestoru. Navrhovaný systém poskytuje možnosť získať ohodnotenie pre danú triedu a taktiež spravovať ohodnotenia.
Zahrnuté prípady použitia:	a. Prístup k ohodnoteniu prislúchajúcemu k danej triede b. Pridanie, modifikácia, zrušenie ohodnotení c. Najvyššie ohodnotená entita d. Prepočítanie ohodnotení

Názov:	*Akceptované formáty*
Identifikátor:	UC05
Frekvencia:	veľmi často
Priorita:	1 = najvyššia
Opis:	Vizualizačný systém pracuje s ontológiami transparentne bez ohľadu na to v akej notácií sú zaznamenané. Keďže uvedenú požiadavku je v princípe veľmi ťažko dosiahnuť, vzhľadom na množstvo formátov zápisu, musí navrhovaný vizualizačný nástroj podporovať minimálne prácu s ontológiami zaznamenanými v jazykoch RDF, RDF Schema a OWL.
Zahrnuté prípady použitia:	a. Načítanie ontológie do úložiska

Názov:	*Vyhľadávanie*
Identifikátor:	UC06
Frekvencia:	často
Priorita:	2 = stredná
Opis:	Vizualizačný systém nesmie predpokladať, že používateľ pozná štruktúru ontológie. Prezentovanie usporiadaných a filtrovaných dát umožní efektívnejšiu prácu s ontológiami, avšak je dôležité ponúknuť používateľovi aj možnosť vyhľadávať v ontológii, minimálne na základe kľúčových slov.
Zahrnuté prípady použitia:	a. Vyhľadávanie v množine názvov tried a popiskov

Názov:	*Interaktívnosť vizualizácie*
Identifikátor:	UC07
Frekvencia:	často
Priorita:	2 = stredná
Opis:	Vizualizačný nástroj umožňuje používateľovi zasahovať do vizualizácie (premiestňovať elementy vizualizácie, modifikovať množstvo zobrazovanej informácie, meniť nastavenia, či aplikovať rôzne rozmiestňovacie algoritmy).
Zahrnuté prípady použitia:	a. Zmena rozmiestnenia uzlov v grafickom prehliadači b. Stanovenie polomeru okolia v grafickom prehliadači c. Stanovenie maximálneho počtu uzlov v grafickom prehliadači

Názov:	*Export vizualizácie*
Identifikátor:	UC08
Frekvencia:	občas
Priorita:	3 = najnižšia
Opis:	Umožniť používateľovi vizualizačného nástroja exportovať vizualizáciu do externého formátu. A to jednak grafického (jpeg, png, gif), ale aj textového (XML, SVG, GraphML).
Zahrnuté prípady použitia:	-

Ilustračné scenáre použitia

Scenár A: *Export okolia najvýznamnejšej triedy spomedzi všetkých vizualizovaných ontológií*

1. Používateľ si zvolí úložisko, v ktorom sú ontológie uložené.
2. Systém prehľadá úložisko a vygeneruje zoznam, ktorý obsahuje všetky ontológie, ktoré sa v úložisku nachádzajú.
3. Používateľ si z vygenerovaného zoznamu vyberie podmnožinu ontológií, v rámci ktorých chce zobraziť okolie najvýznamnejšej triedy.
4. Systém prehľadá ohodnocovacie úložisko a získa zoznam najvyššie ohodnotených tried.
5. Používateľ si zo zoznamu najvýznamnejších tried vyberie tú, ktorú chce vizualizovať. Ak existuje systém automaticky prejde na bod 6.
6. Systém vygeneruje textovú reprezentáciu okna, ktoré prislúcha danej triede, pričom okno obsahuje zoznam predchodcov, nasledovníkov, potomkov, predkov a detail triedy.
7. Používateľ iniciuje zobrazenie grafu v grafickom prehliadači.
8. Systém vygeneruje reprezentáciu okna vo formáte, ktorý sa používa pre vizualizáciu (GraphML).
9. Používateľ aplikuje dostupné rozmiestňovacie algoritmy prípadne filtre.

10. Systém pozmení rozloženie prvkov vizualizácie na v rámci okna.
11. Používateľ iniciuje export vizualizácie do externého formátu.
12. Systém vygeneruje požadovaný súbor a umožní používateľovi uložiť ho na lokálny disk.

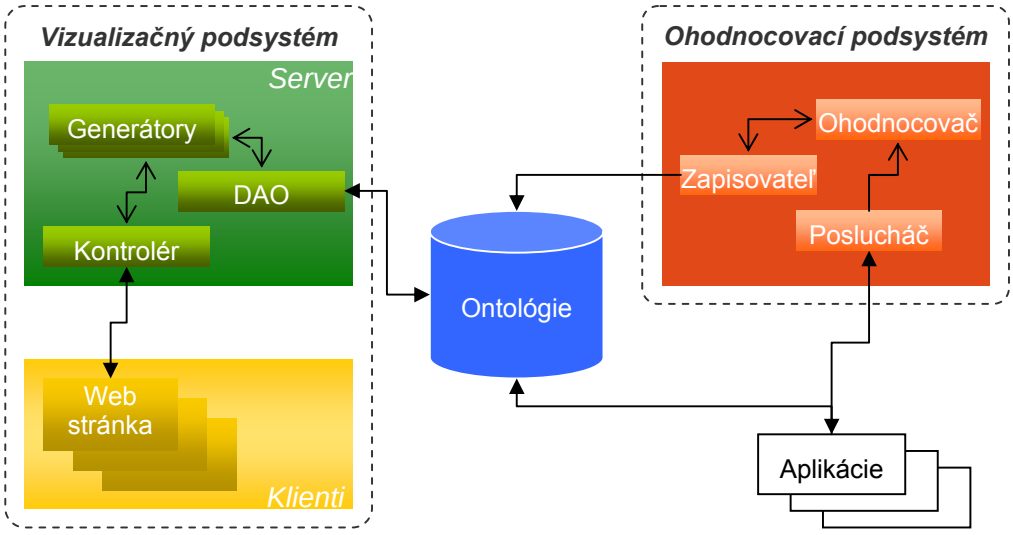
Scenár B: *Zobrazenie detailu triedy XYZ*

1. Používateľ si zvolí úložisko, v ktorom sú ontológie uložené.
2. Systém zaregistruje zvolené úložisko.
3. Používateľ iniciuje zobrazenie vyhľadávacieho mechanizmu.
4. Systém vygeneruje príslušný formulár a zoznam ontológií, ktoré sa budú prehľadávať.
5. Používateľ zadá kľúčové slovo na základe ktorého chce vyhľadávať a zvolí požadovanú podmnožinu ontológií.
6. Systém vygeneruje formálny dopyt do úložiska pričom berie do úvahy zvolené ontológie.
7. Systém odošle dopytu do úložiska a z odpovede vygeneruje zoznam tried, ktoré vyhovujú zadaným kritériám.
8. Používateľ si zvolí želanú triedu.
9. Systém vygeneruje textovú reprezentáciu okna.

5.2 Architektúra systému

Navrhovaný vizualizačný systém pozostáva z dvoch nezávislých podsystémov, ktoré kooperujú nad dátovým úložiskom (pozri Obr. 15):

- vizualizačný podsystém,
- ohodnocovací podsystém.



Obr. 15 Schéma architektúry navrhovaného vizualizátora

5.2.1 Ohodnocovací podsystém

Úlohou ohodnocovacieho podsystému, je sledovať použitie ontológie aplikáciami a zaznamenávať informácie potrebné na určenie miery dôležitosti danej triedy či vlastnosti v ontológii. V našom návrhu napojíme ohodnocovací podsystém na úložisko (Sesame), pričom v tomto prípade je úlohou ohodnocovacieho modulu zaznamenať aktivitu v úložisku, t.j. ktorá aplikácia, ktorá ontológia bola použitá, ktoré triedy, resp. vlastnosti boli v danom dopyte pristupované. Takto získané dáta sa následne vyhodnotia a informácie získané z tohto procesu sa perzistentne uložia. Sú dve možnosti ako zrealizovať perzistenciu získaných dát:

a. vytvoriť samostatnú databázu alebo ontológiu, ktorá dané dáta bude uchovávať,
b. zaznamenať výsledok vyhodnocovacieho procesu priamo do príslušnej ontológie (napr. pridať ku každej triede alebo vlastnosti ďalšie metadáta).

Oba spôsoby majú svoje výhody aj nevýhody. V prípade samostatnej ontológie, dosiahneme síce väčšiu úroveň nezávislosti, ale na druhej strane treba riešiť problém synchronizácie nekonzistencií vzniknutých, ak sa pôvodná ontológia zmení. V druhom prípade sú požadované dáta uložené v pôvodnej ontológii, čo si vyžaduje zásah do ontológie, ktorý v niektorých prípadoch nie je žiaduci alebo dokonca možný. Nekonzistencie vzniknuté zmenou pôvodnej ontológie zahŕňajú prípady modifikácie, pridania alebo zmazania triedy alebo vlastností z ontológie. Tieto prípady je nutné pri práci s ontológiou identifikovať a automaticky aplikovať zmeny aj do ohodnocovacej ontológie alebo pravidelne kontrolovať konzistenciu pomocou nástrojov určených na porovnávanie ontológií a manuálne modifikovať ohodnocovaciu ontológiu.

Ohodnocovacia ontológia

V našom návrhu sme uprednostnili prvú možnosť, pretože nechceme zasahovať do existujúcej ontológie. Na uchovávanie metadát je navrhnutá ohodnocovacia ontológia, ktorá definuje vlastnosti uvedené v Tab. 5.

Tab. 5 Definícia vlastností ohodnocovacej ontológie

Názov vlastnosti	*Opis vlastnosti*	*Definičný obor*	*Obor hodnôt*
hasAccessCount	Vlastnosť priraďuje danej triede hodnotu počítadla prístupov	rdfs:Resource	xsd:int
hasEvaluation	Vlastnosť priraďuje ľubovoľnej triede bližšie nedefinované reťazcové ohodnotenie	rdfs:Resource	xsd:string
lastAccessedBy	Vlastnosť priraďuje danej triede identifikátor naposledy pristupujúcej aplikácie	rdfs:Resource	xsd:string

Keďže sme definovali jednotlivé vlastnosti nad doménou všetkých zdrojov, je možné ohodnocovať ľubovoľný zdroj, t.j. triedu, vlastnosť aj inštanciu.

Moduly ohodnocovacieho podsystému:

1. poslucháč
2. ohodnocovač
3. zapisovateľ

Poslucháč:

Pôvodný návrh počítal s napojením sa na ontologické úložisko Sesame, s cieľom zachytávať vykonávane dopyty. Uvedený spôsob sa ukázal ako nevhodný, pretože jediný spôsob akým by sa dal daný poslucháč realizovať je odpočúvanie HTTP alebo RMI požiadaviek posielaných na Sesame server. Keďže takto získané dáta je potrebné netriviálne parsovať, zhodnotili sme daný spôsob ako nevyhovujúci.

V rámci projektu NAZOU je implementovaný softvérový nástroj *SemanticLog*, ktorý predstavuje logovací nástroj umožňujúci zaznamenať jednak udalosti generované na strane serveru a jednak aktivitu používateľov webových aplikácií, t.j. na strane klienta. Nástroj *SemanticLog* je implementovaný ako webová služba a umožňuje jednoducho centralizovať požadované dáta do spoločného logu. Nad týmto logom je možné vykonávať rozličné analýzy a extrakcie informácií o aktivitách používateľov sledovaných aplikácií a adekvátne prispôsobovať ich používateľský model [3]. V našom návrhu počítame s využitím nástroja SemanticLog pri sledovaní aktivít týkajúcich sa prístupu do úložiska, t.j. prístupy k jednotlivým triedam použitej ontológie. Získané dáta sa následne ukladajú v ohodnocovacej ontológii.

Ohodnocovač:

Metadáta obsiahnuté v ohodnocovacej ontológii sú vyhodnotené, pričom sa sleduje počet prístupov, resp. dopytov jednotlivých aplikácií na danú triedu v ontológii. Takáto jednoduchá heuristika umožňuje stanoviť ich poradie a zvýrazniť používateľom tie, ku ktorým sa pristupuje častejšie, predpokladajúc, že majú v rámci ontológie významnejšie postavenie.

Zapisovateľ:

Zaznamenáva získané dáta do navrhnutej ontológie. Pričom tieto budú slúžiť pri jej prezentácii.

5.2.2 Vizualizačný podsystém

Úlohou tohto podsystému je vizualizovať zvolenú ontológiu používateľovi, pričom pri jej prehliadaní sa využijú metadáta uložené v ohodnocovacej ontológii. Cieľom je umožniť používateľovi orientovať sa v ontológii v každom okamihu, pričom chceme poskytnúť jednak dostatočnú úroveň detailov a na

druhej strane zachovať celkový prehľad. Vizualizačný podsystém je navrhnutý ako webová aplikácia, umožňujúca prehliadanie v dvoch módoch:

a. *textový mód* tvoria 3 pohľady:
   - pohľad *Trieda* – zobrazuje detailné informácie o triede (pozri Tab. 6),
   - pohľad *Predchodci* – zobrazuje zoznam tried, ktoré sú s danou triedou v incidencii, pričom daná trieda je prvkom oboru hodnôt všetkých vlastností, ktorých definičný obor obsahuje triedu zo zoznamu predchodcov:

     zoznam_predchodcov= $\{\forall t \in T \bullet \forall v \in V : t_{daná} \in range(v) \wedge t \in domain(v)\}$

Tab. 6 Atribúty pohľadu Trieda

Atribút	Popis
Názov	názov triedy (určený atribútom rdf:ID)
Komentáre	zoznam komentárov k triede (definovaných v elemente rdfs:comment)
Opisy	zoznam opisov k triede (definovaných v elemente rdfs:label)
Vlastnosti dátového typu	zoznam vlastností dátového typu, ktorých definičný obor zahŕňa danú triedu
Vlastnosti objektového typu	zoznam vlastností objektového typu, ktorých definičný obor zahŕňa danú triedu
Predkovia	zoznam predkov danej triedy (určený elementom rdfs:subClassOf)
Potomkovia	zoznam potomkov danej triedy
Inštancie	zoznam inštancií danej triedy

   - pohľad *Nasledovníky* – zobrazuje zoznam tried, ktoré sú s danou triedou v incidencii, pričom daná trieda je prvkom definičného oboru vlastností, ktorých obor hodnôt obsahuje triedu zo zoznamu nasledovníkov:

     zoznam_nasledovníkov= $\{\forall t \in T \bullet \forall v \in V : t_{daná} \in domain(v) \wedge t \in range(v)\}$

b. *grafický mód*:
   - obsahuje vizualizačný applet, ktorý vizualizuje okolie zvolenej triedy
   - okolie zvolenej triedy predstavuje jej nasledovníkov a predchodcov, spolu s vyznačením typu vlastnosti
   - na jednotlivé triedy je možné kliknúť a presunúť vizualizáciu o jeden krok ďalej, t.j. zobraziť okolie zvolenej triedy

Moduly vizualizačného podsystému:

Vizualizačný podsystém je navrhnutý ako webová aplikácia, kde na strane servera je cieľom získať a na základe ohodnocovacej ontológie adekvátne usporiadať dáta, tieto následne preposlať na stranu klienta, kde sa vhodne zobrazia.

