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

Bc. Marián Šimko

ODHAĽOVANIE VZŤAHOV V OBSAHU VÝUČBOVÉHO KURZU

Diplomová práca

Vedúca diplomového projektu: prof. Ing. Mária Bieliková, PhD.

máj, 2008

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

ZADANIE DIPLOMOVEJ PRÁCE

Meno študenta: **Bc. Marián Šimko**
Študijný odbor: SOFTVÉROVÉ INŽINIERSTVO
Študijný program: Softvérové inžinierstvo
Názov projektu: **Odhaľovanie vzťahov v obsahu výučbového kurzu**

Zadanie:

Pre vzdelávanie sa s výhodou využíva ako prezentačné médium web najmä vďaka jeho dostupnosti. Aby sme dosiahli čo najvyššiu efektívnosť učenia, je dôležité, aby sa výučbový systém dokázal prispôsobovať jednotlivým študentom, resp. skupinám študentov. Skúmajte vlastnosti adaptívnych systémov pre vzdelávanie, ktoré ako prezentačné médium využívajú web. Zamerajte sa na doménový model (reprezentácia prezentovaného obsahu) a skúmajte možnosti automatického vytvorenia častí modelu aj s využitím technológií webu so sémantikou. Navrhnite metódu podpory odhaľovania prepojení jednotlivých konceptov reprezentujúcich výučbový obsah. Návrh overte v doméne výučby programovania pomocou príkladov. Navrhnuté riešenie implementujte a experimentálne overte. Vytvorené riešenie integrujte do prostredia portálu pre podporu výučby programovania, ktorý sa vyvíja v rámci súčasných výskumných projektov na Ústave informatiky a softvérového inžinierstva.

Odporúčaná literatúra:

Brusilovsky, P., Knapp, J., Gamper, J.: Supporting teachers as content authors in intelligent educational systems. In International Journal of Knowledge and Learning, 2006, vol. 2, no. 3/4, 2006, 191–215.

Šimún, M., Andrejko, A., Bieliková, M.: Ontology-based Models for Personalized E-leraning Environment. In: Proc. of ICETA 2007 - 5th Int. Conference on Emerging e-Learning Technologies and Applications. Stará Lesná, Slovak Republic, 2007. pp. 335-340.

Práca musí obsahovať:

- Anotáciu v slovenskom a anglickom jazyku
- Analýzu problému
- Opis riešenia
- Zhodnotenie
- Technickú dokumentáciu
- Zoznam použitej literatúry
- Výstupy celého diplomového projektu vrátane vlastnej diplomovej práce a vytvoreného softvéru (zdrojového kódu s dokumentáciou)

Miesto vypracovania: Ústav informatiky a softvérového inžinierstva, FIIT STU, Bratislava
Vedúci projektu: prof. Ing. Mária Bieliková PhD.

Termín odovzdania práce v letnom semestri: dňa 14. mája 2008

Bratislava, dňa 18. februára 2008

prof. Ing. Pavol Návrat, PhD.
riaditeľ ÚISI

Čestne prehlasujem, že som túto prácu vypracoval samostatne a len s použitím citovaných zdrojov.

Marián Šimko

Ďakujem prof. Márii Bielikovej za neoceniteľné množstvo podnetov a odborných rád počas celej doby riešenia projektu.

ANOTÁCIA

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

Autor: Bc. Marián Šimko
Diplomová práca: Odhaľovanie vzťahov v obsahu výučbového kurzu
Vedenie diplomového projektu: prof. Ing. Mária Bieliková, PhD.
máj, 2008

Predkladaná práca sa zaoberá problematikou podpory tvorby elektronických kurzov pre adaptívne systémy pre vzdelávanie (ASV), ktoré ako prezentačné médium využívajú web. V práci skúmame kľúčové vlastnosti a spôsoby modelovania ASV, pričom sa zameriavame na doménový model (reprezentáciu prezentovaného obsahu). Sústreďujeme sa na možnosti automatického vytvorenia jednotlivých častí modelu.

V práci prezentujeme návrh metódy podpory odhaľovania prepojení jednotlivých konceptov reprezentujúcich výučbový obsah. Jej princíp spočíva v objavení znalostí, ktoré sú zviazané s textovým obsahom kurzu. Využívame vybrané metódy a techniky dolovania v texte. V práci je opísaných niekoľko alternatív riešenia, ktoré sú nakoniec overené a vyhodnotené v doméne výučby programovania pomocou príkladov. Navrhnutá metóda je prostredníctvom autorského nástroja CourseDesigner integrovaná do prostredia portálu pre podporu výučby programovania.

ANNOTATION

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

Author: BEng. Marián Šimko
Diploma thesis: Relationships discovery in educational course content
Supervisor: prof. Ing. Mária Bieliková, PhD.
2008, May

The presented work deals with the problem of authoring adaptive educational systems that use the web as presentation medium. The work aims at an analysis of key features and the ways of modeling such systems. We focus at the domain model (a representation of presented content). The possibilities of automation of its particular parts are examined.

A method of relationships discovery in educational course content is proposed. Relationships are being created between concepts representing educational content. The principle lies within the utilization of information retrieval techniques. Our goal is to discover the knowledge hidden behind the educational text data. For this purpose several alternatives are proposed. These are verified and evaluated in the programming learning domain. CourseDesigner authoring tool was developed for the integration of the proposed method into the portal designed for teaching of programming.

Obsah

1 Úvod

Múdrosť. Taký vágny pojem a pritom taký dôležitý. Odkedy je človek človekom, uvedomuje si, že práve oná múdrosť je kľúčom k úspechu a šťastiu. Múdrosťou riešime problémy, múdrosťou sa vyhýbame prekážkam, múdrosťou zmierňujeme nepriazeň osudu. Múdrosťou dosahujeme víťazstvá. S múdrosťou sa však nerodíme, ani nám sama nenarastie. Musíme ju nejako získať. Tento proces nazývame *vzdelávanie*.

Vzdelávaním človek neustále napreduje. Je prirodzené, že vzdelania chceme nadobudnúť čo najviac a najmä čo najrýchlejšie. V histórii ľudstva bolo len niekoľko momentov, ktoré radikálnym spôsobom zmenili význam tohto fenoménu. Prvým bol azda vznik písma, keď ľudia začali svoje vedomosti archivovať. Neskôr prišiel vynález kníhtlače, ktorý sprístupnil vzdelanie širokým masám. „Revolúciu vzdelávania“ dokonáva internet, ktorý spolu s modernými informačnými technológiami búra bariéry v prístupe k informáciám. Nielen, že možnosť vzdelania máme vždy poruke, vzdelávanie môže byť ešte k tomu šité na mieru.

Pojmy *personalizácia* a *prispôsobovanie* sú, obzvlášť v posledných rokoch, skloňované čoraz častejšie. Adaptívne webové systémy pre vzdelávanie sa dokážu prispôsobiť potrebám používateľa a obracajú paradigmu uniformného vzdelania pre každého o 180 stupňov. Využívajú pri tom sofistikované mechanizmy prispôsobovania, vďaka ktorým študentovi predkladajú to, čo naozaj potrebuje on a nie ostatní. Výučbový proces robia efektívnejším, efektivita si však vyberá svoju daň v podobe veľkej zložitosti celého systému.

Zložitosť sa odráža do procesu tvorby elektronických kurzov. Výučbové materiály nestačí vytvoriť, treba nad nimi definovať určitú metaúroveň, ktorá obsahuje čo najbohatšie informácie o vzťahoch medzi poznatkami. Adekvátna štrukturalizácia kurzu je enormne namáhavá a časovo náročná činnosť. To sú jedny z najväčších prekážok použiteľnosti a rozšírenosti adaptívnych systémov v bežnom živote.

Cieľom tejto práce je priložiť ruku k veľkému dielu adaptívnych systémov pre vzdelávanie. Zameriame sa na zjednodušenie procesu tvorby elektronických kurzov. Navrhneme metódu, ktorá pomocou metód a techník objavovania znalostí napomôže k vytvoreniu doménového modelu kurzu – metaúrovne, ktorá je východiskom pre efektívne prispôsobovanie. Metódu overíme v doméne výučby programovania pomocou príkladov.

V kapitole 2 sa bližšie pozrieme na adaptívne systémy pre vzdelávanie, stručne opíšeme ich architektúru a spôsoby modelovania. Kapitola 3 predstavuje prehľad prístupov k podpore tvorby kurzov. Analýzu problémovej oblasti ukončí

kapitola 4, kde skúmame možnosti metód a techník objavovania znalostí pre získanie potrebných poznatkov o kurze. V kapitole 5 zhrnieme aktuálny stav a definujeme ciele práce. Jadro celej práce – návrh metódy pre automatizované získavanie metadát o kurze, jej jednotlivé varianty – je opísané v kapitole 6. Kapitola 7 obsahuje návrh autorského nástroja CourseDesigner – softvérového riešenia, ktoré o. i. integruje navrhnutú metódu do príjemného používateľského prostredia. V kapitole 8 sa venujeme overeniu riešenia v doméne výučby programovania; opíšeme vykonané experimenty a vyvodíme z nich závery. V poslednej kapitole zhodnotíme celú prácu a prediskutujeme možnosti jej ďalšieho rozšírenia.

Súčasťou tejto práce sú prílohy, ktoré obsahujú technickú dokumentáciu k projektu, používateľskú príručku pre nástroj CourseDesigner, ocenený príspevok prezentovaný na konferencii IIT.SRC 2008 a opis obsahu priloženého dátového nosiča.

2 Adaptívne systémy pre vzdelávanie

Adaptívne systémy pre vzdelávanie (ASV)[1] ponúkajú alternatívu ku klasickému uniformnému spôsobu výučby. Vytvárajú model používateľa, kde udržiavajú charakteristiky a znalosti používateľa a na základe nich sa prispôsobujú používateľovým potrebám. Ich zámerom je ponúknuť používateľovi poznatky na adekvátnej úrovni, v adekvátnom množstve a v adekvátnej forme.

História adaptívnych systémov pre vzdelávanie sa píše už viac ako desaťročie. Na počiatku boli inteligentné výučbové systémy (ITS; Inteligent Tutoring System; Brusilovsky, 1992), ktoré zaviedli individuálne delenie učebných osnov (angl. curriculum sequencing) kvôli nájdeniu optimálneho prechodu výučbovým materiálnom pre každého študenta. Neskôr, vďaka rozmachu internetu, začali adaptívne systémy s obľubou využívať ako prezentačné médium web najmä vďaka jeho dostupnosti. Hovoríme o adaptívnych výučbových hypermédiách (AEH; Adaptive Educational Hypermedia; Beaumont, Brusilovsky, 1995). AEH začali pri prispôsobovaní klásť dôraz na podporu pri navigácii (angl. adaptive navigation) a prezentáciu (angl. adaptive presentation) vyučovaných kurzov. Študent sa tak v hyperpriestore ľahšie orientoval, čo sa odrazilo na efektivite jeho štúdia. Dnes, keď už nemusíme nutne hovoriť iba o hypermédiách, moderné adaptívne systémy pre vzdelávanie súhrnne označujeme ako AIWBES (Adaptive and Intelligent Web-Based System; Brusilovsky, 2003). AIWBES sú zatiaľ najmladšou generáciou adaptívnych systémov pre vzdelávanie a stavajú na základoch, ktoré položili ich predkovia.

Funkcionalita prispôsobovania ASV vychádza z ich architektúry. Tá zvyčajne pozostáva z nasledujúcich modelov:

- základné modely:
  - ◇ doménový model,
  - ◇ model prispôsobovania,
- modely súvisiace s výučbou:
  - ◇ model študenta,
  - ◇ model cieľov,
  - ◇ pedagogický model,
- modely súvisiace s prezentáciou obsahu:
  - ◇ model navigácie,
  - ◇ model prezentácie.

1 Pod pojmom adaptívne systémy pre vzdelávanie máme v celej tejto práci na mysli také, ktoré ako prezentačné používajú médium web.