Vizualizačný server:

Úlohou vizualizačného servera je zaznamenať požiadavky klienta, spracovať ich, pristúpiť do úložiska získať potrebné dáta. Následne pre ne získať ohodnotenia zaznamenané v ohodnocovacej ontológii, aplikovať ich a vygenerovať požadovaný výstup.

Vizualizačný server tvoria tieto komponenty:

1. *Kontrolér* – zaznamenáva požiadavky klienta, riadi tok spracovania výberom ostatných komponentov a aplikuje ohodnocovaciu informáciu.
2. *Prístupové objekty* – realizujú prístup do úložiska, odtieňujú detaily aplikačného rozhrania úložiska.
3. *Sada generátorov* – generujú jednotlivé časti textového alebo grafického výstupu vizualizačného serveru.

Vizualizačný klient:

Predstavuje webový prehliadač pre textový režim a v*izualizačný applet*, ktorý vizualizuje požadované dáta vo forme grafu.

5.3 Návrh používateľského rozhrania

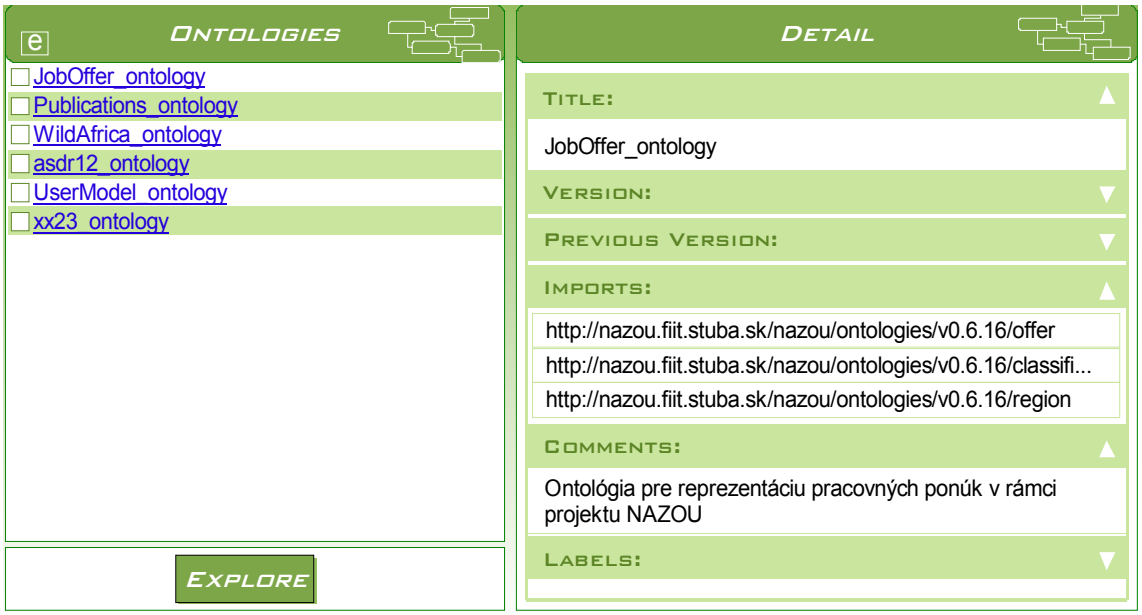
Používateľské rozhranie navrhovaného systému tvoria 3 základne obrazovky:

1. Zoznam ontológií spolu s ich detailom
2. Prehliadač tried - textový režim
3. Prehliadač tried - grafický režim

V tejto podkapitole si opíšeme obsah uvedených základných obrazoviek, pričom podrobný opis všetkých častí používateľského rozhrania je uvedený v prílohe B.

5.3.1 Zoznam ontológií

Po pripojení na úložisko sa zobrazí pohľad „Zoznam ontológií“ (angl. *Ontology list*), ktorý zobrazí všetky ontológie, ktoré sa v ňom nachádzajú (pozri Obr. 16). Po kliknutí na element zoznamu v okne „Ontológie“ (angl. *Ontologies*), t.j. na zvolenú ontológiu, sa v pravej časti, v okne „Detail“ zobrazia jej detailné informácie. Stlačením tlačidla „Prehliadať“ (angl. *Explore*) je možné začať vizualizáciu tried, pričom do vizualizácie sa zahrnú iba zvolené ontológie. Pre každú ontológiu je možné definovať prefix, ktorý sa bude používať v ďalšej vizualizácii.



Obr. 16 Zoznam ontológií spolu s detailnými informáciami o danej ontológii

5.3.2 Prehliadač tried

Pri prvej aktivácii prehliadača tried, je spomedzi zvolených ontológií zvolená trieda s najväčšou mierou dôležitosti, táto predstavuje vstupný bod vizualizácie. Máme na výber zvoliť si mód prehliadania. Rozlišujú sa dva módy, textový (prednastavený) a grafický.

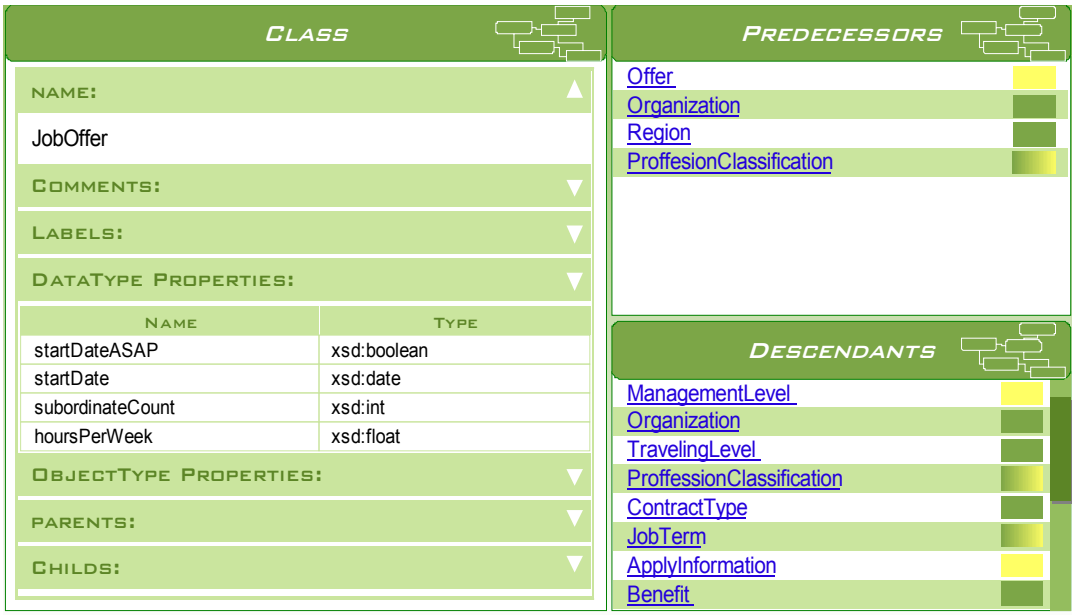
Textový mód

Okno v textovom móde obsahuje tri pohľady (pozri Obr. 18):

1. Pohľad Trieda (angl. *Class*) zobrazuje detail triedy
2. Pohľad Predchodci (angl. *Predecessors*) zobrazuje zoznam priamych predchodcov triedy
3. Pohľad Nasledovníky (angl. *Descendants*) zobrazuje zoznam priamych nasledovníkov triedy.

Elementy zoznamov predchodcov aj nasledovníkov predstavujú možné alternatívy trajektórie prehliadania a všetky sú ohodnotené. Ohodnotenie je vyjadrené farebnou škálou, ktorá definuje 5 stupňov dôležitosti, úplne zelený obdĺžnik vyjadruje triedu s najvyšším a úplne šedý triedu s najnižším ohodnotením (pozri Obr. 17).

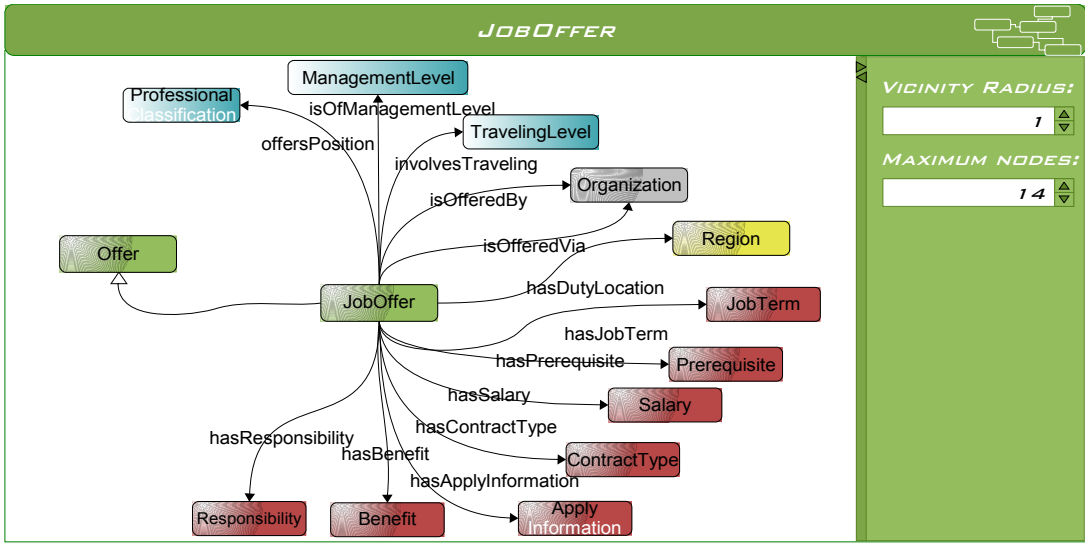
Obr. 17 Farbená škála vyjadrujúca mieru dôležitosti tried



Obr. 18 Textový mód prehliadača tried

Grafický mód

V prípade zobrazenia grafu, záložka Graf (angl. *Graph*), sa vizualizuje okolie aktuálnej triedy (pozri Obr. 19), pričom v pravej časti je možné zobraziť konfiguračný panel a definovať polomer okolia, a maximálny počet tried, ktoré sa v okne môžu vyskytovať. Hrany medzi triedami predstavujú objektové vlastnosti vizualizovanej triedy prípadne vzťah *is-a*, ohodnotenie je vyjadrené pozadím uzla, ktorý ma farbu podľa definovanej škály. Jednotlivé uzly sú „klikateľné“, je teda možné postupne prehliadať okolie zvolených tried. Pre zobrazenie detailu triedy sa stačí prepnúť do textového módu.



Obr. 19 Grafický mód prehliadača tried

6. Overenie návrhu - prototyp vizualizačného systému

Nasledujúca kapitola obsahuje opis prototypu vytvoreného na základe návrhu. V úvode kapitoly sú opísane ciele prototypovania, nasleduje opis implementovaných modulov a záver kapitoly sa venuje zhodnoteniu prototypu a predkladá závery, ktoré z jeho implementácie vyplynuli.

6.1 Ciele prototypovania

V tejto etape riešenia projektu sme si stanovili vytvoriť funkčnú implementáciu navrhnutej vizualizačnej metódy, pričom cieľom je vyhodnotiť jej vhodnosť pri vizualizácii rozsiahlych priestorov metadát. V našom prípade tento priestor predstavuje ontológia pracovných ponúk a jej podrobnejší opis možno nájsť v prílohe D.