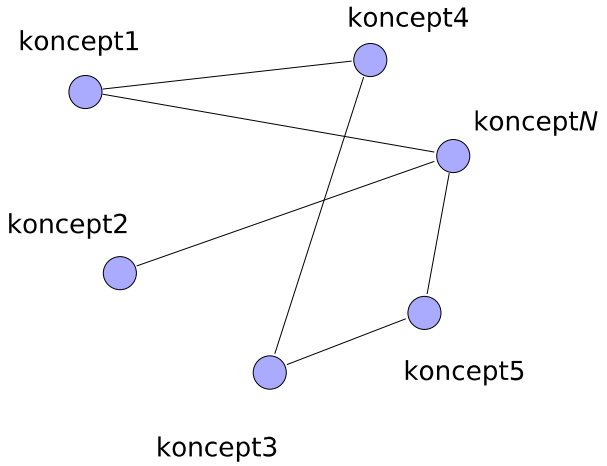
Každý z modelov reprezentuje špecifický aspekt prispôsobovania. Ich implementácia v konkrétnych systémoch vychádza z referenčných modelov, ktoré obsahujú všeobecné myšlienky prístupov k modelovaniu nielen AVS, ale adaptívnych systémov vo všeobecnosti[2]. Jednotlivé modely bližšie opíšeme v nasledujúcich častiach, pričom špeciálnu pozornosť venujeme doménovému modelu.

2.1 Doménový model

Doménový model (tiež model aplikačnej domény) slúži na reprezentáciu štruktúry doménovej oblasti. Základnou štrukturálnou jednotkou je tzv. doménový vedomostný element (angl. *domain knowledge element*; DKE). Veľmi často sa označuje aj ako *koncept*. Koncept je fragment doménovej oblasti, ktorý predstavuje určitý vedomostný „potenciál“ a jeho naštudovanie prinesie používateľovi nejakú pridanú hodnotu (konceptom môže byť napríklad problematika rekurzie, alebo spracovania súboru). Organizácia konceptov v doménovom modeli je rôzna. Rozlišujeme:

- vektorový model,
- sieťový model,
- viacúrovňové modely.

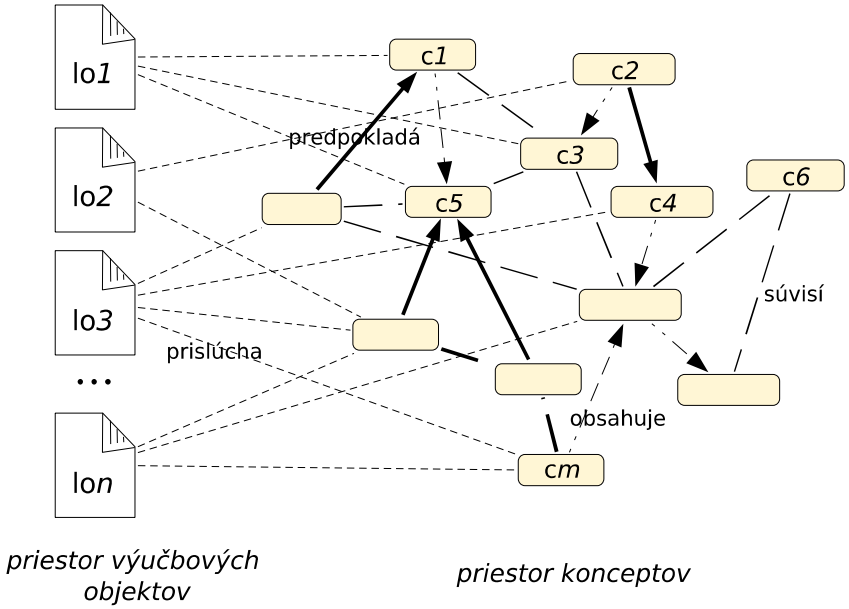
V prípade *vektorového* doménového modelu ide iba o jednoduchú množinu konceptov bez vnútornej štruktúry. S*ieťový* model je o niečo komplexnejší; predstavuje vzájomne prepojenú spleť sémantických väzieb (pozri obrázok 2-1). Najjednoduchším a zároveň najčastejšie používaným príkladom takejto väzby je relácia predpokladu (angl. prerequisity), ktorou modelujeme fakt, že pre pochopenie jedného konceptu je nutné mať určité vedomosti o inom. Tým pádom je možné definovať vhodné poradie učenia sa konceptov.



Obrázok 2-1: Sieťový doménový model (Brusliovsky, 2003).

2 Viac o referenčných modeloch napríklad v (Moravčík, 2006)

Pre dosiahnutie efektívnejšieho prispôsobovania však potrebujeme o kurze explicitne vyjadriť viac informácií. Algoritmy prispôsobovania sú čoraz sofistikovanejšie a vyžadujú bohatší doménový model. Príkladom takého systému je aj ASV projektu PeWePro[3], ktorý zavádza *dvojúrovňový* doménový model. Je reprezentovaný dvoma navzájom prepojenými časťami: priestorom výučbových objektov a priestorom konceptov (pozri obrázok 2-2).



Obrázok 2-2: Dvojúrovňový doménový model (Šimún, 2008).

Priestor výučbových objektov je metamodelom pre samotnú reprezentáciu materiálov, ktoré môžu byť viacerých typov: výklad (*explanation*), cvičenie (*exercise*), schéma programovania (*template*) alebo jednoduchý textu (*simple*). Výučbové objekty, závisiac od typu, môžu pozostávať z fragmentov (*definition*, *hint*, *solution*, *note*). Medzi výučbovými objektmi je definovaná hierarchická štruktúra (relácia *contents*). Jedným z ich atribútov ja aj referencia na konkrétnu podobu textov, ktorá je reprezentovaná v samostatnej dátovej vrstve.

Priestor konceptov predstavuje určité témy výučby, s ktorými dané výučbové objekty súvisia. Koncepty štrukturalizujú oblasť výučby do hierarchie, ktorá pripomína taxonómiu. Existuje medzi nimi viacero typov relácií:

- relácia obsiahnutia (*contains*),
- relácia predpokladu (*prerequisity*),
- relácia súvislosti (*isRelatedTo*).

Medzi oboma priestormi sú definované vzťahy pomocou relácie *belongsTo*. Tá vyjadruje mieru súvislosti medzi výučbovými objektmi a konceptami.

3 Tento projekt v súčasnosti prebieha na FIIT STU v Bratislave a je čiastočne podporovaný Kultúrnou a edukačnou grantovou agentúrou MŠ SR, grant č. KEGA 3/5187/07

2.2 Ďalšie modely ASV

Model prispôsobovania

Týmto modelom sa reprezentujú mechanizmy, ktoré realizujú samotné prispôsobovanie v ASV. Mechanizmy zabezpečujú reakcie na správanie používateľa v systéme. Ich hlavnou úlohou je:

- udržiavanie modelu používateľa,
- personalizácia predkladaných učebných materiálov.

Udržiavanie modelu používateľa spočíva v inicializácii a aktualizácii modelu používateľa. Inicializácia nastáva po prvom vstupe používateľa do systému a pomocou jednej z techník (vyhodnotenie na základe dosiahnutého vzdelania, vyplnenie vstupného testu študentom, a pod.) sa zvolí určitý stereotyp, ku ktorému sa používateľ blíži (je dôležité uvedomiť si, že v tomto momente ešte nemáme takmer žiadne znalosti o vedomostiach používateľa v danej oblasti). O čosi dôležitejšou zodpovednosťou modelu prispôsobovania je aktualizácia modelu používateľa. V tomto prípade prijíma dáta o správaní používateľa z prezentačnej vrstvy a po analýze a aplikácii vybraných heuristík sa vykoná zmena modelu používateľa tak, aby odzrkadľoval aktuálny stav.

Na základe modelu používateľa sa vykonáva personalizácia v podobe:

- prispôsobovania prezentácie,
- prispôsobovania navigácie.

Prispôsobovaním prezentácie sa mení obsah a forma prezentovaných materiálov, využívajú sa pri tom rozličné techniky ako napríklad vkladanie a odstraňovanie fragmentov, usporadúvanie fragmentov, zvýrazňovanie textu, a pod. Prispôsobovanie navigácie spočíva v usmerňovaní používateľa na ceste hyperpriestorom. Technikami môže byť usporiadanie odkazov, schovávanie odkazov, anotácia odkazov, prípadne tvorba mapy odkazov. Viacero techník pre oba typy prispôsobovania aj ich bližší opis je uvedený v (Brusilovsky, 1996).

Mechanizmy prispôsobovania sú v ASV najčastejšie modelované pravidlami, ktoré generujú akcie vykonávajúce zmeny v systéme. Existuje viacero prístupov k reprezentácii pravidiel v závislosti od zložitosti prispôsobovania. Jedným zo spôsobov je použitie jednoduchých IF-THEN konštrukcií. Inou možnosťou je vytvorenie samostatných stratégií prispôsobovania, ktoré môžu prerásť až do komplexného riešenia pozostávajúceho z viacerých úrovní prispôsobenia.

Model študenta

Model študenta uchováva charakteristiky používateľa, na základe ktorých sa v systéme vykonáva prispôsobovanie. Charakteristiky môžeme rozdeliť na závislé (súvisiace s predmetom výučby, napr. vedomosti o jednotlivých konceptoch)

a doménovo-nezávislé (napr. dosiahnuté vzdelanie, používaný jazyk, prípadne hendikepy).

Pre reprezentáciu znalostí o doménovej oblasti sa najčastejšie využíva tzv. prekryvný model (angl. overlay model). Jeho základným princípom je uchovávanie miery znalostí každého konceptu doménového modelu (označuje sa tiež ako *konceptovo-založený* model). Miera znalosti môže byť vyjadrená binárne alebo určitou kvalitatívnou hodnotou, vďaka ktorej je možné rozlišovať viacero úrovní poznania. Takou hodnotou je napr. celé číslo, ktoré vyjadruje odhad množstva používateľových znalostí o príslušnom koncepte, alebo pravdepodobnosť (interval <0;1>), s ktorou používateľ daný koncept ovláda.

Alternatívou je tzv. *stereotypný* model používateľa. V ňom sa nemodelujú explicitné znalosti o konceptoch, resp. stránkach, ale používateľa zaraďujeme do určitých „znalostných" profilov, ktoré nazývame stereotypmi (napr. stereotypy začiatočník, pokročilý, expert). Prispôsobovanie sa vykonáva len na úrovni stereotypov, v pravom zmysle slova sa teda o personalizácii nedá hovoriť.

Model cieľov

Model cieľov je najčastejšie definovaný nad doménovým modelom a modelom študenta. Slúži na modelovanie študijných zámerov. Modelovanie cieľov je špecifikom pre adaptívne systémy pre vzdelávanie. Je nástrojom pre zvýšenie kvality prispôsobovania, pretože pri výučbe berie do úvahy študijný zámer, ktorý si volí sám študent alebo ktorý je zadaný učiteľom. Ciele môžeme vyjadriť jednoducho alebo komplexnejšie. V prvom prípade napríklad výberom konceptu alebo podmnožiny doménovej oblasti, o ktorej chceme získať vedomosti. Komplexnejšie vyjadrenie zahŕňa napríklad nastavenie požadovanej úrovne znalosti, ktorú chceme o koncepte získať.

Modelovanie učebných zámerov sa najčastejšie realizuje samostatnou vrstvou doménového modelu, ktorá obsahuje explicitne vybrané koncepty, ktoré sú predmetom výučby. Hierarchia elementárnych cieľov je rôzna – môže ísť o vzájomne nezávislú sadu konceptov, sekvenciou konceptov alebo stromovú štruktúru.

Pedagogický model

Pedagogický model modeluje pedagogický aspekt výučby. Jeho úlohou je spracovávať charakteristiku používateľa a vyvodzovať z nej závery, ktoré napomôžu vhodnejšiemu prispôsobovaniu používateľovi. Pedagogický model pracuje nad modelom študenta a modelom prispôsobovania. Vo svojej podstate je akousi nadstavbou pre model prispôsobovania špecifickou pre výučbové systémy.