Rozhodli sme sa prototypovať tieto časti navrhovaného vizualizačného systému

1. Vizualizačný podsystém – textový a grafický mód
2. Ohodnocovací modul – získavanie, zápis a vyhodnotenie dát.

6.2 Implementácia

Prototyp bol implementovaný v jazyku Java s použitím nasledovných vývojových prostriedkov:

- Eclipse 3.2 – IDE pre vývoj aplikácií v jazyku Java
- Ant 1.6.5 – zostavovací nástroj

Pre skompilovanie, nasadenie a prevádzkovanie prototypu je nutné mať k dispozícii nasledujúcu sadu programových komponentov:

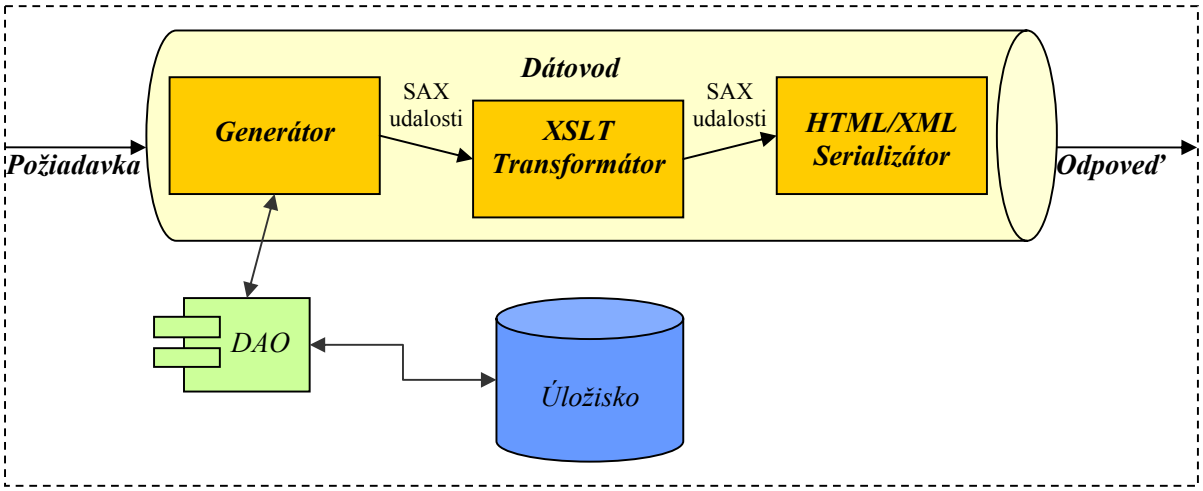
- Apache Tomcat 5.5.20 – servletový kontajner,
- Apache Axis 1.4 – web servisový aplikačný rámec
- Apache Cocoon 2.1 – prezentačný rámec založený na architektúre dátovodov,
- Sesame 1.2.5 – ontologické úložisko,

6.2.1 Vizualizačný podsystém

Vizualizačný podsystém je implementovaný ako webová aplikácia, t.j. klient prostredníctvom webového prehliadača posiela požiadavky na server, ktorý zabezpečuje získanie, spracovanie a transformáciu dát do formátu, ktorý je akceptovaný klientom, t.j. HTML stránka. Keďže obsah stránok závisí od požiadavky používateľa, je nutné obsah stránok generovať dynamicky. Na generovanie stránok sa využíva prezentačný server Cocoon, ktorý slúži ako kontrolér, ktorý deleguje spracovanie požiadaviek na prislúchajúce komponenty. V prípade serveru Cocoon sú to tzv. dátovody. Každý dátovod predstavuje reťazec zo sekvenčným spracovaním, v ktorom sa na začiatku

generujú požadované dáta, tieto sa v ďalšom kroku ľubovoľný počet krát transformujú a nakoniec serializujú do požadovaného výstupu (pozri Obr. 20). V našom prípade rozlišujeme dva druhy výstupov:

1. HTML výstup – používa sa pre textové prehliadanie,
2. XML výstup – používa sa pre grafické prehliadanie.



Obr. 20 Schéma generovania dynamických HTML stránok

Vizualizačný podsystém tvoria tieto dva moduly:

Vizualizačný server

Vizualizačný server tvoria tieto hlavné komponenty:

1. *Sada generátorov*. Generátory slúžia na generovanie odpovedí na klientske požiadavky, pričom každá odpoveď ma formu XML súboru. V Tab. 7 sú uvedené implementované generátorov spolu s ich stručným popisom.

Tab. 7 Množina generátorov

SemViSGenerator	abstraktný generátor implementuje spoločné metódy ostatných generátorov
RepositoryListGenerator	generuje XML obsahujúce zoznam evidovaných úložísk spolu s ich parametrami
OntologyExplorerGenerator	pomocný generátor slúžiaci na zaznamenanie parametrov sedenia pre každého používateľa
OntologyListGenerator	získa na základe dopytu do zvoleného úložiska zoznam všetkých ontológií, ktoré sa v ňom nachádzajú
OntologyDetailGenerator	generuje detailné informácie o zvolenej ontológií
OntologyPrefixGenerator	umožňuje priradiť zvolenej ontológií skrátený názov (tzv. prefix)

ClassExplorerGenerator	pomocný generátor slúži na zaznamenanie zvolených ontológií a aktuálne prehliadanej triedy do sedenia pre každého používateľa
ClassDescendantsGenerator	generuje zoznam nasledovníkov danej triedy
ClassDetailGenerator	generuje detail triedy
ClassPredecessorsGenerator	generuje zoznam predchodcov danej triedy
HistoryCommandGenerator	generuje históriu zadaných príkazov
GraphExplorerGenerator	pomocný generátor slúži na nastavenie módu prehliadania
GraphGenerator	generuje grafický výstup vo forme grafu zakódovaného v súbore GraphML

2. *Prístupové objekty k úložisku*. Prístupové objekty zabezpečujú prístup do úložiska a namapovanie získaných dát na interné objekty, ktoré používajú generátory. Skrývajú rozhranie poskytované triedami Sesame API za jednoduché prístupové metódy, používane pre jednotlivé typy generátorov. V Tab. 8 je uvedený zoznam implementovaných prístupových objektov.

Tab. 8 Zoznam prístupových objektov

SemViSDAO	abstraktný prístupový objekt implementuje spoločné metódy ostatných prístupových objektov
OntologyListDAO	prístupový objekt zabezpečujúci načítanie zoznamu ontológií v danom úložisku
OntologyDetailDAO	prístupový objekt zabezpečujúci načítanie detailu pre zadanú ontológiu
ClassDetailDAO	prístupový objekt zabezpečujúci načítanie detailu pre zadanú triedu
ClassPredecessorsDAO	prístupový objekt zabezpečujúci načítanie zoznamu predchodcov zadanej triedy
ClassDescendantsDAO	prístupový objekt zabezpečujúci načítanie zoznamu nasledovníkov zadanej triedy
ClassEvaluationDAO	prístupový objekt zabezpečujúci načítanie ohodnotenia pre zadanú triedu

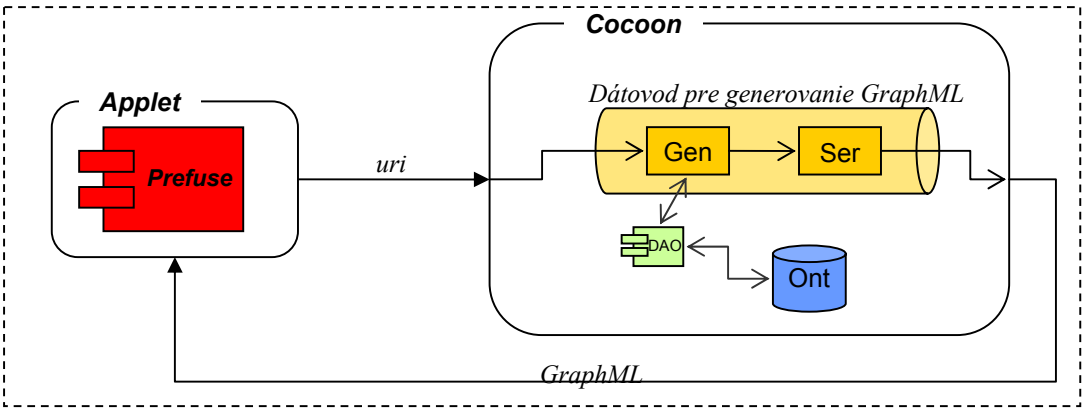
3. *Web servisový klient*. Poskytuje klientske objekty určené na prístup k ohodnocovaciemu modulu, ktorý je implementovaný ako webová služba.
4. *Sada XSL šablón*. Transformujú XML súbory produkované generátormi na HTML stránky.
5. *Pomocné triedy*. Poskytujú túto dodatočnú funkcionalitu požadovanú ostatnými modulmi:

a. sada porovnávačov – umožňuje usporiadavanie zoznamov.
b. zásobník spojení – umožňuje recyklovať spojenia na ontologické úložisko.

Vizualizačný klient

Vizualizačný klient tvoria tieto komponenty:

1. *Statické HTML stránky a CSS štýly*. Predstavujú kostru web aplikácie a zabezpečujú jednotné zobrazenie generovaných stránok.
2. *Vizualizačný applet.* Implementuje grafický prehliadač tried. Je napísaný pomocou knižnice Prefuse, ktorá priamo umožňuje vizualizovať grafy zapísané vo formáte GraphML[22]. S vizualizačným serverom komunikuje prostredníctvom HTTP požiadaviek. Požiadavky obsahujú URI triedy, ktorej okolie chceme vizualizovať. Po prijatí požiadavky vizualizačný server použije na získanie dát rovnaký postup ako v prípade textového prehliadania s tým rozdielom, že výsledné netransformuje do HTML, ale vygenerované dáta serializuje priamo do formátu GraphML (pozri Obr. 21).



Obr. 21 Schéma komunikácie appletu s vizualizačným serverom

6.2.2 Ohodnocovací podsystém

Ohodnocovací podsystém slúži na stanovenie ohodnotení pre jednotlivé triedy, ktoré ontológia obsahuje. Je implementovaný ako webová služba s verejne prístupným rozhraním, cez ktoré je možné pridávať dáta potrebné na stanovenie ohodnotení. V súčasnosti sa pre každú triedu eviduje:

- identifikátor naposledy pristupujúcej aplikácie,
- typ akcie,
- počítadlo prístupov.

[22] http://graphml.graphdrawing.org

Na stanovenie ohodnotení je implementovaná jednoduchá heuristika:

$$evaluation(t_{act}) = \left\lfloor 4 \times \frac{accessCount(t_{act})}{\max_{t \in O}\{accessCount_t\}} + 1 \right\rfloor, \text{ pričom}$$

t_{act} – aktuálna trieda,

O – zjednotenie množina všetkých ontológií v úložisku,

t – ľubovoľná trieda zadnej ontológie.

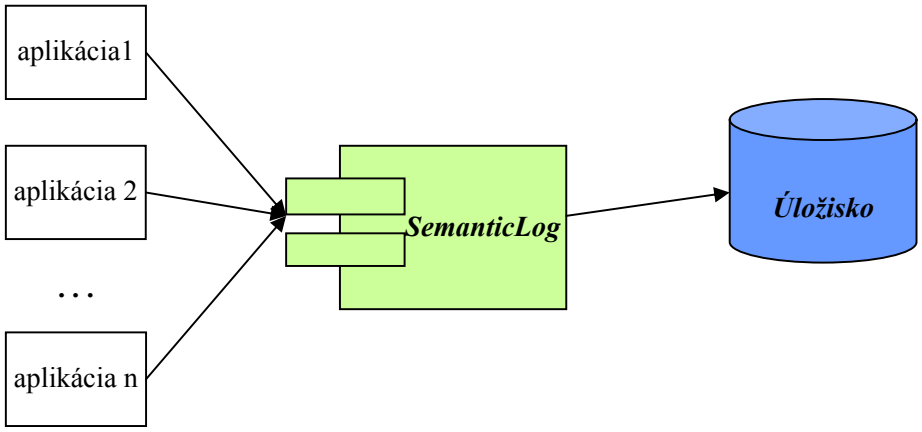
Ohodnotenia sa nevyhodnocujú online, ale v dávke, ktorú je možné inicializovať zavolaním príslušnej metódy webovej služby, ktorá ohodnocovací modul implementuje.

Ohodnocovací modul tvoria tieto komponenty:

1. modul získavania dát,
2. modul zaznamenania dát,
3. modul ohodnocovania.

Získavanie dát

Získavanie dát sa realizuje pomocou nástroja *SemanticLog*, ktorý zaznamenáva odchytené akcie používateľa. Akcie sa spracovávajú a zaznamenávajú v ohodnocovacej ontológií (pozri Obr. 22). Aktuálne nie je ukončená integrácia nástroja SemanticLog a ohodnocovacieho modulu, a preto sú používateľské akcie simulované pomocou externého nástroja.



Obr. 22 Schéma získavania dát, ktoré budú použité pri ohodnocovaní tried

Zaznamenanie

Zaznamenanie používateľských akcií zahŕňa konverziu akcie do interného formátu a následný zápis do ontologického úložiska. Zaznamenanie je implementované transparentne, t.j. klient sa nemusí strať o spracovanie predchádzajúcich dáta.

Postup spracovania akcie ilustruje nasledovný pseudokód:

```
INPUT: resUri: URI, app:STRING, action:ACTION_TYPE
GLOBAL: repository: REPOSITORY
query ← createQuery(resUri,app,action);
graph ← constructGraph(query);
if repository.contains(graph) then
        repository. update(graph);
else
        repository.insert(graph);
```

Pričom *graph* prislúcha internému formátu uchovávania RDF modelu, t.j. reprezentuje množinu RDF výrokov.

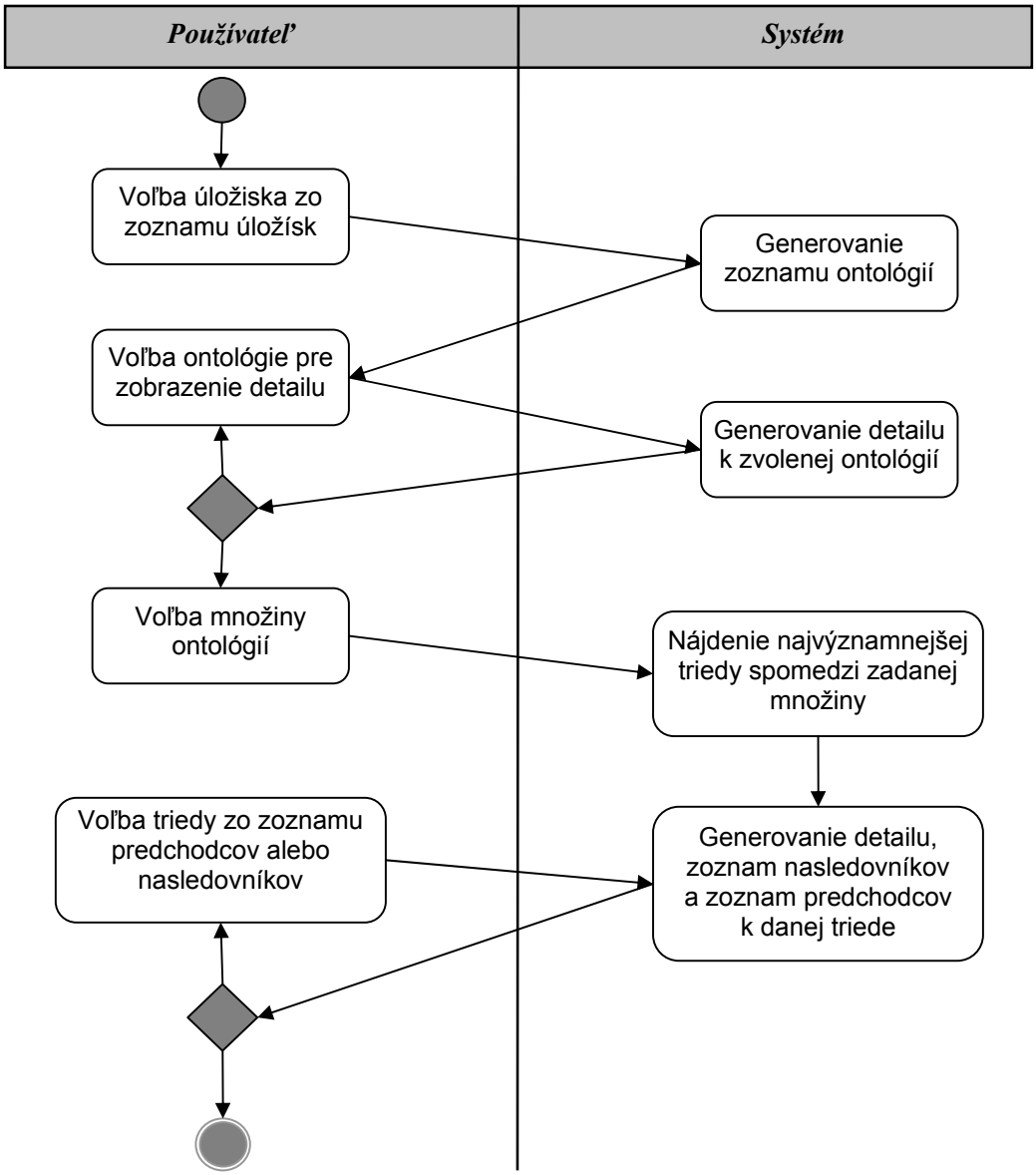
Vyhodnocovanie

Vyhodnocovanie zaznamenaných akcií sa realizuje dávkovým spracovaním. Pre dávkové spracovanie sme sa rozhodli pretože sa môže jednať o časovo náročnú operáciu, ktorú v princípe nie je vhodné realizovať online. Ďalej v reálnom prostredí sa môže stať, že pri používaní vizualizačného systému sa budú veľmi radikálne meniť ohodnotenia jednotlivých tried, čo môže pôsobiť mätúco pre používateľov, ktorý v jednej odpovedi na požiadavku dostanú vysoké ohodnotenie triedy a v ďalšom (aj následnom) výrazne zmenené hodnoty. Algoritmus stanovenia ohodnotení je nasledovný:

```
INPUT: repository: REPOSITORY
GLOBAL: updateEvalQuery : QUERY
maxAccessCount ← getMaxAccesCount(repository);
graph ← constructGraph(updateEvalQuery);
foreach statement in graph do
        accessCount ← getAccessCount(statement);
        newEval ← computeEvaluation(accessCount,maxAccessCount);
        statement.update(newEval);
end;
repository. update(graph);
```

Používateľské rozhranie vizualizátora

Používateľské rozhranie tvoria dve časti, textový prehliadač a grafický prehliadač. Podrobný popis všetkých obrazoviek je uvedený v prílohe B. Na tomto mieste uvádzame len sekvenciu akcií ktorá vedie k aktivovaní prehliadača tried v textovom móde (pozri Obr. 23).



Obr. 23 Sekvencia možných akcií pri prehliadaní tried

6.3 Zhodnotenie prototypu

Implementovali sme vizualizátor ontológií, ktorý umožňuje prehliadať štruktúru ontológie v dvoch módoch, v textovom móde a grafickom móde. Pričom jednotlivé entity vizualizácie sú kvalitatívne ohodnotené, čo umožňuje používateľom jednoducho sa orientovať vo vizualizácii a zároveň je podporená aj navigácia v systéme.

Hlavným cieľom prototypovania bolo ohodnotenie vhodnosti navrhnutej metódy inkrementálneho vizuálneho prehliadania pri prehliadaní rozsiahlych priestorov metadát. Vzhľadom na to, že prototyp bol nasadení iba v testovacom prostredí a neboli vykonané žiadne reálne experimenty, je veľmi ťažké vyriecť konečné závery. Avšak z lokálneho použitia v rámci testovania môžeme vyriecť tieto, aj keď možno subjektívne, závery:

- Navigácia vo vizualizácii, ktorá je podporená dodatočnou informáciou o dôležitosti jednotlivých entít, je výrazne jednoduchšia ako v prípade, keď takúto informáciu nemáme. Jedná sa najmä o prípad, kedy používateľ nie je oboznámený s vizualizovaným priestorom.
- Z hľadiska zapamätania si jednotlivých častí vizualizovaného priestoru, je veľmi vhodné umožniť používateľovi inkrementálne budovať vizualizáciu.
- Poskytnutie viacerých oddelených pohľadov na vizualizovaný priestor napomáha rýchlejšie pochopiť jeho štruktúru.
- Separovanie informácií do samostatných skupín umožňuje výrazne sprehľadniť vizualizáciu.

7. Zhodnotenie

Preložená práca sa zaoberá problematikou vizualizácie ontológií, snaží sa poskytnúť ucelený prehľad o súčasnom stave v tejto oblasti, pričom diskutuje problémy, ktoré vznikajú pri tomto procese, analyzuje ich a snaží sa nájsť a taktiež overiť vhodné riešenie, ktoré čo najviac eliminuje tieto problémy.

V úvodnej časti sme zamerali najmä na analýzu problémovej oblasti. Cieľom bolo porozumieť princípom sémantického webu a oboznámiť sa s jazykmi na reprezentáciu metadát, ktoré obohacujú informácie publikované na webe. V ďalšom sme identifikovali a analyzovali existujúce nástroje umožňujúce vizualizovať takto reprezentované metadáta a snažili sme sa vyhodnotiť ich kladné a záporné stránky. Ako najväčší problém sme identifikovali rozsah a mnohotvárnosť priestoru metadát, ktoré chceme vizualizovať, kde už aj priemerne veľké ontológie obsahujú pomerne veľké množstvo tried a ešte väčšie množstvo vlastností rôzneho typu. Takáto štruktúra dát si vyžaduje pri vizualizácii špecifický prístup. Nie všetky analyzované nástroje berú tento fakt na zreteľ. Napríklad veľmi častým prípadom je zahltenie používateľov obrovským množstvom entít, čoho dôsledkom je, že navigácia v takomto priestore je takmer nemožná.

Hlavným cieľom tejto práce bolo navrhnúť metódu, ktorá tento problém čo najviac minimalizuje. Navrhli sme modifikáciu všeobecne známej metódy inkrementálneho prehliadania, ktorá sa nesnaží vizualizovať celý priestor naraz, ale postupne. Kľúčovým je v prípade tejto metódy pojem okna, ktoré predstavuje obmedzenú časť z celého priestoru, ktorá sa v danom momente prezentuje používateľovi. Problémom takéhoto prístupu k vizualizácii je strata globálneho prehľadu a s tým spojené ťažkosti pri navigácií, ktorá v tomto prípade predstavuje posun okna.

Riešenie, ktoré sme navrhli tkvie v možnosti kvalitatívne ohodnotiť jednotlivé entity vizualizovaného priestoru, pričom na základe týchto ohodnotení je možné automatizovane generovať obsah okna a podporiť používateľa pri navigácii. Navrhli sme všeobecnú ohodnocovaciu ontológiu, ktorá umožňuje uchovávať metadáta o ľubovoľných zdrojoch, ktoré ontológia opisuje. Pričom pri napĺňaní tejto ontológie môžeme využívať spätnú od jej používateľov, čo môžu byť jednak ľudia, ale aj externé aplikácie.

Následne sme navrhli vizualizačný nástroj, ktorý túto metódu implementuje, pričom snahou bolo dosiahnuť čo najvšeobecnejšiu architektúru určenú pre vizualizáciu rozsiahlych priestorov metódu inkrementálneho prehliadania. Následne sme na základe tohto návrhu implementovali prototyp vizualizátora, ktorý umožňuje vizualizovať obsah ontologického úložiska Sesame. Vizualizátor poskytuje textovú a grafickú vizualizáciu okna, ktorá predstavuje detail aktuálne hlavnej triedy a jej kontext vo forme ohodnoteného zoznamu jej nasledovníkov a predchodcov.

Implementovaný prototyp je vhodné rozšíriť o ďalšiu funkcionalitu, ktorá by umožnila komplexnejšie definovať obsah okna. Napríklad stanovením globálnych obmedzení na hodnotu ohodnotenia alebo počtu elementov v okne, alebo polomeru okna je možné efektívnejšie definovať jeho obsah.

Ďalším možným vylepšením systému je sofistikovanejšie vyhodnocovanie metadát uložených v ohodnocovacej ontológii, jej rozšírenie o ďalšie parametre v kombinácii s inteligentnou analýzou štruktúry ontológie, ktorá dokáže ohodnotiť sémantiku jednotlivých entít v ontológii. Ak uvedený prístup zovšeobecníme, je možné vytvárať ohodnotenia pre kombinácie entít, t.j. nie jednotlivo, ale pre celé štruktúry. Ak by sa tento proces dostatočné zautomatizoval je možné takýmto spôsobom vyselektovať určitú obmedzenú množinu vzorov, na ktoré by sa dali následne aplikovať špecifické pohľady, ktoré by sa mohli prispôsobovať potrebám používateľov.

Referencie na zdroje

Použitá literatúra

1. Antoniou, G., van Harmelen, F.: *A Semantic Web Primer*. MIT Press, 2004, 272 pp.
2. Aduna: *Cluster Map Library 2005.1. Integration Guide*, 2005. http://aduna.biz/products/technology/clustermap/docs/Aduna-Cluster-Map-2005.1-Integration-Guide.pdf.
3. Barla, M., Tvarožek, M.: *Automatic Acquisition of Comprehensive Semantic User Activity Logs*. In: Návrat, P. et al: Tools for Acquisition, Organisation and Presenting of Information and Knowledge, 2006, pp. 169-174.
4. Broekstra, J., Kampman, A., Van Harmelen, F.: The Sesame: *A Generic Architecture for Storing and Querying RDF and RDF Schema.* In: The Semantic Web - ISWC 2002, Lecture Notes in Computer Science, vol. 2342, Springer, 2002, pp 54-68.
5. Bederson, B., Shneiderman, B., Wattenberg, M.: *Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies*. In: *ACM Transactions on Graphics*, October 2002, pp. 833-854.
6. Champin, P. A.: *RDF Tutorial*, 2000, http://www710.univ-lyon1.fr/~champin/rdf-tutorial/rdf-tutorial.pdf.
7. Eklund, P., Roberts, N., Green, S.: *OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser*. In: First International Symposium on Cyber Worlds (CW'02), 2002. pp. 405-411.
8. Fluit, Ch., Sabou, M., van Harmelen, F.: *Supporting User Tasks through Visualisation of Light-weight Ontologies*. In: Handbook on Ontologies, Steffen Staab and Rudi Studer (Editors), Springer Verlag, 2004, pp. 415-434.
9. Heer, J., Card, S. K., Landay, J.: *Prefuse: a toolkit for interactive information visualization*. In: Proceeding of the SIGCHI conference on Human factors in computing systems, April 2004, pp. 421-430.
10. Herman, I., Melançon, G., Marshall, M. S.: *Graph Visualisation in Information Visualisation*: *a Survey*. In: IEEE Transactions on Visualization and Computer Graphics, 2000, pp. 24-44.
11. Manola, F., Miller, E.: *RDF primer*, 2004. http://www.w3.org/TR/rdf-primer.
12. Marshall, M.S., Herman, I., Melançon, G.: *An object-oriented design for graph visualization*, In: Software - Practice & Experience, 2001, 31, pp. 739-756.
13. North, S: *Incremental Layout in DynaDAG*. In: Proceedings of GD'95, Springer-Verlag, Lecture Notes in Computer Science, vol. 1027, 1996, pp. 409-418.
14. Darken, R., Perez, M.: *Interaction Techniques for Large Directed Graph*, 1992, http://lsdis.cs.uga.edu/~ravi/academic/LargeGraphNavigationalTechniques/93-graph-navigation.pdf.
15. Rutlege, L., van Ossenbruggen, J., Hardman, L.: *Making RDF Presentable: Ontegrated Global and Local Semantic Web Browsing*. In: Proc. of WWW 2005, ACM Press, 2005, pp.199-205.
16. Smith, M. K., Welty, Ch., McGuinness, D. L.: *OWL Web Ontology Language Guide*, 2004. http://www.w3.org/TR/owl-guide.

17. Storey, M. A., et al.: *Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege*. In: Workshop on Interactive Tools for Knowledge Capture, Victoria, B.C. Canada, October 2001.
18. Svátek, V: *Ontologie a WWW*, Datakon *2002*, Brno, 19.-20. 10 2002, pp.1-35.
19. Wills, G. J.: *Nicheworks - interactive visualization of very large graphs*. In: Journal of Computational and Graphical Statistics, Volume 8, Number 2, 1999, pp. 190-212.