Okrem spracovávania informácii o používateľovi a generovania znalostí o najvhodnejšom spôsobe jeho výučby má pedagogický model za úlohu spracovávať a navrhovať rôzne pedagogické scenáre výučby podľa charakteru doménovej oblasti.

Pedagogický model je na dostatočnej úrovni zatiaľ implementovaný vo veľmi malom množstve adaptívnych systémov. Je to najmä kvôli zložitosti modelovania pedagogických procesov, ktoré musia byť odborného charakteru a nemôžu byť navrhované vývojovým tímom ASV.

Model navigácie

Model navigácie vychádza najmä z doménového modelu a definuje navigáciu v hyperpriestore. Model navigácie v doméne e-vzdelávania slúži na prepojenie znalostí (reprezentovanými konceptami) s im prislúchajúcim študijným materiálom. Tento proces sa často označuje ako indexovanie, kvôli analógii medzi priradením množiny konceptov a priradením kľúčových slov k stránke v hyperpriestore. Prístupy k prepojeniu priestoru znalostí a webových stránok zahŕňajú štyri aspekty vzťahov mapovania (Brusilovsky, 2003):

- kardinalitu,
- výrazovú silu,
- granularitu,
- navigáciu.

Kardinalita mapovania definuje, či ide o indexovanie v pomere 1:1 alebo k jednému informačnému fragmentu prislúcha viacero konceptov (tzv. jedno-konceptové alebo viac-konceptové indexovanie). Jedno-konceptové indexovanie je jednoduchšie a pre autora elektronického kurzu prirodzene intuitívne. Ak je však cieľom vytvorenie naozaj kvalitného vzdelávacieho systému, mapovanie 1:1 nepostačuje. Viac-konceptové indexovanie je na jednej strane omnoho komplexnejšie, avšak o to viac zložitejšie pre tvorcu kurzu. Autori nie sú vždy schopní zachytiť alebo identifikovať všetky väzby medzi konceptami a materiálmi, ktoré ich prezentujú, a príprava kurzu sa tak stáva namáhavou a zložitou. V tejto oblasti sa preto otvára určitý priestor pre automatizáciu procesu mapovania medzi oboma priestormi.

Výrazová sila vyjadruje množstvo informácií, ktoré autori môžu do mapovaného vzťahu uložiť. Ak ide iba o samotné deklarovanie vzťahu (nejaký vzťah vôbec existuje), hovorí sa o tzv. plytkom indexovaní (angl. flat indexing). Systémy s rozsiahlym hyperpriestorom a dômyselnejšími technikami adaptácie však potrebujú do mapovania priradiť informácií omnoho viac. Možnosť, ako to urobiť, je zavedenie rol (napr. *isExample*, *isTest* pre vyjadrenie, že materiál súvisiaci s konceptom je napísaný v podobe príkladu alebo testu) alebo váhovania (pre vyjadrenie miery súvislosti materiálu s konceptom).

Granularita mapovania determinuje „jemnosť“ priradenia konceptov k výučbovému materiálu. Niekedy je vhodné priradiť k jednému konceptu práve jednu stránku v hyperpriestore, inokedy je potrebné identifikovať v materiáli určité fragmenty znalostí a koncepty priradiť k nim. Pre to, aby bol ASV prispôsobivý v najväčšej možnej miere, mali by sme sa pri tvorbe kurzu snažiť objaviť čo najviac

fragmentov. Stále však treba dbať na to, aby zostala vystihnutá podstata a zámer kurzu a aby nedošlo k zbytočnému preinformovaniu používateľa irelevantnými informáciami.

Posledným aspektom je samotná *navigácia*. Tú je možné navrhnúť tak, že prepojenia medzi konceptami a stránkami rovno determinujú navigačné cesty alebo existujú len na konceptuálnej úrovni a sú využívané iba vnútornými mechanizmami prispôsobovania. V prvom prípade hovoríme o tzv. konceptovo-založenej navigácii (Brusilovsky, Schwarz, 1997). Druhý, zložitejší, prípad vyžaduje zavedenie explicitných vzťahov. Vtedy hovoríme o navigácii založenej na indexovaní stránok, resp. navigácii založenej na indexovaní fragmentov.

Model prezentácie

Model prezentácie definuje prezentačnú vrstvu adaptívneho systému. Brusilovsky v (Brusilovsky, 2003) identifikoval štyri prístupy pri štruktúrovaní hyperpriestoru:

- neštruktúrovaný,
- hierarchický,
- tzv. „ASK“,
- objektovo-orientovaný.

Tvorba štruktúry prezentácie nijako nezávisí (resp. nemala by) od doménového modelu a prezentovaných znalostí. Ide len o modelovanie spôsobu doručenia obsahu pre používateľa.

Neštruktúrovaný prístup sa využíva najmä v starších ASV a vychádza z predpokladu, že štruktúra prezentácie materiálov je určená väzbami medzi konceptami v doménovom modeli.

Hierarchický prístup zavádza väzby aj medzi stránky v hyperpriestore. Študijné materiály sú tak okrem väzieb založených na doménovom modeli a určených odporúčaním konceptov implicitne štruktúrované aj na webe.

ASK je špecifický prístup, ktorý prichádza s myšlienkou prepojenia pomocou „rétorických“ odkazov. Tieto odkazy sú formulované ako otázky z výučbovej oblasti a prostredníctvom nich sa používateľ dostane k odpovediam v podobe študijného materiálu.

Poslednou možnosťou je pohľad na hyperpriestor ako na sadu objektov obaľujúcich učebné texty. V takom prípade je možné k návrhu hyperpriestoru pristupovať pomocou štandardných objektovo-orientovaných techník (UML a pod.) a využiť moderné webové technológie.

2.3 Príklady vybraných ASV

V tejto časti zhrnieme poznatky nadobudnuté analýzou vybraných ASV. Záverečné zhodnotenie prezentujeme pomocou prehľadnej tabuľky, kde porovnáme vybrané charakteristiky skúmaných modelov.

Systém AHA! (Adaptive Hypermedia for All!; De Bra, Calvi, 1998) bol navrhnutý s cieľom stať sa univerzálnym rámcom pre adaptívne aplikácie. Z toho vyplývajú aj jeho nevýhody pre doménu elektronického vzdelávania: systém je príliš všeobecný, doménový model je príliš „chudobný“ na opis vzťahov charakteristických pre vzdelávanie. Taktiež neposkytuje takmer žiadnu flexibilitu v modelovaní cieľov výučby. Možnosti prispôsobovania sú nedostatočné, pretože používané techniky sú príliš jednotvárne. Systém neobsahuje explicitný model pre podporu štýlov učenia, čo iba utvrdzuje v dojme, že sa celkovo pre dôsledné a kvalitné vzdelávanie veľmi nehodí.

MOT (My Online Teacher; Cristea, De Mooij, 2003) je predovšetkým nástrojom na tvorbu kurzov. Doménový model už omnoho viac vyhovuje doméne elektronického vzdelávania. Systém zahŕňa model cieľov. Medzi konceptami je viacero relácií, ktoré umožňujú širšie možnosti prispôsobovania. Tie sú podporené robustným troj-vrstvovým modelom prispôsobovania, čo robí zo systému MOT veľmi silný nástroj pre tvorbu komplexných kurzov. Keďže MOT je orientovaný na tvorbu konceptových máp vyučovaných oblastí, jeho nevýhoda spočíva v prezentačnej vrstve, ktorá nie je veľmi prepracovaná. Preto boli vyvinuté nástroje pre transformáciu do iných adaptívnych systémov pre vzdelávanie s bohatšími možnosťami doručenia učebných materiálov používateľovi (Cristea, et al., 2005; Cristea, Smits, De Bra, 2005).

INSPIRE (an INtelligent System for Personalized Instruction in a Remote Environment; Grigoriadou, et al., 2001) je zo skúmaných systémov najkomplexnejší a pedagogicky najprofesionálnejší. Jeho návrh a architektúra vytvára široký priestor pre prispôsobovanie študentovi. Systém má implementované moderné metódy vyvíjané v oblasti ASV: priamu podporu modelovania cieľov a štýlov učenia, navigáciu založenú na indexovaní fragmentov. Ide o integrovaný nástroj, ktorý poskytuje rozhrania nielen pre študentov, ale aj pre tvorcov kurzov.

V prípade systému WebEx (Brusilovsky, 2001) autori kladú dôraz na techniky prispôsobovania. Využíva jednoduchý doménový model, ktorý pre realizáciu výučby programovania, na ktorú je systém určený, postačuje. Zaujímavou vlastnosťou systému je možnosť sémantického obohatenia študijných materiálov pomocou anotácií, ktorými sú podporené vlastnosti prispôsobovania.

Posledným analyzovaným systémom je ASV projektu PeWePro (Šimún, 2008). Ten zakladá na ešte dôslednejšom modelovaní doménovej oblasti ako u predchádzajúcich riešení. Prichádza s členitejšou hierarchizáciou, dvoma časťami doménového modelu a výrazne väčším počtom relácií v oboch priestoroch. Vytvára tak väčší potenciál na precíznejšie prispôsobovanie.

Porovnanie opísaných modelov je uvedené v tabuľke 2-1.

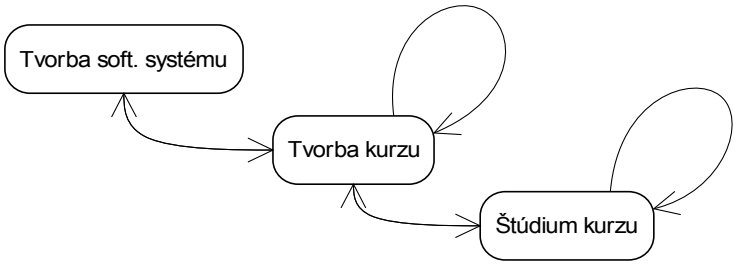
Tabuľka 2-1: Porovnanie vybraných modelov ASV

ASV	Doménový model	Model používateľa	Model navigácie	Techniky prispôsobovania	Pedagogický prístup
AHA!	sieťový	prekryvný binárny	IF	PO, PN	nie
MOT	sieťový[1]	prekryvný váhovaný	IS	PN[2]	možnosť definovania stratégií prispôsobovania
INSPIRE	sieťový[1]	prekryvný váhovaný	IF	PO, PN	podpora štýlov učenia
WebEx	sieťový	prekryvný váhovaný	IS	PO, PN	nie
PeWePro	dvoj-úrovňový	zmiešaný	IF	PO, PN	nie

Legenda: *PO – prispôsobovanie obsahu*
PN – prispôsobovanie navigácie
IF – navigácia založená na indexácii fragmentov
IS – navigácia založená na indexácii stránok
[1] *podpora modelom cieľov*
[2] *rozdelenie na dve časti: priestor konceptov a priestor vedomostných jednotiek*

2.4 Životný cyklus ASV

ASV je softvérový systém, preto jeho životný cyklus spočíva v jeho vytvorení a používaní (pozri obrázok 2-3). *Tvorba* je otázkou uplatnenia metód a techník softvérového inžinierstva, kde tím programátorov (niekedy pod dozorom pedagóga) navrhne a vyvinie webový systém spĺňajúci kritériá ASV. Súčasťou tejto fázy životného cyklu je analýza doménovej oblasti, návrh potrebných modelov a implementácia softvérového riešenia. Samotné používanie softvérového systému je možné chápať v dvoch rovinách: v tvorbe elektronických kurzov (ďalej len kurzov) a ich štúdiu.



Obrázok 2-3: Životný cyklus ASV.

Pri *tvorbe kurzu* vytvára autor (učiteľ) kurz priamo použitím ASV alebo pomocou externého autorského nástroja. Jeho úlohou je naplnenie kurzu študijným materiálom, definovanie jeho štruktúry, výučbových cieľov a pod.

Štúdium kurzov tvorí dominantnú časť používania ASV. Študenti získavajú vedomosti o učebnej látke, pričom učivo je im predkladané v neuniformnej podobe.