Analyzované nástroje

1. Protégé, http://protege.stanford.edu
2. OntoViz, http://protege.stanford.edu/plugins/ontoviz/ontoviz.html
3. RDF-Gravity, http://semweb.salzburgresearch.at/apps/rdf-gravity
4. OntoRama, http://www.ontorama.com
5. Jambalaya, http://www.cs.uvic.ca/~chisel/projects/jambalaya/jambalaya.html
6. ClusterMap, http://aduna.biz/products/technology/clustermap
7. OWLViz, http://www.co-ode.org/downloads/owlviz/co-ode-index.php
8. Sesame, http://www.openrdf.com

Prílohy

Príloha A – Technická dokumentácia

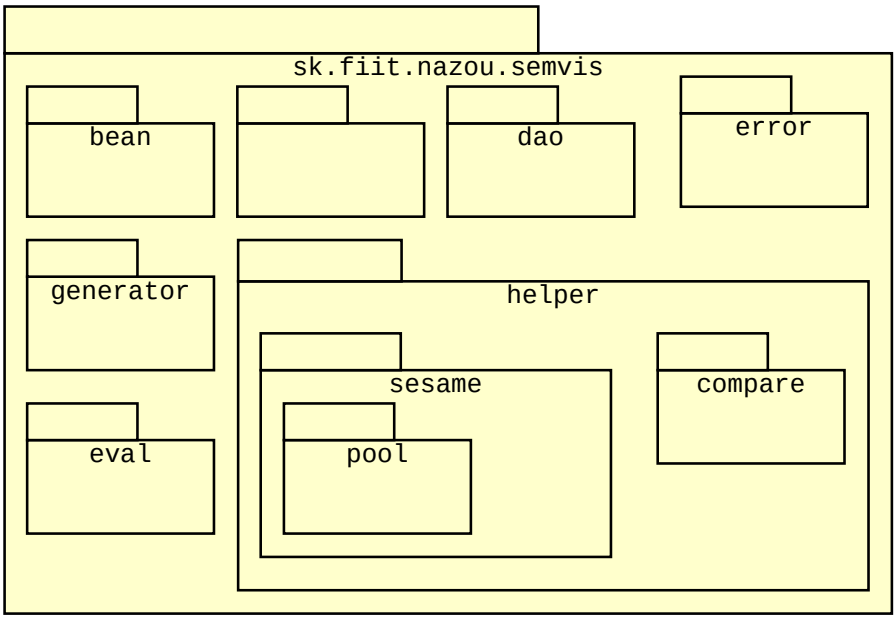
A. 1 Popis modulov

I. Vizualizačný modul

Slúži na vizualizovanie štruktúry ontológie.

Server

Aplikačná logika vizualizačného serveru sa nachádza v balíku `sk.fiit.nazou.semvis` a v jeho dcérskych balíkoch (pozri Obr. 24)

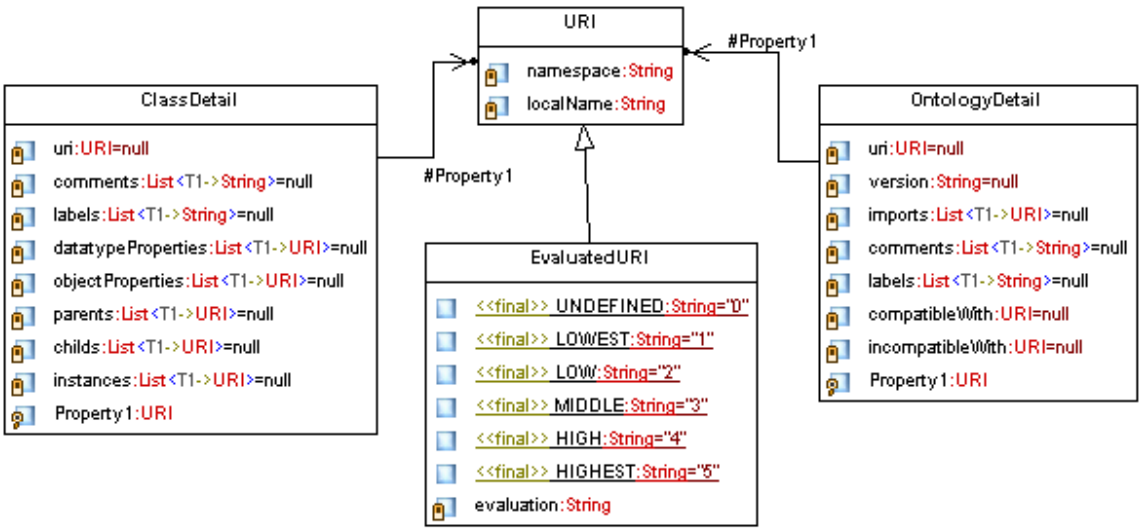


Obr. 24 Štruktúra balíčkov vizualizačného serveru

Balíček bean

Obsahuje triedy slúžiace na interné uchovávanie dát slúžiacich na ďalšie spracovanie. Obsahuje tieto triedy:

`URI`	uchováva internú reprezentáciu URI
`EvaluatedURI`	potomok triedy URI, obsahuje navyše ohodnotenie
`ClassDetail`	uchováva atribúty triedy požadované vo vizualizácii
`OntologyDetail`	uchováva atribúty ontológie požadované vo vizualizácii



Obr. 25 Triedy balíčku `bean`

Balíček config

Obsahuje jedinú triedu zabezpečujúcu manažment konfigurácií a nastavení systému:

`Configuration` obsahuje metódy potrebné na prácu s konfiguračným súborom a uchováva jednotlivé nastavenia systému

Balíček dao

Obsahuje triedy implementujúce prístup do úložiska pre jednotlivé druhy pohľadov. Obsahuje tieto triedy:

`SemViSDAO`	abstraktný prístupový objekt implementuje spoločné metódy ostatných prístupových objektov
`ClassDescendantsDAO`	prístupový objekt zabezpečujúci načítanie zoznamu nasledovníkov zadanej triedy
`ClassDetailDAO`	prístupový objekt zabezpečujúci načítanie detailu pre zadanú triedu
`ClassEvaluationDAO`	prístupový objekt zabezpečujúci načítanie ohodnotenia pre zadanú triedu
`ClassPredecessorsDAO`	prístupový objekt zabezpečujúci načítanie zoznamu predchodcov zadanej triedy
`OntologyDetailDAO`	prístupový objekt zabezpečujúci načítanie detailu pre zadanú ontológiu
`OntologyListDAO`	prístupový objekt zabezpečujúci načítanie zoznamu ontológií v danom úložisku

Obr. 26 Triedy balíčku dao

Balíček error

Obsahuje triedy implementujúce výnimky indikujúce špecifické stavy aplikácie. Všetky implementované výnimky sú podtriedy triedy `java.lang.Exception`:

`InvalidUriException`	výnimka vyhadzovaná pri transformácií URI z reťazca na triedu `sk.fiit.nazou.semvis.bean.URI`
`RepositoryAccessException`	výnimka vyhadzovaná pri akejkoľvek chybe pri prístupe do úložiska
`RepositoryQueryingException`	výnimka vyhadzovaná pri akejkoľvek chybe pri dopytovaní dát z/do úložiska
`SessionExpiredException`	výnimka vyhadzovaná pri vypršaní sedenia

Balíček generator

Obsahuje generátory udalosti SAX, ktoré interne používa Cocoon na reprezentáciu XML súborov, predstavuje jadro systému. Generátory tvoria hierarchiu, na ktorej vrchole je trieda `SemViSGenerator`, ktorá je potomkom triedy `org.apache.cocoon.generation.AbstractGenerator`. Z tohto dôvodu implementujú všetky generátory iba nasledovné 3 metódy:

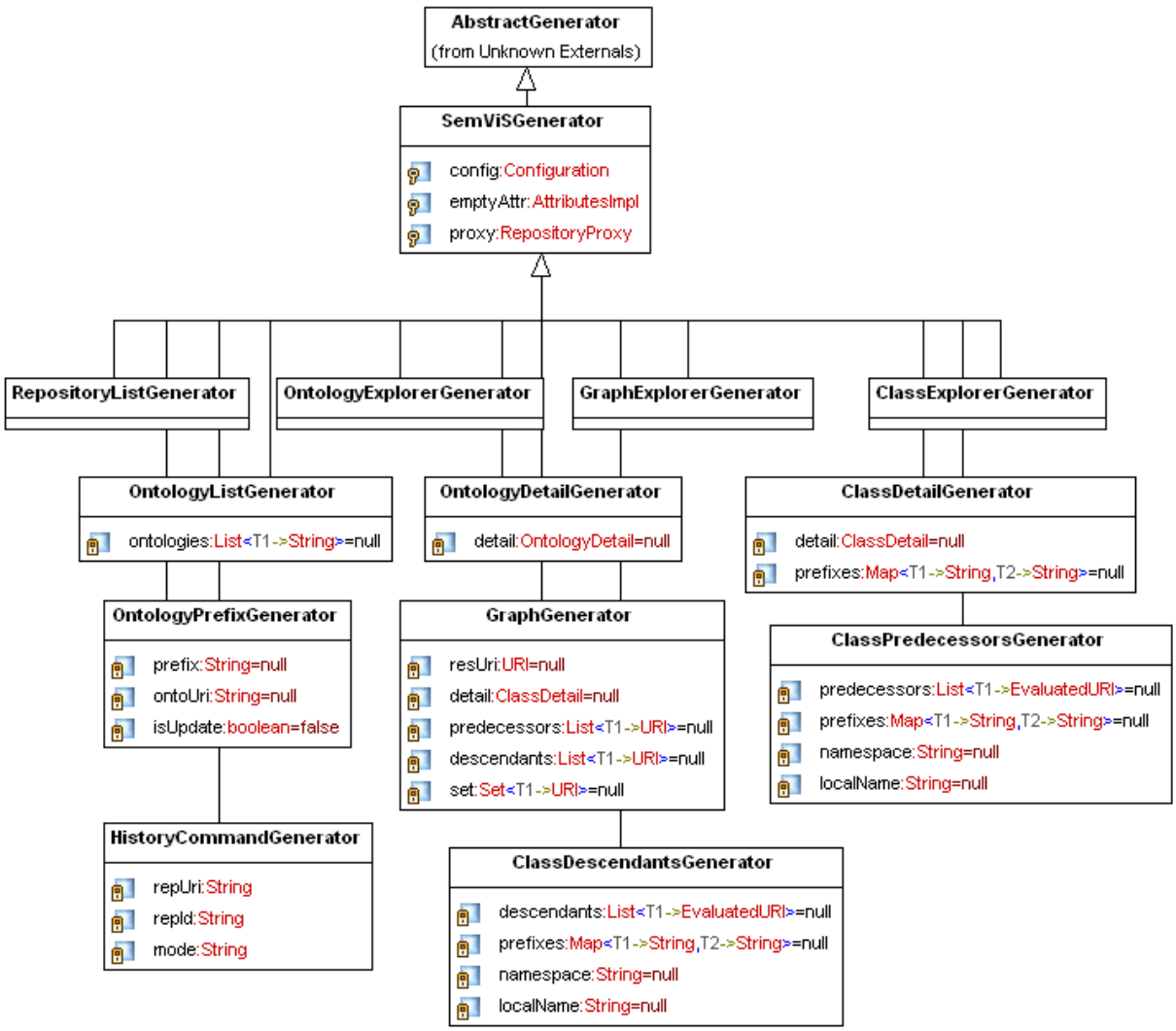
- `setup(SourceResolver res, Map objMod, String src, Parameters par)` – umožňuje získať parametre požiadavky, nastavuje parametre sedenia a cez prístupové objekty získava požadované dáta z úložiska, ktoré ukladá ako inštančné premenné
- `generate()` – slúži na generovanie SAX udalostí, reprezentujúcich výstupný XML dokument
- `recycle()` – umožňuje upratať používané dáta

Balíček obsahuje nasledovné triedy:

`SemViSGenerator`	abstraktný generátor implementuje spoločné metódy ostatných generátorov
`RepositoryListGenerator`	generuje XML obsahujúce zoznam evidovaných úložísk spolu s ich parametrami
`OntologyExplorerGenerator`	pomocný generátor slúžiaci na zaznamenanie parametrov sedenia pre každého používateľa
`OntologyListGenerator`	získa na základe dopytu do zvoleného úložiska zoznam všetkých ontológií, ktoré sa v ňom nachádzajú
`OntologyDetailGenerator`	generuje detailné informácie o zvolenej ontológií
`OntologyPrefixGenerator`	umožňuje priradiť zvolenej ontológií skrátený názov (tzv. prefix)
`ClassExplorerGenerator`	pomocný generátor slúži na zaznamenanie zvolených ontológií a aktuálne prehliadanej triedy do sedenia pre každého používateľa
`ClassDescendantsGenerator`	generuje zoznam nasledovníkov danej triedy

`ClassDetailGenerator`	generuje detail triedy
`ClassPredecessorsGenerator`	generuje zoznam predchodcov danej triedy
`HistoryCommandGenerator`	generuje históriu zadaných príkazov
`GraphExplorerGenerator`	pomocný generátor slúži na nastavenie módu prehliadania
`GraphGenerator`	generuje grafický výstup vo forme grafu zakódovaného v súbore GraphML

Hierarchia generátorov spolu s ich atribútmi je zobrazená na Obr. 27.



Obr. 27 Triedy balíčku `generator`

Balíček eval

Obsahuje vygenerovaného klienta webovej služby ktorá implementuje ohodnocovací modul, t.j. klientske stuby pre transparentné volanie jej operácií. Patria sem tieto triedy:

`SemVisEvaluatorPortType`	rozhranie definujúce operácie webovej služby
`SemVisEvaluatorService`	rozhranie definujúce metódy potrebne na získanie inštancie klientskeho stubu
`SemVisEvaluatorServiceLocator`	implementácia predchádzajúceho rozhrania
`SemVisEvaluatorSoapBindingStub`	klientsky stub

Balíček helper.compare

Obsahuje pomocné triedy potrebné pre zostupné a vzostupné usporiadanie zoznamov tried:

`EvaluationComparator`	implementuje zostupné usporiadanie podľa ohodnotenia triedy
`ReverseEvaluationComparator`	implementuje vzostupné usporiadanie podľa ohodnotenia triedy
`NameComparator`	implementuje zostupné usporiadanie podľa mena triedy
`ReverseNameComparator`	implementuje vzostupné usporiadanie podľa mena triedy

Balíček helper.sesame

Obsahuje pomocné triedy pre transparentný prístup do úložiska Sesame.