Kladie sa dôraz na jednotlivca a systém sa plne prispôsobuje aktuálnemu stupňu študentových vedomostí a „vyučuje“ také témy, ktoré potrebuje pre pochopenie učiva. Sumár procesov prítomných v životnom cykle ASV uvádza tabuľka 2-2.

Tabuľka 2-2: Životný cyklus ASV

Fáza		**Popis**	**Zainteresovaná osoba**
Tvorba softvérového systému	Analýza doménovej oblasti	Získanie potrebných informácií o doménových oblastiach výučby	programátor
	Návrh modelov	Návrh doménového modelu	
		Návrh modelu prispôsobovania	
		Návrh modelu používateľa	
		...	
	Implementácia riešenia	Fyzická realizácia softvérového riešenia	
Používanie softvérového systému	Tvorba kurzu	Naplnenie obsahu, resp. vytvorenie jednotlivých lekcií	učiteľ
		Definovanie štruktúry kurzu	
		Vytvorenie metadát o kurze	
		Definovanie pedagogických cieľov	
		Definovanie pravidiel prispôsobovania	
		...	
	Štúdium	Učenie sa z kurzu	študent

Cieľom ASV je vytvorenie takého média, ktoré zaručí kvalitný výučbový proces s dôrazom na individuálny prístup k študentovi. Kvalita netkvie len v samotnom princípe prispôsobovania, ale aj v obsahu toho, čo sa študentovi prostredníctvom ASV predkladá – v študijných materiáloch. Naplnenie vhodným obsahom však z hľadiska kvality nestačí. Kľúčová je primeraná hierarchizácia, ktorá sleduje výučbové ciele. Zostrojenie kurzu tak, aby spĺňal všetky požadované vlastnosti, je často veľmi zložitý a časovo náročný proces. A práve ťažkopádnosť tvorby kurzov je momentálne jednou z najväčších prekážok pre ideálne využitie potenciálu ASV (Brusilovsky, et al., 2006). Odbúranie tohto aspektu by výrazne urýchlilo prípravu elektronických kurzov, odbremenilo učiteľov od rutinnej „drevorubačskej“ práce a umožnilo nielen efektívnejšiu tvorbu, ale aj štúdium elektronických materiálov.

V súčasnej dobe existuje viacero prístupov a metód, ktoré adresujú vyššie uvedené problémy a snažia sa zjednodušiť proces prípravy elektronického kurzu. Ich analýze sa venujeme v nasledujúcej kapitole.

3 Podpora tvorby kurzov

Prvé adaptívne systémy pre vzdelávanie boli vytvárané tak, že autorské tímy navrhovali softvérové riešenie a zároveň tvorili obsah výučbových kurzov. Zvyčajne išlo o jednoduchšie systémy a naplniť ich materiálom nebol pre zainteresovaného učiteľa veľký problém.

S vývojom ASV však prišlo rozšírenie ich funkcionality, zlepšili sa metódy prispôsobovania, čo malo za dôsledok potrebu použitia dômyselnejšej reprezentácie nielen konceptov, ale tiež ASV ako celku. Komplexnosť a zložitosť celého systému narastala. Rozvoj adaptívnych systémov priniesol aj ich rozšírenie z akademických kruhov medzi širokú verejnosť, kde sa autormi kurzov stávajú „netechnicky" vzdelaní učitelia. Je prirodzené, že v súčasnosti už od takéhoto používateľa neočakávame, aby bol tiež súčasťou kolektívu tvorcov systému (Brusilovsky, et al., 2006).

Tvorba kurzov je dnes samostatná etapa tvorby ASV a ako sme uviedli v časti 2.4, ide de facto o používanie adaptívneho systému (eventuálne externého nástroja). Pre učiteľov je ASV prostriedkom skvalitnenia výučby a nástrojom, ktorý používajú pri práci. Očakáva sa, že túto prácu bude v čo najväčšej možnej miere zjednodušovať a zefektívňovať. Aká je však realita? Snaha o sofistikovanejšie prispôsobovanie priniesla zvýšenie zložitosti celého systému. Tvorba kurzu už nespočíva len v naplnení materiálom, potrebné je namapovanie na jednotlivé časti doménového modelu. Kurz musí byť vhodne štruktúrovaný, je potrebné definovať väzby medzi jednotlivými konceptami. Zložitejší doménový model so sebou prináša väčšiu námahu pri vytváraní jednotlivých lekcií. Tento proces je časovo veľmi náročný a odrádza autorov-učiteľov od používania ASV vôbec.

Reakciou na tento negatívny faktor je snaha tvorcov ASV o návrh metód na automatizovanú podporu tvorby kurzov. Je potrebné poznamenať, že neexistuje všeobecne platný postup ako zjednodušiť proces tvorby kurzov pre všetky typy adaptívnych systémov pre vzdelávanie. Každý ASV má iný doménový model, špecifické metódy prispôsobovania a funkcionalitu prispôsobenú vlastnej doméne výučby, čo sa reflektuje aj do samotnej tvorby kurzov. V nasledujúcej časti predstavíme vybrané prístupy k podpore tvorby kurzov pre ASV.

3.1 Prístupy k podpore tvorby kurzov

Prispôsobovanie v kontexte e-vzdelávania je predmetom výskumu už viac ako desaťročie. Ten so sebou od začiatku nesie aj otázku tvorby a napĺňania obsahu elektronických kurzov. O tom, že význam podpory tvorby kurzov nie je vôbec

podceňovaný, svedčí aj existencia pravidelných pracovných dielní konajúcich sa na viacerých medzinárodných konferenciách[4]. Medzi témy, ktorými sa akademická obec (a nielen ona) v súčasnosti najčastejšie zaoberá, patria (Cristea, 2004b):

- návrhové vzory pre ASV,
- prepojenie ASV s kognitívnymi štýlmi učenia,
- tvorba kurzov pre ASV,
- problémy kolaboratívnej tvorby ASV,
- vhodné techniky prispôsobovania založené na modelovaní používateľa,
- transformácie modelov prispôsobovania medzi ASV.

Ako môžeme vidieť, pojem tvorba – či už kurzov alebo celých ASV – sa skloňuje relatívne často. My sa v tejto práci zameriame na tvorbu kurzov.

Zo snáh o podporu procesu vytvárania elektronických kurzov v dnešnej dobe dominujú dve iniciatívy (Brusilovsky, et al., 2005):

- tzv. *AI* prístup a
- tzv. *HCI* prístup.

AI (z angl. A*rtificial Intelligence*) prístup sa snaží o vytvorenie inteligentného systému, ktorý celú úlohu vytvorenia kurzu (od momentu naplnenia obsahu učiteľom) prevezme na seba. HCI (z angl. *Human-Computer Interraction*) prístup zakladá na vytvorení príjemného používateľského rozhrania. Ide o tendenciu zefektívniť prácu učiteľa vytvorením čo najvhodnejšieho prostredia pre tvorbu kurzov. Okrem dvoch menovaných prístupov existujú aj hybridné prístupy, ktoré kombinujú ich vlastnosti a riešenie vidia vo vzájomnej kooperácii autora kurzu a počítača. V nasledujúcich častiach sa budeme venovať ich opisu v kontexte vybraných ASV.

3.1.1 AHA!

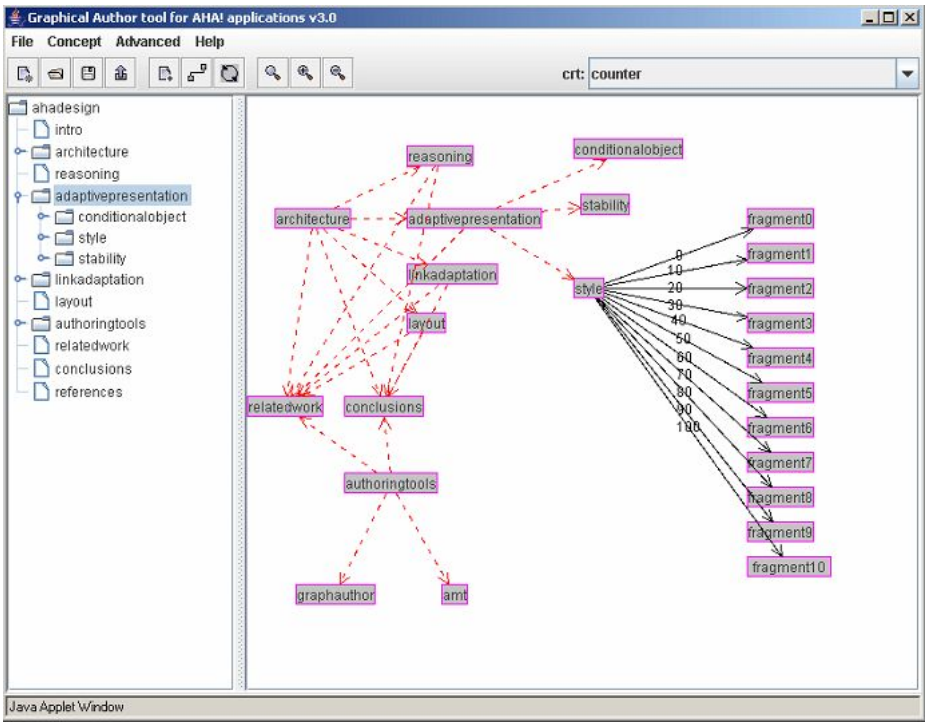
Systém AHA! má vyvinutých hneď niekoľko autorských nástrojov: *Form editor*, *Concept Editor* a *Graph Author*. *Concept Editor* je jeden z najstarších a umožňuje upravovať aspekty doménového modelu a modelu prispôsobovania bez nutnosti poznania syntaxe XML, pomocou ktorej sú modely v systéme AHA! definované. Z pohľadu napĺňania obsahu kurzu a tvorby jeho štruktúry je zaujímavý posledný menovaný nástroj.

Graph Author je jednoduchá aplikácia s grafickým používateľským rozhraním (pozri obrázok 3-1), kde môže tvorca kurzu v príjemnom používateľskom prostredí navrhnúť štruktúru kurzu: definovať koncepty a vzťahy medzi nimi. Nástroj tiež umožňuje automatizáciu tvorby modelu prispôsobovania, ktorý odvodí z navrhnutej štruktúry. Automatizácia však spočíva iba vo vytvorení

4 Pracovné dielne sa poriadajú pod názvom „International Workshop on Authoring of Adaptive and Adaptable Hypermedia“, viac informácií na webovom sídle http://www.win.tue.nl/~acristea/A3H/

generických pravidiel prispôsobovania na základe typov relácií, ktoré je vo väčšine prípadov potrebné manuálne upraviť.

Nástroj *Form editor* slúži na vytváranie formulárov pre koncových používateľov a s podporou tvorby kurzov nesúvisí.



Obrázok 3-1: Autorský nástroj Graph Author (De Bra, et al., 2007).

3.1.2 MOT

Pojem *adaptívnej* podpory tvorby kurzov zavádzajú Cristea a Aroyo v (Cristea, Aroyo, 2002). V práci sumarizujú skúsenosti nadobudnuté rokmi výskumu a prezentujú metodiku systematizovaného a organizovaného prístupu k napĺňaniu kurzov, ktorá má za cieľ zlepšiť motiváciu autorov kurzov a zvýšiť popularitu ASV u učiteľov. Metodika vychádza z práce v MyET a AIS, dvoch nezávislých a intenzívne používaných adaptívnych systémov pre vzdelávanie. Jej jadrom je konceptovo-založená tvorba kurzov, ktorej princíp spočíva v rozdelení procesu napĺňania kurzu do viacerých etáp:

1. vytvorenie konceptov, ich hierarchizácia,
2. definovanie hlavných a vedľajších atribútov pre koncepty,
3. naplnenie týchto atribútov,
4. definovanie predpokladov, alternatív,
5. definovanie mapy kurzu (členenie do lekcií a pod.),
6. pridanie čŕt prispôsobovania vzhľadom na použitú techniku prispôsobovania.

Prezentovaná metodika v skutočnosti proces tvorby neautomatizuje, skôr by sa dalo hovoriť o „kváziautomatizácii“, keď jasná postupnosť krokov odbúrava zmätok prítomný počas tvorby. Celá práca na kurze sa vykonáva manuálne. Autori sa spoliehajú na postupne sa zväčšujúcu bázu znalostí (pridávaním ďalších a ďalších konceptov), ktorú bude tvoriť rastúca ontológia, a nezameriavajú sa na softvérovo-riadenú podporu tvorby. Metóda tak nesúvisí s odbremenením autorov od ručného napĺňania, iba im túto prácu sprehľadňuje a v tomto zmysle čiastočne zefektívňuje.

Zaujímavý prístup je prezentovaný v (Cristea, Okamoto, 2001), kde autori vyvinuli pre systém MyET jednoduché autorské prostredie pre kolaboratívnu tvorbu kurzov. Dôraz sa kladie na znovupoužiteľnosť už vytvorených materiálov. Systém asistuje autorom učebných materiálov pri mapovaní konceptov, t.j. pri ich vzájomnom usporadúvaní a hierarchizácii. Podpora tvorby spočíva v generovaní najpravdepodobnejších kandidátov na prepojenie (systém ponúkne zoznam kľúčových slov a názvov konceptov) a učiteľ z nich v grafickom používateľskom prostredí vyberá tie najvhodnejšie. Aj keď systém automatizuje určitú časť práce, inicializačné nastavenie podobnosti medzi konceptami je na tvorcovi kurzu. Vyžaduje sa ručná anotácia väzieb medzi konceptami (určenie podobnosti) a manuálne zadávanie kľúčových slov. ASV v tomto prípade pracuje so študijným materiálom v rôznych formách (text, video, zvuk) a štýloch (kolaboratívny prístup), preto je plne automatické zistenie podobnosti komplikované. Zaujímavou vlastnosťou systému je, že vzťahy medzi konceptami sa dynamicky upravujú v čase používania kurzu študentmi, takže výsledná podoba kurzu (po niekoľkých semestroch používania) môže byť odlišná od tej na počiatku definovanej autorom.

Otázku, či je poloautomatická tvorba v ASV vôbec možná, si kladie autorka v (Cristea, 2004). Predmetom skúmania je automatizácia procesu tvorby v adaptívnom výučbovom systéme MOT. MOT vychádza z referenčného modelu LAOS (Cristea, De Mooij, 2003b), prostriedkom podpory tvorby je preto dôraz na odstránenie rutinných úkonov tvorcu kurzu pri návrhu jednotlivých vrstiev modelu. Princípy automatizácie sú zahrnuté v dvoch bodoch:

- tvorba väzieb v doménovom modeli,
- transformácie medzi jednotlivými modelmi AEH.

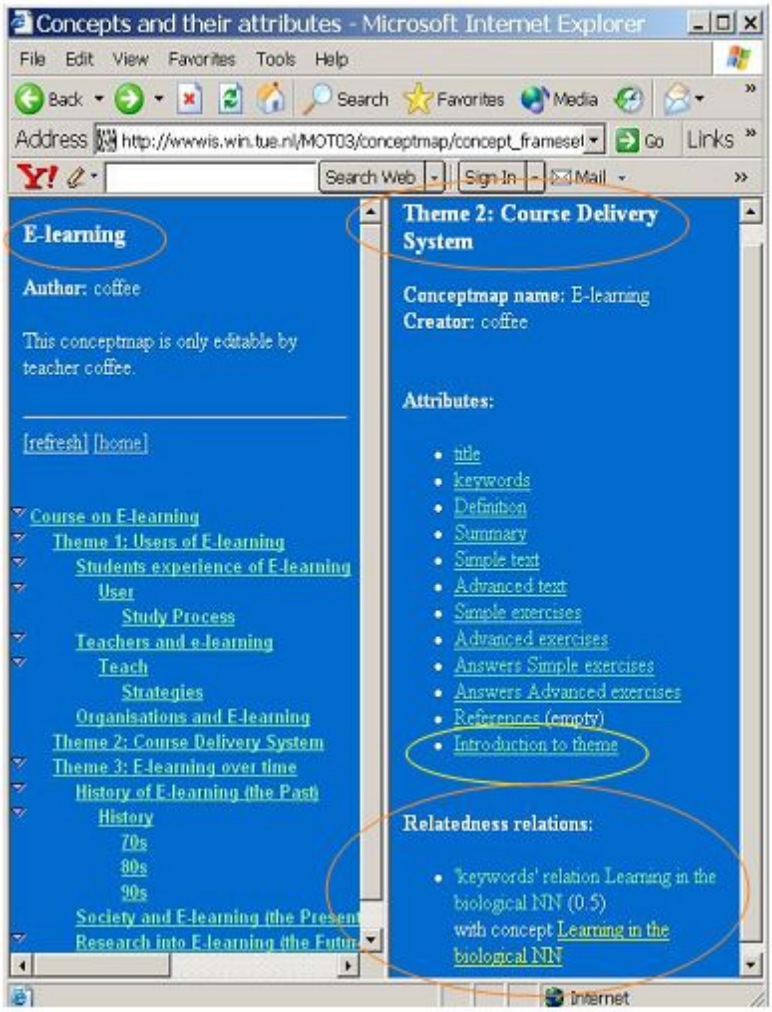
Transformácie medzi modelmi sa týkajú najmä transformácie z vytvoreného doménového modelu. Hierarchizácia doménovej oblasti je na takej úrovni, že je možné odvodiť niektoré informácie o ostatných modeloch (model cieľov a obmedzení, model prispôsobovania, atď.). Pravidlá odvodzovania sú špecifické pre architektúru ASV a referenčný model LAOS a je nad rámec tejto práce sa nimi zaoberať. Čo je v tomto prípade dôležité, je tvorba väzieb v doménovom modeli (Cristea, de Mooij, 2003c).

Poloautomatické generovanie väzieb v doménovom modeli zmierňuje dopady rutinnej hierarchizácie priestoru konceptov používateľom. Vygenerované

väzby umožňujú odkazovanie sa na sémanticky podobný učebný materiál a zlepšujú mieru prispôsobovania študentovi (pozri obrázok 3-2). Tvorba odkazov (prepojení) sa realizuje na základe výpočtu miery podobnosti medzi konceptami. Miera podobnosti sa zisťuje z výskytu kľúčových slov (ktoré sú na začiatku definované autorom kurzu) v jednotlivých doménových atribútoch. Doménovými atribútmi v systéme MOT sú napr. názov (*title*), opis (*description*), príklad (*example*), samotný text (*text*) konceptu, atď. Výsledná podobnosť medzi konceptami *i* a *j* je určená vzťahom

$$correspondence(i,j) = \frac{\sum_{h=1}^{A_j}\sum_{s=1}^{K_i} \frac{importance(a_j[h]) * occ_{ij}(k_i[s], a_j[h])}{maxocc_{ij}(k_i[s], a_j[h])}}{K_i * A_j} \qquad (1)$$

kde $a_j[]$ je množina doménových atribútov konceptu *j*, A_j je ich počet, $k_i[]$ je množina kľúčových slov konceptu *i* a K_i ich počet. Funkcia *importance(a)* určuje mieru dôležitosti (relevanciu) doménového atribútu *a*, funkcia $occ_{ij}(k, a)$ určuje počet výskytov kľúčového slova *k* konceptu *i* v doménovom atribúte *a* konceptu *j*



Obrázok 3-2: Odporúčanie podobných konceptov (Cristea, 2004).

a $maxocc_{ij}(k, a)$ je maximálny možný výskyt kľúčového slova k v doménovom atribúte a (čo je vlastne počet slov doménového atribútu).

Na základe výpočtu sa autorovi kurzu ponúknu potenciálne prepojenia medzi konceptami, z ktorých si môže vybrať také, ktoré sú podľa jeho úsudku najvhodnejšie (pozri obrázok 3-3). Tie sa nakoniec použijú pri prispôsobovaní v procese odporúčania študijného materiálu.

Current concept: Theorem of Batch Perceptron Convergence

The following relatedness weights are found:

Concept name	Relation Type	Relation weight
NN Introduction [add]	text	0.217
Learning [add]	text	0.245
DISCRETE-NEURON PERCEPTRONS [add]	keywords	0.515
DISCRETE-NEURON PERCEPTRONS [add]	text	0.308
DISCRETE-NEURON PERCEPTRONS [add]	title	0.539
Introduction [add]	keywords	0.471
Introduction [add]	text	0.394
Introduction [add]	title	0.204
One-layer Discrete Perceptrons [add]	keywords	0.354
One-layer Discrete Perceptrons [add]	title	0.289
NN Introduction [add]	text	0.217
Learning in the ANN [add]	text	0.245
Discrete -Neuron Perceptrons [add]	keywords	0.515
Discrete -Neuron Perceptrons [add]	text	0.308

Obrázok 3-3: Miera podobnosti konceptov (Cristea, 2004).

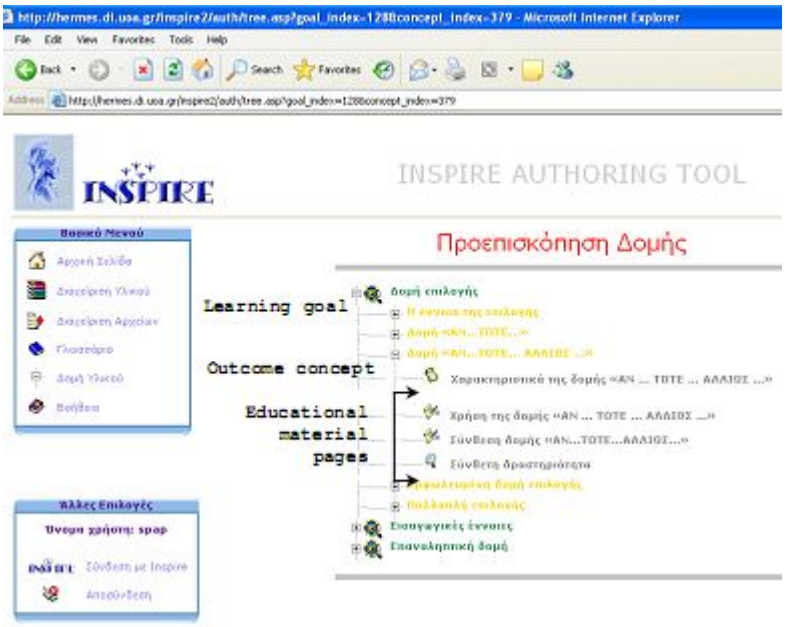
Predstavená metóda zjednodušuje prácu a stavia podporu tvorby kurzu na vyššiu úroveň. Automatizácia procesov stále nie je ideálna, nakoľko autori kurzov musia ručne vypĺňať metadáta o konceptoch (doménové atribúty).

3.1.3 INSPIRE

Podpore tvorby kurzov sa venujú aj autori systému INSPIRE, ktorí sa vybrali cestou prístupu HCI. Autorský nástroj obsahuje veľmi príjemné používateľské rozhranie (pozri obrázok 3-4), absentuje však akákoľvek automatizácia procesu tvorby kurzov.

3.1.4 WebEx (NavEx)

Metóda automatizovaného indexovania obsahu je súčasťou prístupu k podpore tvorby kurzov v systéme WebEx (Sosnovsky, Brusilovsky, Yudelson, 2004). Doménovou oblasťou nástroja je výučba programovania v jazyku C pomocou príkladov. Automatizované indexovanie pozostáva z dvoch stupňov:



Obrázok 3-4: Používateľské prostredie INSPIRE (Grigoriadou,Papanikolaou, 2006).

- parsovanie obsahu (angl. *content parsing*),
- identifikácia predpokladov a záverov (angl. *prerequisite/outcome identification*).