`RepositoryParams`	uchováva parametre potrebné na pripojenie sa k úložisku
`RepositoryPool`	reprezentuje zásobník objektov slúžiacich na realizáciu spojenia do úložiska, pričom mapuje objekty triedy SesameRepository na prislúchajúce objekty SesameService
`RepositoryProxy`	mapuje identifikátory úložísk na prislúchajúce zásobníky spojení

Balíček helper.sesame.pool

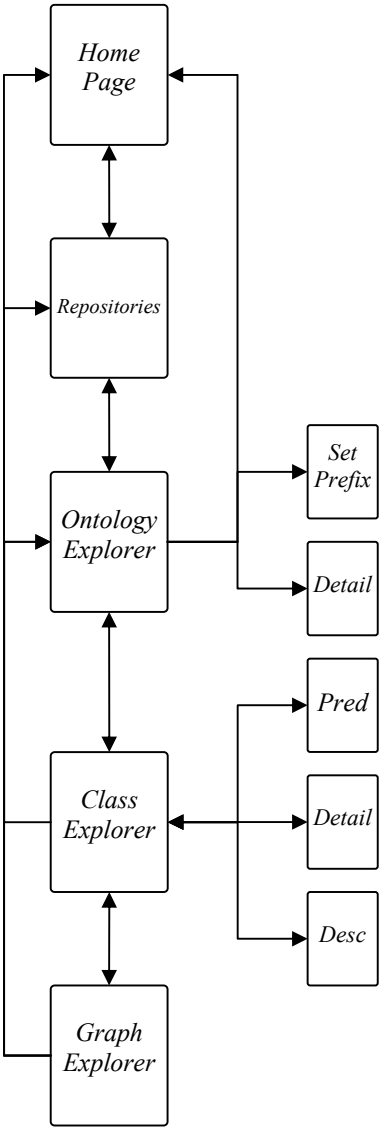
Implementácia zásobníka spojení do Sesame úložiska. Používa rámec pre tvorbu zásobníkov z projektu *Jakarta Commons Pool*.

`SesameServicePool`	implementácia zásobníka objektov SesameService
`SesameServicePoolableFactory`	továrenská trieda umožňujúce generovať objekty triedy SesameService

Klient

Vizualizačný klient tvorí sada HTML stránok, ktorých navigačnú postupnosť je možné vidieť na Obr. 28. Súčasťou stránky „*Graph Explorer*“ je Java applet implementujúci grafický prehliadač tried. Applet, ktorý využíva vizualizačnú knižnicu Prefuse a umožňuje vizualizovať najbližšie okolia danej triedy. Implementujú ho tieto triedy:

`GraphView`	Hlavná trieda vytvára komponenty rozhrania a a registruje poslucháčov reagujúcich na používateľské akcie
`MyFocusControl`	Implementuje reakciu na klik na uzol vo vizualizácii, pričom volá metódu triedy `GraphView`, ktorá zabezpečí načítanie potrebných dát zo serveru.



Obr. 28 Mapa stránok

II. Ohodnocovací modul

Ohodnocovací modul je implementovaný ako webová služba pomocou rámca Apache Axis 1.4. Webová služba je opísaná WSDL súborom, ktorý definuje nasledovné operácie:

getEvaluation:

Popis	Získa hodnotu ohodnotenia pre danú triedu	
	Atribút	***Typ***
Vstup	resourceUri	xsd:string
	evaluation	xsd:string
Výstup	evaluation	xsd:string

updateEvaluations:

Popis	Inicializuje prepočítanie ohodnotení pre všetky triedy	
	Atribút	***Typ***
Vstup	–	–
Výstup	–	–

writeAccessInformation:

Popis	Zaznamená používateľskú akciu do ohodnocovacej ontológie	
	Atribút	***Typ***
Vstup	resourceUri	xsd:string
	application	xsd:string
	action	xsd:string
Výstup	–	

getMostSignificantResource:

Popis	Vráti URI najvýznamnejšej triedy v zadaných ontológiách.	
	Atribút	***Typ***
Vstup	namespaces	array_of_xsd:string
Výstup	uri	xsd:xstring

Ohodnocovacia ontológia

Ohodnocovacia ontológia definuje tieto funkcionálne vlastnosti:

hasEvaluation:

Popis: Vlastnosť priraďuje ľubovoľnej triede bližšie nedefinované reťazcové ohodnotenie
Typ: http://www.w3.org/2002/07/owl#DatatypeProperty
Definičný obor: http://www.w3.org/2000/01/rdf-schema#Resource
Obor hodnôt: http://www.w3.org/2001/XMLSchema#string

lastAccessedBy:

Popis: Vlastnosť priraďuje danej triede identifikátor naposledy pristupujúcej aplikácie
Typ: http://www.w3.org/2002/07/owl#DatatypeProperty
Definičný obor: http://www.w3.org/2000/01/rdf-schema#Resource
Obor hodnôt: http://www.w3.org/2001/XMLSchema#string

hasAccessCount:

Popis: Vlastnosť priraďuje danej triede hodnotu počítadla prístupov
Typ: http://www.w3.org/2002/07/owl#DatatypeProperty
Definičný obor: http://www.w3.org/2000/01/rdf-schema#Resource
Obor hodnôt: http://www.w3.org/2001/XMLSchema#int

A .2 Ukážka implementácie

```
/**
   * Method gets detailed information about selected ontology (specified by URI)
   *
   * @param uri
   *           unique ontology identfier
   * @return
   *           ontology detail
   * @throws RepositoryQueryingException
   * @throws InvalidUriException
   */
public OntologyDetail getOntologyDetail(URI uri)
throws RepositoryQueryingException, InvalidUriException {

    OntologyDetail detail = new OntologyDetail();
    try {
      detail.setUri(uri);

      String query = setUri(ONTOLOGY_VERSION_SERQL, uri.toString());
      detail.setVersion(queryString(query));

      query = setUri(ONTOLOGY_IMPORTS_SERQL, uri.toString());
      detail.setImports(queryURIList(query));

      query = setUri(ONTOLOGY_COMMENTS_SERQL, uri.toString());
      detail.setComments(queryStringList(query));

      query = setUri(ONTOLOGY_LABELS_SERQL, uri.toString());
      detail.setLabels(queryStringList(query));

      query = setUri(ONTOLOGY_BACKWARD_COMPATIBLE_WITH_SERQL, uri.toString());
      detail.setCompatibleWith(queryURI(query));

      query = setUri(ONTOLOGY_INCOMPATIBLE_WITH_SERQL, uri.toString());
      detail.setIncompatibleWith(queryURI(query));

    } catch (MalformedQueryException e) {
      log.error(e);
      throw new RepositoryQueryingException(e);
    } catch (IOException e) {
      log.error(e);
      throw new RepositoryQueryingException(e);
    } catch (QueryEvaluationException e) {
      log.error(e);
      throw new RepositoryQueryingException(e);
    } catch (AccessDeniedException e) {
      log.error(e);
      throw new RepositoryQueryingException(e);
    } catch (InvalidUriException e) {
      log.error(e);
      throw e;
    }

    return detail;
}
```

Príloha B – Používateľská príručka

B. 1. Inštalácia vizualizátora

Predpoklady inštalácie:

1. Nainštalované dve inštancie servletového kontajneru Apache Tomcat, schopné bežať súčasne
2. Nainštalovaný Sesame server na inštancii Tomcatu #1
3. Nakonfigurované a inicializované úložisko Sesame
4. Nainštalovaný a nakonfigurovaný prezentačný server Apache Cocoon na inštancii Tomcatu #1

Postup inštalácie:

A. Inštalácia vizualizačného modulu

1. rozbaliť rar archív *semvis.rar* do adresáru `%TOMCAT1%\webapps\cocoon`
2. nakopírovať Java archív *semvis-cocoon.jar* do adresáru
   `%TOMCAT1%\webapps\cocoon\WEB-INF\lib`
3. nastaviť požadované úložiská a ich parametre a URL na web servisu reprezentujúci ohodnocovací modul v konfiguračnom súbore:
   `%TOMCAT1%\webapps\cocoon\semvis\config\semvis.xconf`
4. a. Pri spúšťaní serveru Tomcat z príkazového riadku:
   Modifikovať súbor `%TOMCAT1%\bin\catalina.bat` pridať nasledovný reťazec pri nastavovaní premennej JAVA_OPTS:
   `-Dconfig.file="%TOMCAT1%\webapps\semvis-eval\config\semvis.xconf"`

   b. Pri spúšťaní serveru Tomcat ako servisu v OS Windows

   i. Inicializácia programu *tomcat5w.exe* , ktorý umožňuje nastavovať parametre servera Tomcat
   ii. Pridať na záložke "Java" v "Java Options" reťazec:
   `-Dconfig.file="%TOMCAT1%\webapps\semvis\eval\config\semvis.xconf"`
   iii. Uložiť nastavenia

B. Inštalácia ohodnocovacieho modulu

1. importovať ohodnocovaciu ontológiu do zvoleného úložiska
2. rozbaliť rar archív *semvis-eval.rar* do adresáru `%TOMCAT2%\webapps`
3. nastaviť parametre úložiska, v ktorom sa nachádza ohodnocovacia ontológia v konfiguračnom súbore:
   `%TOMCAT2%\webapps\semvis-eval\config\eval.xconf`
4. a. Pri spúšťaní serveru Tomcat z príkazového riadku:
   Modifikovať súbor `%TOMCAT2%\bin\catalina.bat` pridať nasledovný reťazec pri nastavovaní premennej JAVA_OPTS:

```
-Dconfig.file="%TOMCAT2%\webapps\semvis-eval\config\semvis.xconf"
```

b. Pri spúšťaní serveru Tomcat ako servisu v OS Windows

i. Inicializácia programu *tomcat5w.exe* , ktorý umožňuje nastavovať parametre servera Tomcat

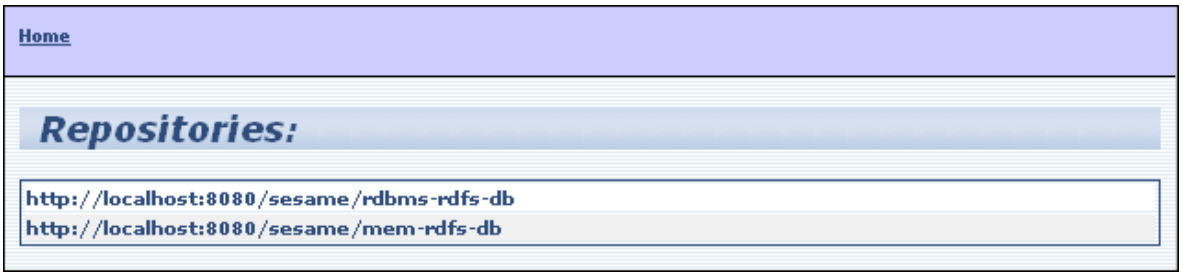
ii. Pridať na záložke "Java" v "Java Options" reťazec:
`-Dconfig.file="%TOMCAT2%\webapps\semvis\eval\config\eval.xconf"`

iii. Uložiť nastavenia

B. 2. Popis používateľského rozhrania

Zoznam úložísk

Obrazovka zobrazuje zoznam úložísk, s ktorými je možné pri vizualizácii pracovať (pozri Obr. 29). Pre inicializáciu vizualizácie je nutné kliknúť na linku želaného úložiska. Je možné zvoliť iba jedno úložisko. Parametre zvoleného úložiska sa automaticky zaznamenajú v systéme, budú potrebné na inicializáciu spojenia.

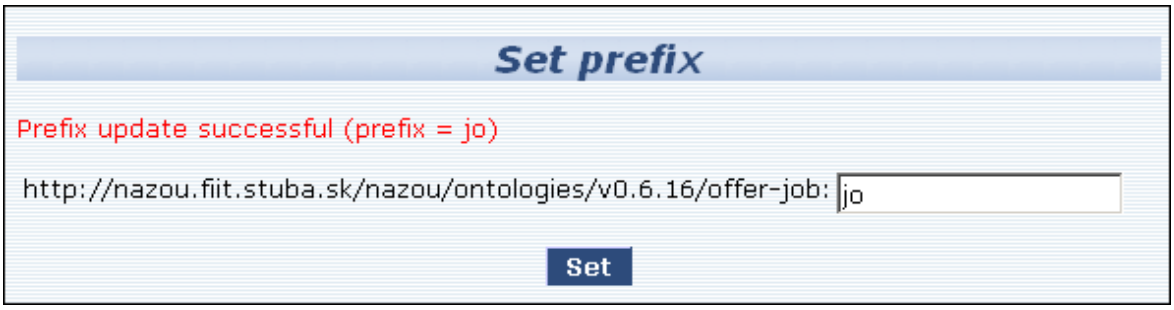


Obr. 29 Okno zoznamu úložísk

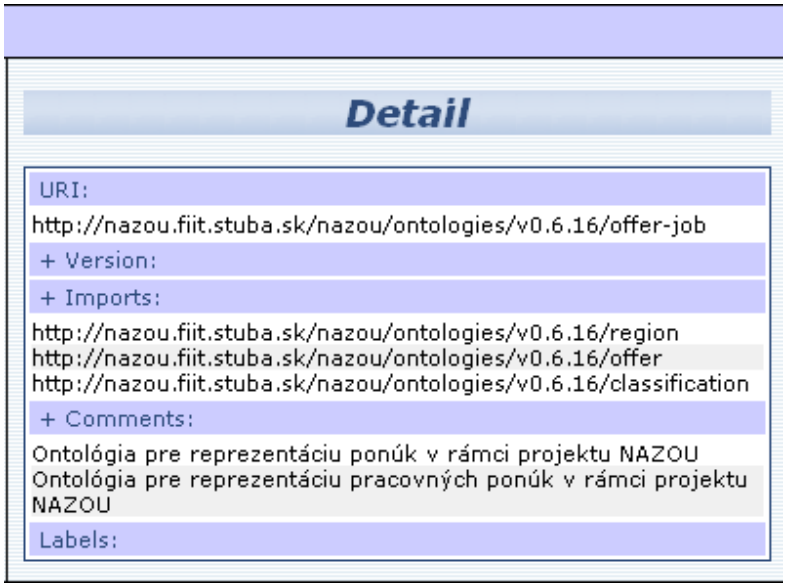
Prehliadač ontológií

Obrazovka zobrazuje zoznam ontológií, ktoré sa vo zvolenom úložisku nachádzajú. Pre každú ontológiu je možne definovať skrátený názov, tzv. prefix a zobraziť detailné informácie o zvolenej ontológii. Okno zoznamu ontológií je uvedené na Obr. 30. Ak chceme ontológii priradiť prefix, klikneme na symbol  , čo otvorí okno „Set prefix", v ktorom je možné prideliť prefix (pozri Obr. 31). Ak chceme aktivovať zobrazenie detailu zvolíme si v zozname linku prislúchajúcu želanej ontológií. V pravej časti sa nám zobrazí detail (pozri Obr. 32), ktorý obsahuje tieto informácie: jednoznačný identifikátor ontológie (angl. URI), verziu ontológie (angl. Version), zoznam importovaných ontológií (angl. Imports), komentáre (angl. Comments), zoznam popiskov (angl. Labels). Všetky položky majú formu rozbaľovacieho zoznamu, pričom nie všetky dáta sú vždy vyplnené. V prípade, že daná položka obsahuje aspoň jednu hodnotu, zobrazí sa v jej titulku znamienko +. Prehliadanie tried aktivujeme zvolením požadovaných ontológií a stlačením tlačidla „Explore".