Parsovanie obsahu spočíva v lexikálnej analýze študijných materiálov (príkladov, otázok, stránok s obsahom) a následnom generovaní konceptov, ktorými nie sú iba jednoduché kľúčové slová z domény programovania, ale opisy konkrétnych syntaktických štruktúr – napr. *include, void, main_func, decl_var, long, decl_var, assign, ne_expr, pre_inc, while, compl_printf.*

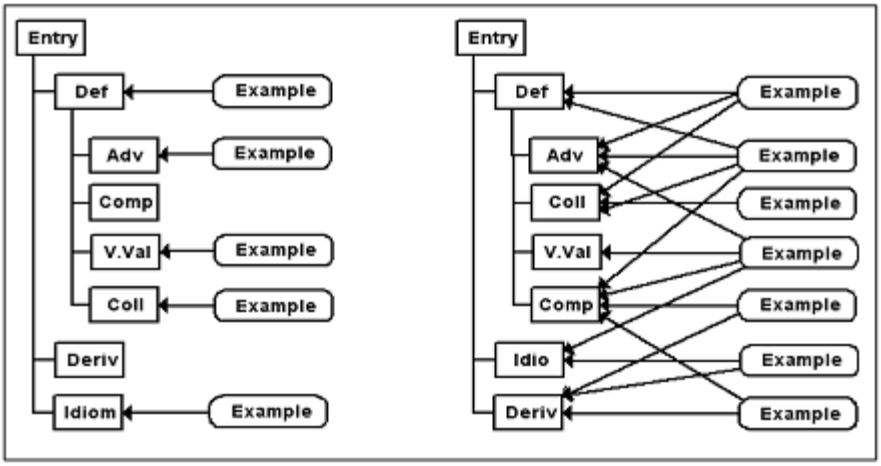
Po tomto kroku prichádza na rad generovanie relácií medzi konceptami v podobe identifikácie predpokladov a záverov (vyplývajúcich zo štúdia daného konceptu). Autor kurzu na začiatku definuje kurz ako postupnosť lekcií v študijných materiáloch (ale nie v konceptoch). Potom sa informácie o predpokladoch, resp. záveroch odvodzujú nasledovne:

- pre každú lekciu sú predpokladmi všetky koncepty, ktoré už boli identifikované v lekciách nachádzajúcich sa v kurze *pred* danou lekciou,
- pre každú lekciu sú závermi štúdia koncepty, ktoré sa v kurze v danej lekcii objavujú *prvýkrát*.

Výsledkom odvodzovania je kompletný model hyperpriestoru s navigáciou založenou na indexovaní stránok. Keďže pri indexovaní je potrebná interakcia učiteľa (definovanie postupnosti lekcií), metóda je v podstate iba rozdelením „záťaže“ tvorby kurzu medzi používateľa a softvér. Je ukážkou hybridného prístupu k podpore procesu tvorby kurzov.

3.1.5 ELDIT

Aj v prípade systému ELDIT – nemecko-talianského výkladového elektronického slovníka – vidia autori možnosť zjednodušenia naplňovania obsahu v hybridnom prístupe k podpore tvorby, keď vzájomne kooperujú ľudia a umelé agenty. Na začiatku bolo v slovníku v spolupráci s odborníkmi naplnené jadro slovníka (tzv. *core content*), ktoré zahŕňa definície, gramatické pravidlá a frazeológiu jednotlivých slov, ktoré sú zároveň konceptami doménového modelu. Problém nastal s vytváraním výučbového obsahu (tzv. *educational content*) – ukážky, príklady, vysvetlivky, obrázky k slovám z jadra – pretože doplniť takýto obsah pre približne 7000 slov nie je triviálnou úlohou. Aby bola výučba čo najefektívnejšia, je potrebné mať pre každé slovo (koncept) čo najväčší možný počet ukážok. Riešenie zakladá na myšlienke znovupoužitia výučbového materiálu: autori navrhli metódu automatickej extrakcie príkladov, ktoré už boli použité na iných miestach v slovníku. Príklady sa k slovám v systéme priraďujú pomocou jednoduchých relácií, ktoré tvoria vzťahy medzi konceptami a materiálom. Práve objavenie týchto vzťahov je základom použitej metódy (pozri obrázok 3-5).

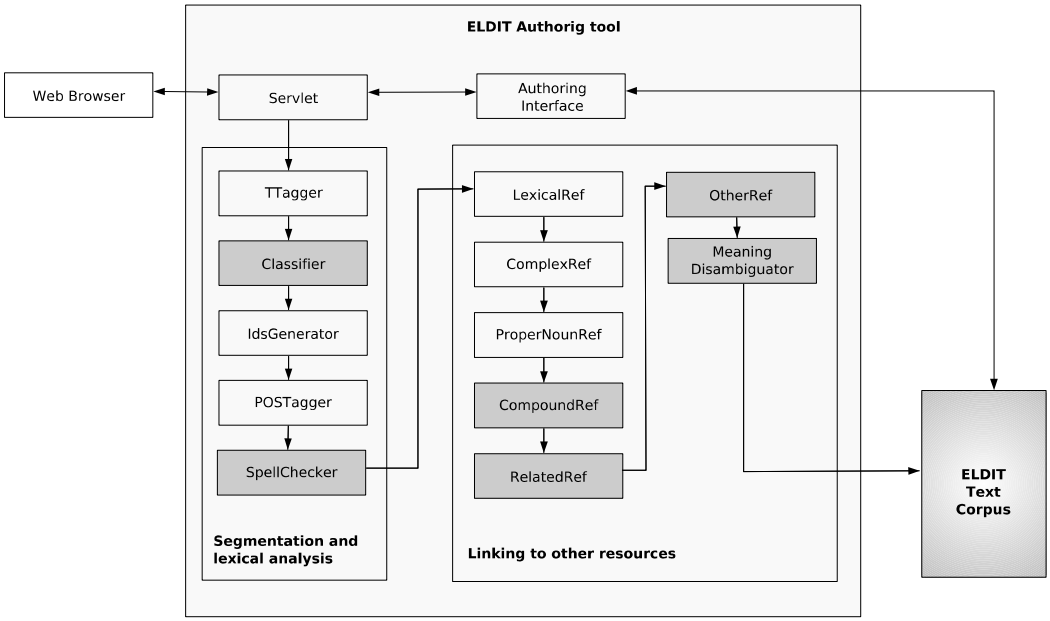


Obrázok 3-5: Podpora tvorby obsahu v projekte ELDIT – vzťahy medzi ilustračnými príkladmi (Example) a konceptmi (naľavo) sú generované použitím techník spracovávania textu (Brusilovsky, Knapp, Gamper, 2006).

Metóda pozostáva z vykonania dvoch základných procesov:

1. segmentácie a lexikálnej analýzy textu a
2. tvorby relácií (odkazov) medzi konceptami a príkladmi.

Proces segmentácie a lexikálnej analýzy je, keďže ide o slovník, obzvlášť citlivý na problém s ohýbaním a tvaroslovím spracovávaných slov. Autori vyvinuli inteligentný mechanizmus spracovania textu pomocou techník získavania znalostí a spracovania prirodzeného jazyka (angl. Natural Language Processing, NLP; pozri obrázok 3-6).



Obrázok 3-6: Algoritmus spracovania textu v systéme ELDIT (Brusilovsky, Knapp, Gamper, 2006).

Nástroj postupne aplikuje techniky značkovania, klasifikácie a označenia slovných druhov vo vetách textu. Po kontrole pravopisu pošle výstupy do etapy tvorby relácií medzi konceptmi a príkladmi. V tejto časti je využitých viacero (aj existujúcich) nástrojov. Základné referencie sú vytvorené na lexikálnej úrovni pomocou komponentu *LexicalRef*. Príklady na základe extrahovaných slov prepojí s existujúcimi konceptmi na základe podobnosti (koncept ako slovníkový záznam je reprezentovaný slovom). Samostatne sú spracované vlastné podstatné mená (*ProperRef*). Na odhalenie synoným sa využíva projekt EuroWordNet[5] (*OtherRef*), ktorý zastrešuje databázu vybraných európskych jazykov. Pre redukciu významovej viacznačnosti je použitá metóda kontextového vektora na určenie podobnosti medzi jednotlivými príkladmi a konceptami (*MeaningDisambiguator*). Na základe výsledkov oboch procesov sa odvodí výsledná štruktúra vzťahov, ktorá sa predloží používateľovi na posúdenie. Tento krok je nutný kvôli nie vždy vyhovujúcej úrovni strojového spracovania prirodzeného jazyka. Neodhalená významová viacznačnosť niektorých slov (a tiež ďalšie „typické“ problémy NLP) musia byť preto riešené na úrovni interakcie používateľa s počítačom (Knapp, Gamper, Brusilovsky, 2004).

Úspešnosť systému ELDIT bola vyhodnotená ako pomer správne priradených príkladov k nesprávne priradeným príkladom. Úspešnosť metódy kolíše v závislosti od zložitosti priraďovaných príkladov. Pre jednoduché je vysoká, pre zložité (napr. zložené nemecké slová) nižšia. Uvedený prístup pokrýva spracovanie textu nemeckého a talianskeho jazyka je špecifický pre domény, ktoré kladú dôraz na detaily v prirodzenom jazyku.

5 http://www.illc.uva.nl/EuroWordNet/

3.2 Zhrnutie podpory tvorby kurzov vo vybraných ASV

V tabuľke 3-1 uvádzame sumár podpory tvorby kurzov vo vybraných ASV. Druhý a tretí stĺpec vyjadrujú mieru podielu jednotlivých prístupov (AI, HCI) v procese tvorby kurzu, posledný štvrtý stĺpec obsahuje informáciu o tom, či sa prístup využívaný ASV dá považovať za hybridný.

Tabuľka 3-1:

ASV systém	Úroveň prístupu k podpore tvorby		
	AI	HCI	hybridný
AHA!	poloautomatické generovanie pravidiel prispôsobovania	autorský nástroj s grafickým prostredím	✓
MyET / MOT	poloautomatické generovanie modelu prispôsobovania a modelu cieľov	HTML formuláre	✓
INSPIRE	–	HTML formuláre s grafickými prvkami	×
WebEx, NavEx	poloautomatické generovanie časti doménového modelu	HTML formuláre	✓
ELDIT	dodatočná hierarchizácia doménového modelu	HTML formuláre	✓

Každý z analyzovaných ASV využíva buď AI alebo HCI prístup. U väčšiny analyzovaných systémov sa dá hovoriť o implementácii hybridného prístupu. Priestor na vylepšenie sa naskytá v oblasti HCI prístupu, kde iba jeden systém obsahoval vhodný autorský nástroj použiteľný pre autorov – neinformatikov. V prípade AI prístupu žiaden systém neautomatizoval tvorbu kurzu do takej miery, aby používateľovi výrazným spôsobom skrátil čas ňou strávený. Problém s automatizáciou tvorby spočíva v tvorbe inteligentného obsahu kurzu pretože, ako tvrdí Brusilovsky, „Sila inteligentného obsahu je ukrytá v znalostiach o každom jeho fragmente.“ (Brusilovsky, et al., 2005). V tomto momente sa naskytá otázka, akým spôsobom je možné takýto „inteligentný obsah“ vytvoriť. Uvedený citát naznačuje, že odpoveďou budú metódy a techniky objavovania znalostí. V nasledujúcej kapitole sa venujeme opisu niektorých z nich.

4 Objavovanie znalostí v učebných textoch

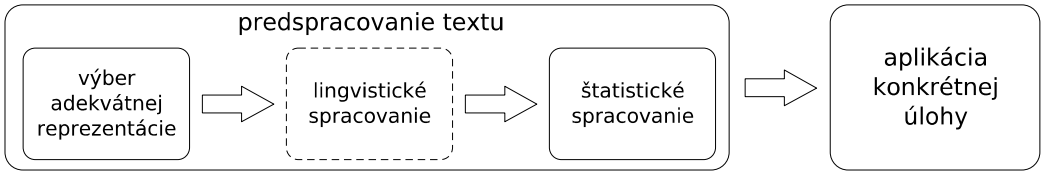
Východiskom pre získanie znalostí o výučbovom kurze je jeho obsah. V učebných textoch sú „ľudskou rečou“ zakódované informácie nielen o samotných témach a pojmoch z vyučovanej oblasti, ale tiež o ich prepojeniach a vzájomných súvislostiach. Ak našim cieľom má byť vytvorenie elektronickej podoby kurzu, musíme tieto informácie vedieť dekódovať do podoby jednotlivých modelov ASV. V tejto kapitole sa budeme najprv venovať metódam a technikám spracovania (učebného) textu a potom, keďže sa v práci zameriavame na doménový model, analyzujeme vybrané grafové algoritmy, ktoré bude možné v procese automatizácie tvorby kurzu využiť.

4.1 Spracovanie textu

Medzi tradičné funkcie spracovania textu (alebo tiež dolovania v texte – angl. *text mining*) patrí:

- kategorizácia textu,
- zhlukovanie textu,
- extrakcia entít/pojmov,
- tvorba taxonómií,
- dolovanie názorov,
- sumarizácia dokumentu.

Priebeh typickej úlohy spracovania textu môžeme zjednodušene vyjadriť schémou zobrazenou na obrázku 4-1. Jednotlivé segmenty predspracovania sú opísané ďalej v tejto časti. Pred tým, ako sa zrealizuje samotná aplikácia konkrétnej úlohy, predchádza jej určité predspracovanie textu pozostávajúce spravidla z výberu adekvátnej reprezentácie, lingvistického spracovania (prirodzeného jazyka) a štatistického spracovania.



Obrázok 4-1: Spracovanie textu (prerušovanou čiarou je znázornený nepovinný krok).

Pre väčšinu uvedených úloh existuje ďalší spoločný činiteľ: určovanie *podobnosti*. Či už sa jedná o určenie podobnosti medzi dopytom a dopytovaným dokumentom, podobnosti medzi dokumentmi navzájom alebo podobnosti medzi jednotlivými časťami dokumentov, vždy sa snažíme jednotlivé objekty porovnať a určiť mieru ich podobnosti. V kontexte automatizácie tvorby kurzu je podobnosť nemenej dôležitá – pri objavovaní vzťahov v spracovávaných učebných textoch je potrebné ohodnotiť mieru vzájomnej súvislosti častí ich obsahu. Preto si pri opise konkrétnych možností spracovania textu budeme pozorne všímať práve možnosti výpočtu podobnosti.

4.1.1 Možnosti reprezentácie dokumentov

Existuje viacero možností ako reprezentovať dokumenty pre potreby ich ďalšieho spracovania. Reprezentácia je spravidla prispôsobená cieľu získavania znalostí. Medzi základné modely reprezentácie patria (Kuropka, 2004):

- množinové modely,
- algebraické modely,
- pravdepodobnostné modely.

Množinové modely

Množinové modely reprezentujú dokumenty ako množinu slov, resp. slovných spojení (fráz). Podobnosti medzi dokumentmi sú odvodené od množinových operácií nad týmito množinami. Bežným príkladom množinového modelu je *štandardný booleovský model*, kde sa pri výpočtoch podobnosti berie do úvahy iba fakt, či dokument obsahuje, resp. neobsahuje porovnávané slová.

Algebraické modely

Algebraické modely reprezentujú dokumenty najčastejšie vektormi, maticami alebo usporiadanými n-ticami. Podobnosti medzi dokumentmi sú vyjadrené skalárnou veličinou. Príkladmi algebraických modelov sú *vektorový model*, *generalizovaný vektorový model*, *témovo-založený vektorový model* alebo *model latentnej sémantickej analýzy*.

Vektorový model je reprezentáciou dokumentu ako vektora slov (zvyčajne v základnom – slovníkovom – tvare), ktoré sa v ňom nachádzajú. Ohodnotením slova je váha, ktorá vyjadruje mieru jeho dôležitosti v danom dokumente. Spôsobov výpočtu váh je viacero, touto témou sa zaoberáme neskôr v tejto časti. Generalizovaný a témovo-založený vektorový model sú rozšírením tohto modelu.

V prípade modelu latentnej sémantickej analýzy sú vstupné dáta reprezentované tzv. maticou výskytov, v ktorej riadky prislúchajú dokumentom a stĺpce slovám. Prvkami matice sú váhy. Bližší opis tejto techniky uvádzame v časti 4.1.3.

Pravdepodobnostné modely

Pravdepodobnostné modely sa pozerajú na spracovanie dokumentov ako na odvodzovanie na základe pravdepodobnosti. Podobnosti sú určené mierou pravdepodobnosti, že dokument je pre daný dopyt relevantný. Veľmi často sa používa známy Bayesov klasifikátor. Príkladom pravdepodobnostného modelu je *model pravdepodobnostnej relevancie BM25* alebo tzv. *jazykové modely*.

Príklad výpočtu váh

V súčasnosti najrozšírenejšou technikou výpočtu váh pre vektorovú reprezentáciu dokumentu je technika *tf-idf* (z angl. term frequency – inverse document frequency). Váha slova nie je daná len frekvenciou výskytu slova v dokumente, ale tiež jeho frekvenciou v celom korpuse (Salton, Wong, Yang, 1975). Cieľom je znížiť váhu tých slov, ktoré sa v korpuse vyskytujú až príliš často, čo degraduje ich dôležitosť oproti ostatným slovám. Váha slova t_i v dokumente d_j je je daná vzťahom:

$$w(i,j) = tf(i,j) \cdot idf(i) \tag{2}$$

kde *tf(i,j)* je frekvencia slova t_i v dokumente d_j a *idf(i)* je inverzná frekvencia slova t_i v korpuse:

$$tf(i,j) = \frac{n(i,j)}{\sum_{t_k \in D} n(k,j)} \tag{3}$$

$$idf(i) = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|} \tag{4}$$

kde *n(i,j)* je počet výskytov slova t_i v dokumente d_j, a D je množina všetkých dokumentov (korpus).

V súčasnosti existujú viaceré modifikácie tejto metriky (Salton, Buckley, 1988), ale aj iné prístupy k výpočtu váh (Allan, et. Al, 1995; Amati, van Rijsbergen, 2002), ktoré berú do úvahy špecifiká súvisiace s doménovou oblasťou spracovania.

4.1.2 Lingvistické spracovanie prirodzeného jazyka

V procese objavovania znalostí v texte je potrebné spracovať informácie, ktoré sú vyjadrené prirodzeným jazykom človeka, do podoby zrozumiteľnej pre počítač. Tu sa však stretávame s problémom nejednoznačností, ktoré tvoria najväčšiu prekážku pri analýze prirodzeného jazyka (Páleš, 1994). Príklady niektorých z nich:

- tvarová homonymia – jeden tvar patrí viacerým slovným druhom,
- lexikálna homonymia – jeden tvar označuje významovo rozličné slová,
- polysémia – slovo má viacero významov,
- synonymia – viac slov má podobný alebo rovnaký význam,
- syntaktická viacznačnosť – jedna fráza sa dá interpretovať viacerými spôsobmi,

Nezanedbateľným problémom je tiež použitie ustálených slovných spojení alebo frazeologizmov, kde vzájomný význam skupiny slov je zvyčajne iný ako význam jednotlivých slov zvlášť.

Riešenia uvedených problémov sa snažíme hľadať pomocou tzv. lingvistických metód a techník spracovania prirodzeného jazyka (NLP). Tieto metódy sa zameriavajú najmä na lingvistický aspekt spracovania textu. Ich úlohou je identifikácia lexikálnych, morfologických, syntaktických, sémantických (a ďalších) vlastností spracovávaného textu, ktoré napomôže v procese získania znalostí obsiahnutých v texte.

Medzi najpoužívanejšie techniky patria *lematizácia*, resp. *stemming*. Ide o procesy priraďovania tzv. lem, resp. stemov[6] k ich slovným tvarom (tokenom) v texte. Kým pre niektoré jazyky je tento proces ľahko algoritmizovateľný (napr. anglický jazyk využíva Porterov algoritmus (Porter, 1980)), pre flexívne[7] jazyky, medzi ktoré patrí aj slovenčina, je to netriviálna úloha (berúc do úvahy množstvo existujúcich výnimiek). Problematike lematizácie v slovenskom jazyku sa špeciálne venujeme v nasledujúcej časti.

Ďalšími technikami sú napríklad morfologická dekompozícia textu, značkovanie vetných členov, rozpoznávanie typov viet, kontextová analýza a pod. Tieto techniky sémanticky obohacujú dáta, v ktorých sa doluje, avšak na rozdiel od prístupov štatistického spracovania vyžadujú dodatočnú réžiu, keďže takmer vo všetkých prípadoch je potrebný istý podiel práce človeka na manuálnej anotácií zdrojových textov.

Lematizácia v slovenskom jazyku

Slovenčina je typickým reprezentantom jazyka, ktorého bohatá morfológia spôsobuje ťažkosti pri strojovom spracovaní. Riešenia fungujúce pre jazyky, ktoré majú menej pádov, časov, stupňov alebo iných gramatických kategórií, sú pre slovenčinu nepoužiteľné.

Pre spracovanie slovenského jazyka dlhšiu dobu prebiehala snaha o vytvorenie sofistikovaného algoritmického riešenia v projekte SAPFO (Páleš, 1994), ktorý však nesplnil očakávania. Ukázalo sa, že počet výnimiek v slovenskom jazyku je taký veľký, že akceptovateľným riešením bude použitie konštantnej databázy, v ktorej budú uchované všetky základné tvary slovenských slov (a pravidlá odvodzovania pre každý základný tvar zvlášť). V súčasnej dobe už práca s takouto databázou, ktorej veľkosť predstavuje zopár desiatok MB, nie je problém. Takýto lematizátor bol vytvorený na JÚĽŠ SAV v Bratislave (Garabík, 2006). Obsahuje slová z Krátkeho slovníka slovenského jazyka v jeho 3. vydaní a ďalšie frekventované slová zo slovenského národného korpusu[8], čím je možné vyhľadávať

6 *Lema* – základný (slovníkový) tvar slova pri ohybných slovných druhoch. Pre podstatné mená nominatív singuláru, pre slovesá neurčitok a pod. Pr.: rybníkmi → rybník
Stem – koreň slova. Pr.: rybníkmi → ryb-

7 t.j. také, keď odvodzované slovné tvary vznikajú ohýbaním (koncoviek).

8 http://korpus.juls.savba.sk/

tvary pre 56269 lem (a naopak). Lematizátor v súčasnosti nepodporuje prácu s negáciou slovies (predpona ne-) a stupňovaním prídavných mien a prísloviek (3. stupeň a predpona naj-). Výhodou tohto riešenia je, že aj napriek dvom výnimkám disponuje možnosťou lematizácie pre všetky slovné druhy. Naopak, nevýhodou je, že neobsahuje cudzie slová a termíny špecifické pre doménové oblasti rôznych vedných odborov, čím je vysoká úspešnosť jeho použitia obmedzená len na bežnú slovenčinu.

Alternatívou je softvérový nástroj *Tvaroslovník*, ktorý sa vracia k pôvodnej paradigme a využíva algoritmus generovania lem na základe vzorov (Krajči, Novotný, 2006). Metóda vychádza z pozorovania, že na ohýbanie slovenského slova má najväčší vplyv jeho koncovka. Aj keď sa toto riešenie týka len lematizácie podstatných mien, nie je obmedzené počtom spracovateľných slov. V prípade výskytu nového slova je veľká pravdepodobnosť, že „zapadne“ do už existujúceho vzoru a ak nie, tak stačí vytvoriť nový vzor pre celú triedu podobných slov.

4.1.3 Štatistické metódy

Pri štatistickom spracovaní textu sa využívajú stochastické, pravdepodobnostné alebo štatistické metódy, ktoré sa zameriavajú na štatistické atribúty slov a nie lingvistický aspekt textu. Štandardným problémom je už skôr spomínané určovanie podobnosti dokumentov.