Obr. 30 Zoznam ontológií



Obr. 31 Priradenie prefixu

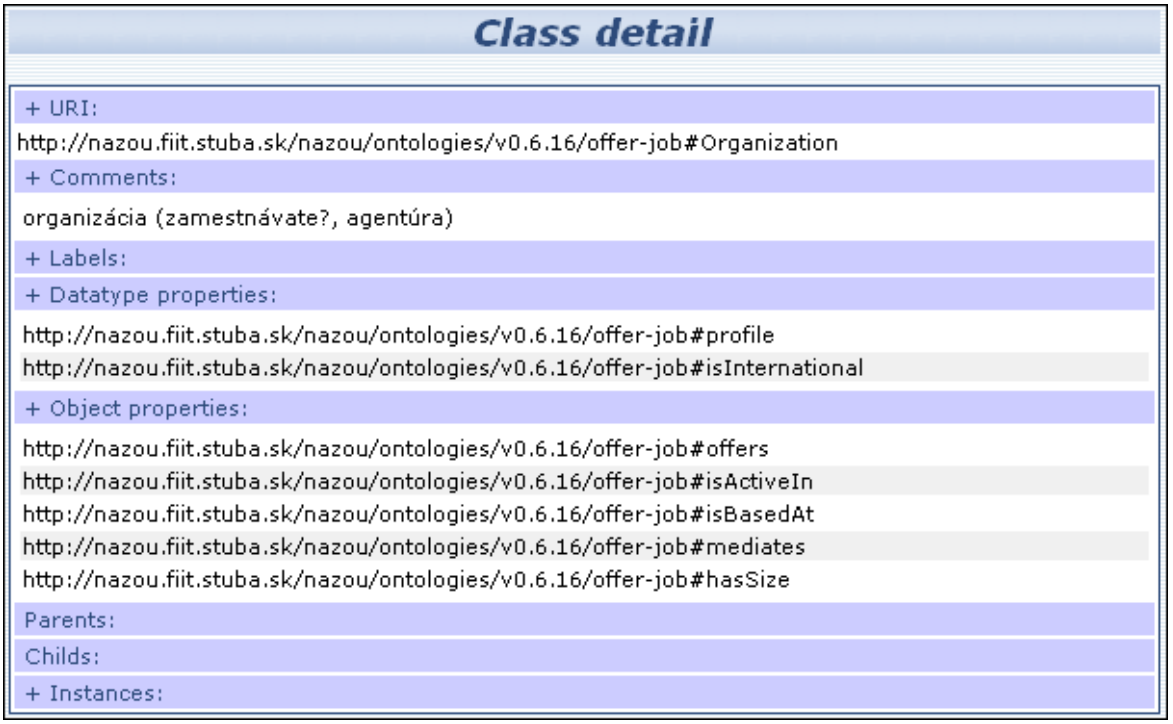


Obr. 32 Detail ontológie

Prehliadač tried

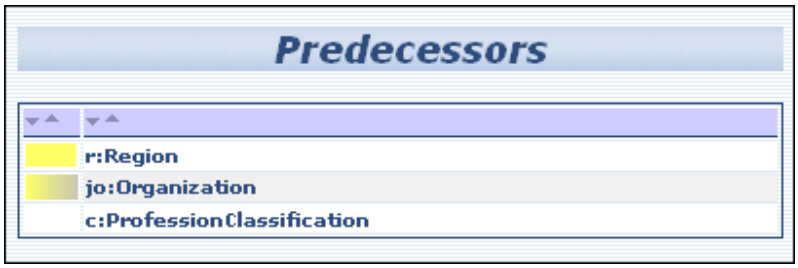
Textový režim

Obrazovka zobrazuje pre každú prehliadanú triedu v ľavej časti jej detail a v pravej časti zoznamy predchodcov a nasledovníkov. Detail triedy (pozri Obr. 33) obsahuje nasledovné informácie: jednoznačný identifikátor triedy (angl. URI), zoznam komentárov (angl. Comments), zoznam popiskov (angl. Labels), zoznam vlastností dátového typu (angl. Datatype properties), zoznam vlastností objektového typu (angl. Object properties), zoznam rodičovských tried (angl. Parents), zoznam detských tried (angl. Childs), zoznam inštancií (angl. Instances).

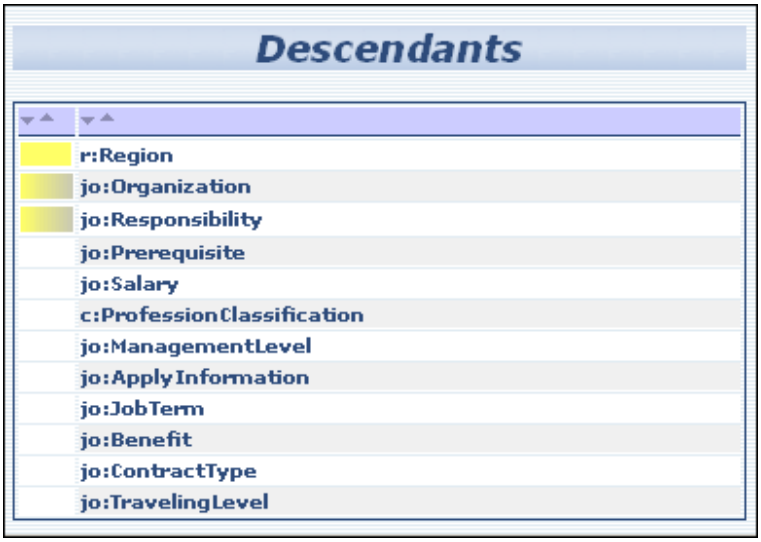


Obr. 33 Detail triedy

Zoznamy predchodcov (angl. Predecessors, pozri Obr. 34) a nasledovníkov (angl. Descendants, pozri Obr. 35) obsahujú triedy zoradené implicitne podľa ohodnotenia, ktoré je vyjadrené 5-timi možnými farbami (pozri škálu ohodnotení uvedenú v kapitole 5.3 Návrh používateľského rozhrania). Ak ohodnotenie obsahuje prázdne pole bielej farby, potom daná trieda nie je ešte v ohodnocovacej ontológii vyhodnotená. Okrem toho je možné usporiadať oba zoznamy podľa mena v zostupnom alebo zostupnom poradí.



Obr. 34 Zoznam predchodcov



Obr. 35 Zoznam nasledovníkov

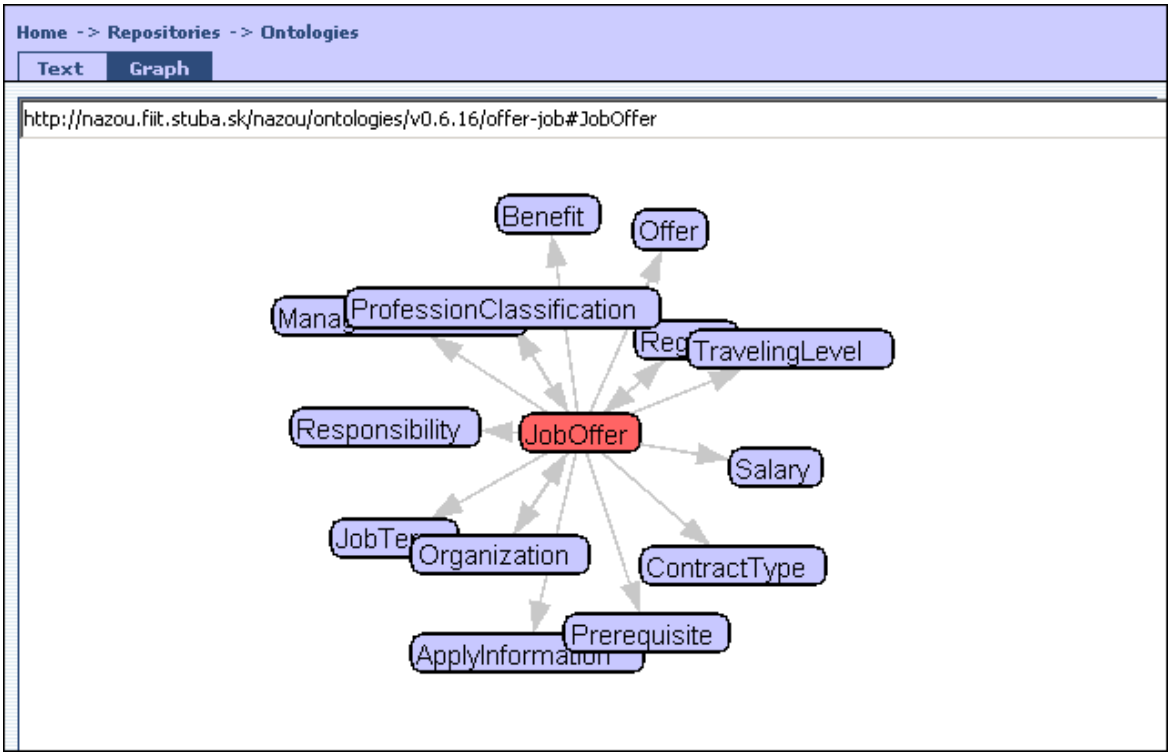
Zoznamy predchodcov, nasledovníkov, predkov a potomkov aktuálnej triedy reprezentujú možné smery prehliadania. Ak chceme zobraziť detail a oba zoznamy pre ďalšiu triedu, stačí na ňu kliknúť a požadované dáta sa dotiahnu zo serveru.

Grafický režim

Po aktivácii grafického prehliadača z navigačného panelu sa zobrazí applet, ktorý automatický načíta zo serveru dáta potrebné pre vizualizáciu okolia aktívnej triedy (pozri Obr. 36). Pre vizualizáciu okolia ďalšej triedy stačí kliknúť na uzol ktorý, jej prislúcha. Pre opätovnú aktiváciu textového prehliadača, stačí z navigačného panelu zvoliť záložku „Text“. Okrem toho máme niekoľko možností ako prispôsobovať vizualizáciu:

- ľavý klik myši na uzol vygeneruje graf prislúchajúci jej okoliu
- pridržaný ľavý klik a súčasný pohyb myši zabezpečí pohyb celého grafu
- pravý klik myši na ľubovoľné miesto zabezpečí prispôsobenie grafu veľkosti okna a vycentrovanie
- kolieskom myši môžme približovať a vzďaľovať vizualizáciu

- pridržaním klávesy CTRL počas kliku ľavým tlačidlom myši umožňuje zvoliť viacero uzlov, ak potom klikneme na pivota z vizualizácie sa odstránia neoznačené uzly, ich opätovné vizualizovanie je možné ak opäť stlačíme kombináciu CTRL + klik na pivota
- v textovom poli v hornej časti môžeme zadať úplné URI triedy, ktorú chceme vizualizovať, ak potom stlačíme tlačidlo „Send“ sa vygeneruje vizualizácia okolia zadanej triedy



Obr. 36 Grafický režim prehliadača tried

Navigačný panel

Počas prehliadania je v hornej časti stále prístupný navigačný panel, ktorý umožňuje návrat na predchádzajúce stránky a v prípade prehliadača tried aj prepnutie medzi textovým a grafickým režimom (pozri Obr. 37).

Obr. 37 Navigačný panel

Príloha C – Základné charakteristiky RDF a OWL

C. 1. RDF a RDF Schema

RDF (angl. Resource Description Framework) predstavuje prostriedok na formálny opis *zdrojov* (angl. resources) v prostredí Internetu [6]. Pod pojmom zdroj sa myslí ľubovoľný objekt, ktorý je identifikovateľný na Internete pomocou jednoznačného identifikátora. Model RDF je založený na trojiciach *subjekt-predikát-objekt.* Takáto trojica sa označuje pojmom *výrok* (angl. statement), ktorý predstavuje jedno tvrdenie o danom subjekte. Pričom zdrojom môže byť nielen subjekt, ale aj objekt, t.j. pomocou výroku sa vytvára akýsi vzťah medzi jednotlivými zdrojmi. Predikát definuje tento vzťah a predstavuje nejakú vlastnosť subjektu. Navyše aj predikát môže mať určený jednoznačný identifikátor, v tom prípade predstavuje taktiež zdroj. Takto definovaný model umožňuje vytvárať vzťahy aj medzi predikátmi. Objektom, ako už bolo spomínané, môže byť zdroj, ale aj tzv. *literál*, ktorý predstavuje atomickú jednotku – elementárnu dátovú hodnotu (napr. string).

Formálne je možné tento model reprezentovať graficky – pomocou orientovaného grafu (pozri Obr. 39) alebo textovo – pomocou rozličných notácií (napr.: N3, RDF/XML, NTriples). Dôležitou vlastnosťou z pohľadu odvodzovania nad takto definovaným modelom, je princíp *reifikácie* výrokov. Reifikácia (zvecnenie) výroku znamená jeho transformáciu na atomický objekt, ktorý sa dá následne chápať ako zdroj. Na ten možno aplikovať uvedený model. Týmto môžeme doplniť ďalšiu meta úroveň, t.j. doplniť ďalšiu sémantiku. Teoreticky opakovanou aplikáciou reifikácie možno pridávať ďalšie a ďalšie meta úrovne, pri zachovaní jednotnej formálnej opisnej štruktúry.

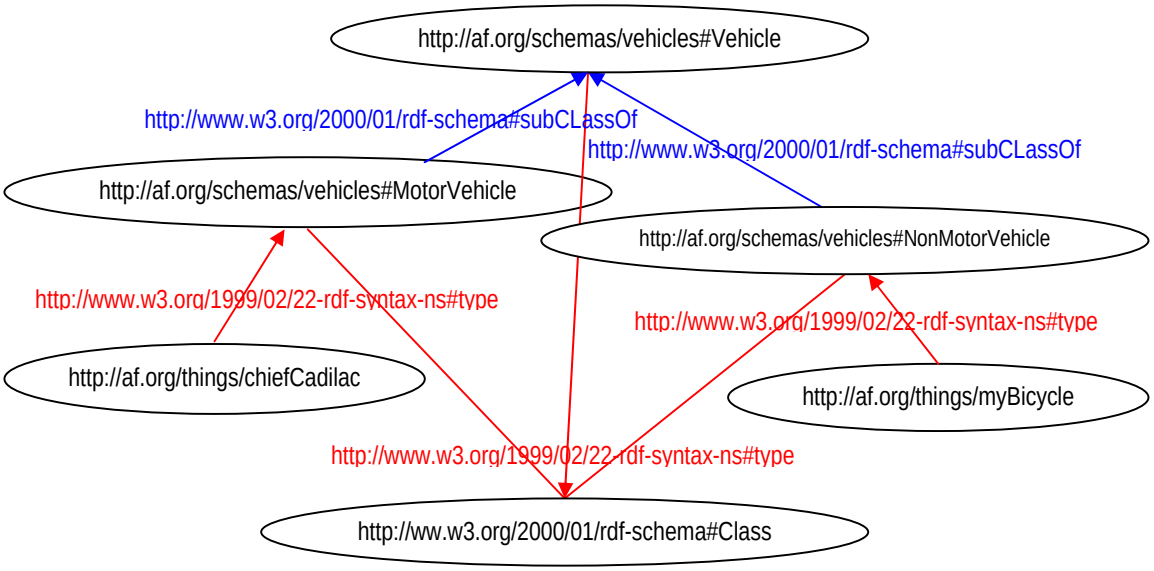
RDF poskytuje spôsob zápisu jednoduchých tvrdení o zdrojoch pomocou pomenovaných vlastností a ich hodnôt. Avšak používatelia potrebujú možnosť definovať si vlastné slovníky (angl. vocabularies), obsahujúce pojmy týkajúce sa ich špecifickej domény. Takýto slovník poskytuje jednotný spôsob definovania tried zdrojov a im prislúchajúcich vlastností. Na tento účel vzniklo rozšírenie RDF s názvom *RDF Schema*[23] (RDFS). RDFS poskytuje možnosť definovať triedy a vlastnosti tried, t.j. formálne vyjadriť obor hodnôt, definičný obor, špecifikovať hierarchiu tried. Zdroje z RDF je možné takto priraďovať triedam definovaným pomocou RDFS. Triedy sa definujú pomocou zdroja `rdfs:Class`, napr. ak chce automobilová firma definovať slovník opisujúci danú oblasť, môže definovať nasledujúce triedy a im prislúchajúce vzťahy:

[23] Alebo tiež RDF Vocabulary Description Language 1.0

```
1   af:Vehicle              rdf:type          rdfs:Class .
2   af:MotorVehicle         rdf:type          rdfs:Class .
3   af:MotorVehicle         rdf:SubClassOf    af:Vehicle .
4   af:NonMotorVehicle      rdf:type          rdfs:Class .
5   af:NonMotorVehicle      rdf:SubClassOf    af:Vehicle .
6   afthings:chiefCadilac   rdf:type          af:MotorVehicle  .
7   afthings:myBike         rdf:type          af:NonMotorVehicle .
```

Obr. 38 Ukážka doménového slovníka formalizovaného pomocou RDFS [11]

Vo výpise sa na Obr. 38 sa najskôr definuje trieda `af:Vehicle` a jej prislúchajúce podtriedy `af:MotorVehicle` a `af:NonMotorVehicle`. Následne je možné priradiť existujúce zdroje `afthings:chiefCadilac` a `afthings:myBike` k týmto triedam. Grafická reprezentácia uvedenej ontológie je na Obr. 39.



Obr. 39 Grafická reprezentácia definovaného slovníka

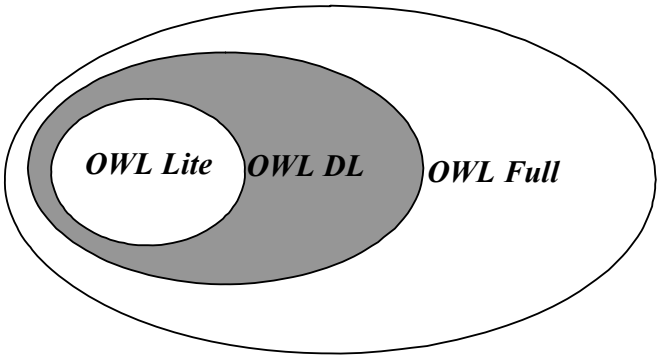
C. 2. OWL

Jazyk OWL (angl. Web Ontology Language) je značkovací jazyk navrhnutý pre definovanie ontológií. OWL rozširuje RDFS o ďalšie elementy súvisiace s triedami a ich vlastnosťami [16]. Pomocou jazyka OWL je možné vytvárať triedy, opisovať ich pomocou definovaných vlastností a vytvárať vzťahy medzi triedami. Takto je možné formálne modelovať určitý systém znalostí o danej oblasti záujmu. Aktuálne sa rozlišujú tri varianty (podjazyky) OWL:

1. *OWL Lite* podporuje tvorbu hierarchií tried a jednoduchých obmedzení,
2. *OWL DL* poskytuje väčšiu expresivitu a možnosť odvodzovania, pričom je stále zaručená vypočítateľnosť a rozhodnuteľnosť. OWL DL je založená na teórii deskripčnej logiky.

3. *OWL Full* poskytuje neobmedzenú expresivitu, čoho dôsledkom je nemožnosť dokázať niektoré tvrdenia, t.j. nie je zaručená vypočítateľnosť.

Vzťah jednotlivých variant je znázornený na Obr. 40, pričom zvýraznený je najpoužívanejší variant OWL DL.



Obr. 40 Vzťah jednotlivých podjazykov definovaných v rámci OWL

Triedy a indivíduá

Základom každej ontológie sú *triedy* a ich inštancie, označované pojmom *indivíduá*. OWL definuje dve triedy `owl:Thing` a `owl:Nothing`. Každá trieda definovaná v ontológií je podtriedou triedy `owl:Thing`, t.j. trieda `owl:Thing` predstavuje množinu všetkých inštancií definovaných v ontológií. Trieda `owl:Nothing` reprezentuje prázdnu množinu, t.j. trieda `owl:Nothing` je podtriedou každej triedy definovanej v ontológii.

Vlastnosti

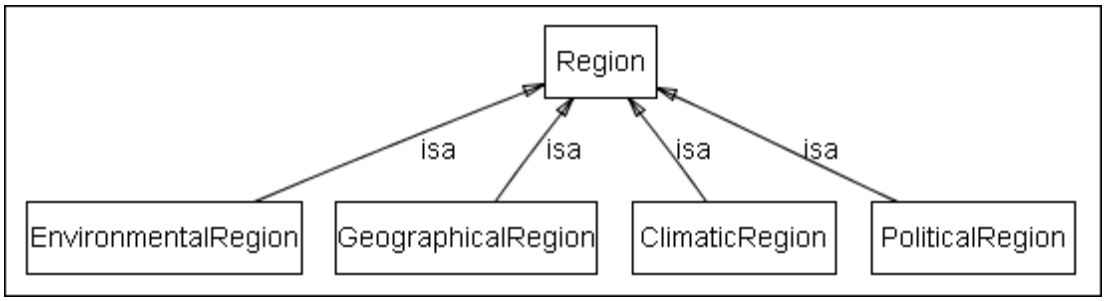
Vlastnosti definujú jednak tvrdenia o triedach a jednak vzťahy medzi triedami. V OWL predstavujú vlastnosti binárne relácie (medzi triedami, resp. indivíduami medzi triedou a hodnotou dátového typu). Na definovanie vlastností triedy sa využíva element `owl:ObjectProperty`, na definovanie vlastností dátových typov element `owl:DatatypeProperty`.

Príloha D – Ontológia pracovných ponúk

Pri analýze existujúcich nástrojov, ale najmä pri overovaní návrhu, t.j. tvorbe prototypu vizualizátora sme využili existujúcu ontológiu pracovných ponúk, ktorá bola vytvorená v rámci program NAZOU. V analýze vykonanej v rámci programu NAZOU boli identifikované jednotlivé koncepty pracovnej ponuky a vzťahy medzi nimi. Výsledná ontológia pracovných ponúk predstavuje explicitnú formálnu špecifikáciu konceptualizácie pracovnej ponuky.

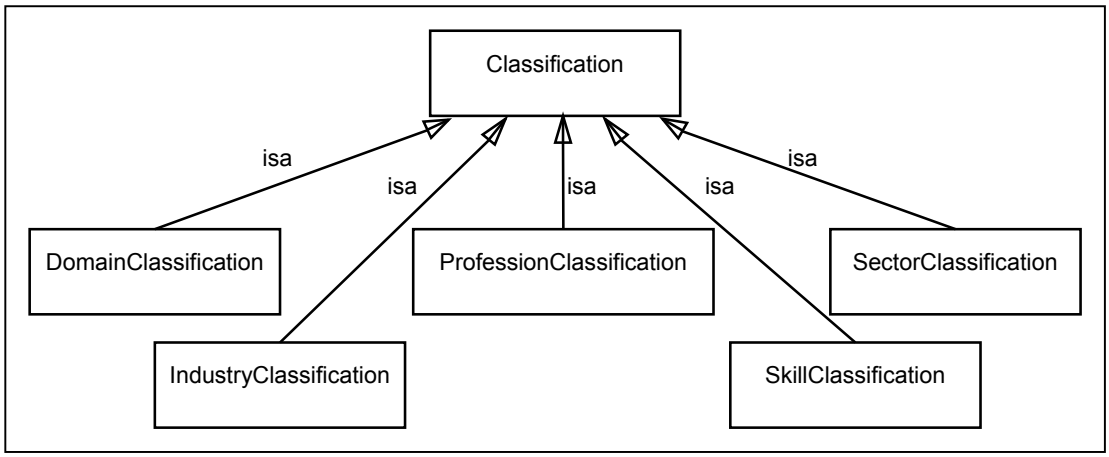
Ontológiu pracovných ponúk možno rozdeliť na doménovo závislú a nezávislú časť. Doménovo závislú časť tvorí ontológia *offer-job*, ktorá importuje zvyšné doménovo nezávislé časti ontológie a využíva ich na definovanie všeobecných konceptov. Doménovo nezávislú časť tvorí:

- Ontológia *Region* – definuje koncepty regiónov, krajín, jazykov a mien, ktoré sa používajú v daných regiónoch (pozri Obr. 41)



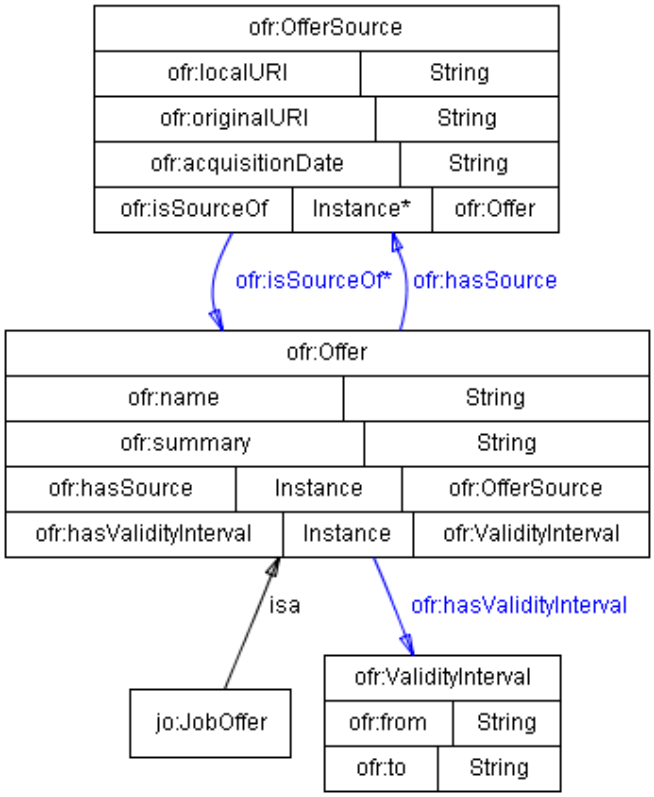
Obr. 41 Štruktúra ontológie Region

- Ontológia *Classification* – definuje hierarchie pre priemyselné odvetvia, profesie, úrovne vzdelania, kvalifikácie a rôzne usporiadania (pozri Obr. 42)



Obr. 42 Štruktúra ontológie Classification

- Ontológia *Offer* – definujúca všeobecnú ponuku (pozri Obr. 43)



Obr. 43 Štruktúra ontológie Offer spolu s detailným opisom jej tried

Príloha E – Obsah elektronického nosiča

Prílohu tejto práce je aj elektronické médium, na ktorom sa nachádzajú všetky dokumenty, ktoré súvisia s touto prácou a aj jej elektronická forma. Okrem toho sa na ňom nachádzajú zdrojové súbory ako aj skompilované verzie prototypu vizualizačného nástroja. Štruktúra média je nasledovná:

CD/
- **doc/** *{dokumentácia a informačné zdroje}*
  - **praca/** *{diplomová práca}*
    - dp-jemala.doc
    - dp-jemala.pdf
  - **literatura/** {použitá literatúra}
- **src/** *{zdrojové súbory}*
  - **eclipse/** *{projekty pre vývojové prostredia Eclipse}*
    - **semvis-eval/** *{ohodnocovací modul}*
    - **semvis-graph/** *{vizualizačný applet}*
    - **semvis-java/** *{vizualizačný server}*
    - **semvis-web/** *{web aplikácia pre Cocoon}*
- **dist/** *{distribučný adresár}*
  - semvis.rar *{web aplikácia pre Cocoon}*
  - semvis-cocoon.jar *{generátory}*
  - semvis-eval.rar *{web aplikácia pre Tomcat}*
- **lib/** *{knižnice a ostatné programové komponenty}*