Najjednoduchším spôsobom je použitie metriky *kosínusovej vzdialenosti*. Pri použití tejto metriky vychádzame z predpokladu vektorovej reprezentácie dokumentu. Podobnosť je vyjadrená kosínusom uhla medzi vektormi x a y:

$$Cosim(x, y) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \qquad (5)$$

kde *Cosim(x, y)* je výsledná kosínusová vzdialenosť a x_i, resp y_i je i-tý prvok vektora x, resp. y. Obor hodnôt uvedenej funkcie sa pohybuje v intervale <0; 1>, kde 0 znamená, že vektory sú úplne odlišné a 1, že vektory sú totožné. Nevýhodou použitia tejto metriky je, že neidentifikuje žiaden z problémov adresovaných v úvode časti .

Na odhalenie skrytých vzťahov medzi slovami sa používa metóda *latentnej sémantickej analýzy (LSA)*. LSA využíva algebraický model reprezentácie vstupných dát – maticu slov a dokumentov (tzv. maticu výskytov), ktorá má tú vlastnosť, že je riedka[9]. Proces latentnej sémantickej analýzy spočíva vo využití definovaných maticových transformácií nad maticou výskytov, ktorými sa odhalia koncepty určujúce skryté väzby medzi slovami a dokumentmi. Každé slovo je v relácii s konceptom a koncept je v relácii s dokumentom. Odhalené skryté väzby sa použijú na výpočet podobnosti medzi dokumentmi alebo medzi slovami, čím do určitej miery rieši problém polysémie a synonymie.

9 Riedka matica je matica, ktorej väčšina prvkov je rovná 0.

Medzi ďalšie štatistické metódy môžeme zaradiť rôzne aplikácie neurónových sietí alebo techniky kategorizácie, či zhlukovania dokumentov. Podrobný opis týchto a ďalších riešení je nad rámec tejto práce.

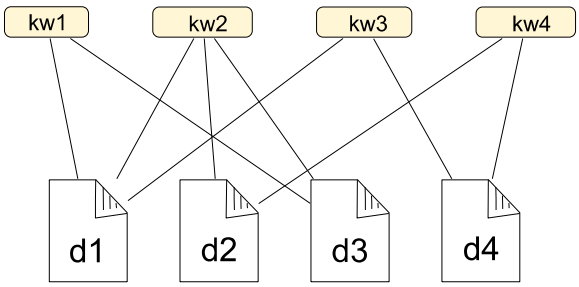
4.2 Vybrané grafové algoritmy

Pri objavovaní znalostí v texte môže nastať situácia, že získané dáta (na určitom medzistupni spracovania) môžeme reprezentovať pomocou grafu. Aplikácia vhodného algoritmu tak môže zlepšiť úspešnosť celej metódy.

Z podoby doménového modelu (pozri 2.1) je zrejmé, že takáto situácia nastane aj v procese tvorby kurzu. Je preto užitočné preskúmať možnosti použitia grafových algoritmov, ktoré sú v súčasnosti využívané na poli získavania znalostí. Nakoľko je problematika grafových algoritmov veľmi rozsiahla, vyberáme len niektoré z nich.

4.2.1 Šírenie aktivácie

Metóda šírenia aktivácie sa využíva pri prehľadávaní asociatívnych (alebo tiež kontextových) sietí (Ceglovsky, Coburn, Cuadrado, 2003). Šírenie aktivácie vychádza z poznatkov odboru kognitívnej psychológie, kde slúži na modelovanie procesov v ľudskej pamäti. Aplikácia v oblasti získavania znalostí spočíva v myšlienke reprezentácie vstupných dokumentov a ich kľúčových slov ako vrcholov grafu (pozri obrázok 4-2).



*Obrázok 4-2: Model šírenia aktivácie v doméne získavania znalostí (kw*X *– kľúčové slová, d*Y *– dokumenty).*

Nad takýmto grafom aplikujeme jednoduchý algoritmus rekurzívneho šírenia energie z počiatočného vrcholu do celého grafu. Počiatočný vrchol nabijeme energiou E a tú pomocou definovanej funkcie prešírime do susedných vrcholov. Na konci algoritmu je miera podobnosti jednotlivých dokumentov je vyjadrená pomerom medzi energiami v ostatných vrcholoch k energii v počiatočnom vrchole. Princíp šírenia môžeme vyjadriť pseudokódom:

```
function spread(E) {
      this.activation += E;
      E' = E / this.degree;
      if(E' > threshold) {
            for(this.neighbors : neighbor) {
                  neighbor.spread(E');
            }
      }
}
```

4.2.2 PageRank

Asi jedným z najznámejších vyhľadávacích algoritmov je PageRank (Brin, Page, 1998). Doménou jeho použitia je webový hyperpriestor, kde slúži na ohodnotenie dôležitosti jednotlivých webových stránok. Algoritmus je založený na markovských procesoch. Samotná metrika dôležitosti[10] stránky (jej „PageRank“) je v podstate vyjadrením pravdepodobnosti, že človek náhodne klikajúci na webové odkazy sa dostane na danú webovú stránku (známe tiež ako model náhodného surfera).

Na určenie dôležitosti stránky na webe (vrcholu v grafe) sa používa výpočet založený na vzťahu[11]:

$$PageRank(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PageRank(p_j)}{L(p_j)} \qquad (6)$$

kde p_i je webová stránka, $M(p_i)$ je množina stránok odkazujúcich na stránku p_i, $L(p_j)$ počet odkazov zo stránky p_j a N je celkový počet stránok. Koeficient d je tzv. faktor útlmu, ktorý vyjadruje pravdepodobnosť, že náhodný surfer pri surfovaní neprestane v klikaní na webové odkazy. Prvý sčítanec vo vzťahu sa označuje tiež ako *zdroj ohodnotenia*.

Uvedený vzťah vyjadruje fakt, že ohodnotenie stránky je úmerné počtu a dôležitosti odkazov, ktoré na ňu ukazujú. Výsledné hodnoty *PageRank* pre webové stránky sú výsledkom iteratívneho procesu. Ohodnotenie dôležitostí stránok sa využíva pri generovaní odpovede na vytvorený dopyt. Odpoveďou sú najrelevantnejšie stránky (dokumenty), pričom miera relevancie s dopytom je určená práve hodnotou *PageRank* danej stránky.

4.2.3 Rozšírenie algoritmu PageRank

Zaujímavé rozšírenie algoritmu PageRank prezentujú autori v (White, Smith, 2003), kde sa zaoberajú odhadovaním *relatívnej* dôležitosti vrcholov v grafoch. Relatívna dôležitosť je, na rozdiel od globálnej dôležitosti, vypočítavaná vzhľadom na určitú skupinu koreňových vrcholov a nie absolútne vzhľadom na celý graf (všetky jeho vrcholy).

10 Dôležitosť uzlov grafu je inde označovaná tiež ako autorita, prestíž alebo váha.

11 Konkrétna podoba vzťahu nie je zverejnená, nakoľko ide o registrovanú ochrannú známku spoločnosti Google, Inc.

Pre modifikáciu algoritmu PageRank nazvanú *PageRank with Priors* platí:

$$PageRank'(p_i) = (1-\beta)\left(\sum_{p_j \in M(p_i)} PageRank'(p_j)\, PB(p_i|p_j)\right) + \beta\, PB(p_i) \qquad (7)$$

kde $PB(p_j)$ je koeficient relatívnej dôležitosti (angl. prior bias) stránky p_j a β je tzv. spätná pravdepodobnosť, ktorá vyjadruje pravdepodobnosť, že sa surfer pri náhodnom surfovaní vráti na „koreňové“ stránky. Ostatné veličiny sú zhodné s použitými vo vzťahu (6). Pre $PB(p_j)$ ďalej platí:

$$\sum_{p_k \in R} PB(p_k) = 1 \qquad (8)$$

kde R je množina všetkých koreňových stránok. Pre každú stránku $p_i \notin R$ platí, že $PB(p_i) = 0$.

Princíp výpočtu relatívnej dôležitosti zohľadňuje dva faktory:

1. náhodný surfer je pri pohybe „navádzaný“ koeficientom β,
2. *zdroj ohodnotenia* je funkciou relatívnej dôležitosti stránky.

Na konci iteratívneho procesu je tak determinovaná relatívna dôležitosť každej stránky voči množine koreňových stránok.

Možnosť použitia pri objavovaní znalostí načrtá nasledujúci scenár: Uvažujme, že vrcholmi grafu sú pojmy (kľúčové slová, koncepty) doménovej oblasti. Nech je daný pojem p_i. Potom pojmy relevantné k pojmu p_i nájdeme použitím vzťahu (7), pričom koreňovým vrcholom bude jediný pojem p_i, pre ktorý platí $PB(p_i) = 1$.

5 Ciele práce

Vďaka adaptívnym systémom pre vzdelávanie je štúdium pohodlnejšie a efektívnejšie. Kvalita výučby priamo súvisí so schopnosťou systému prispôsobiť sa študentovým potrebám a cieľom. Pre efektívne vzdelávanie sú potrebné dômyselné mechanizmy prispôsobovania definované modelom prispôsobovania. Zložitosť modelu sa prenáša do celého systému, vrátane jeho jadra – doménového modelu, čo výrazne ovplyvňuje proces tvorby elektronických kurzov. Ten sa už netýka iba vytvorenia učebného materiálu, ale zahŕňa tiež definovanie metadát a prepojení na konceptuálnej úrovni. Pre neodborníka v informatike je to netriviálna úloha. Naplnenie adaptívneho kurzu adekvátnym obsahom pre dosiahnutie želanej funkcionality je zložitý a časovo náročný proces, čo je jedna z najväčších prekážok k lepšiemu etablovaniu AVS.

Trendy vo vývoji ASV naznačujú, že tvorbu kurzu je potrebné automatizovať. Učiteľ by sa tak mohol sústreďovať len na jednotlivé texty a koncepciu kurzu a spracovanie rutinných úloh by bolo prenesené na samotný systém. Odborníci za týmto procesom vidia nutnosť objavenia znalostí, ktoré stoja za textovým obsahom kurzov. Existujúce prístupy k podpore tvorby kurzov však veľmi málo využívajú potenciál, ktorý poskytujú metódy a techniky získavania znalostí a ktoré boli úspešne použité v iných oblastiach, akou je elektronické vzdelávanie.

V procese tvorby kurzov nemôžeme zanedbať ani aspekt vhodného používateľského rozhrania. Iba spojením s ním má prispôsobenie učiteľovi reálny význam. Bez pohodlného prostredia bude vytváranie kurzov stále komplikované.

Cieľom tejto práce je:

- návrh metódy automatizovaného vytvárania časti doménového modelu, ktorá:
  - ◊ bude založená na metódach a technikách získavania znalostí,
  - ◊ umožní odhaľovanie prepojení konceptov reprezentujúcich prezentovaný obsah,
- overenie návrhu v doméne výučby programovania pomocou príkladov,
- vytvorenie autorského nástroja, ktorý:
  - ◊ bude pokrývať časť tvorby kurzov determinovanú navrhnutou metódou,
  - ◊ umožní tvorbu časti doménového modelu používaného v portáli pre podporu výučby programovania[12].

12 Podrobný opis doménového modelu používaného v portáli môžeme nájsť v (Šimún, 2008).

6 Metóda automatizovaného získavania metadát o kurze

V tejto kapitole predstavíme metódu, ktorá slúži na automatizované vytváranie časti doménového modelu pre ASV. Uvažujeme dvojúrovňový doménový model projektu PeWePro, ktorý bol opísaný v časti 2.1. Vychádzame z predpokladu, že v okamihu začiatku tvorby kurzu sú dostupné určité výučbové materiály[13], ktoré chceme previesť na adaptívny kurz. Výstupom metódy je štruktúrovaná časť doménového modelu zahŕňajúca:

- relácie medzi objektmi výučby a konceptmi,
- relácie medzi konceptmi navzájom.

V návrhu metódy najväčší dôraz kladieme na odhaľovanie prepojení konceptov. Navrhujeme niekoľko variantov vychádzajúcich z rôznych techník získavania znalostí. Úspešnosť jednotlivých variantov vyhodnotíme a porovnáme v kapitole 8.

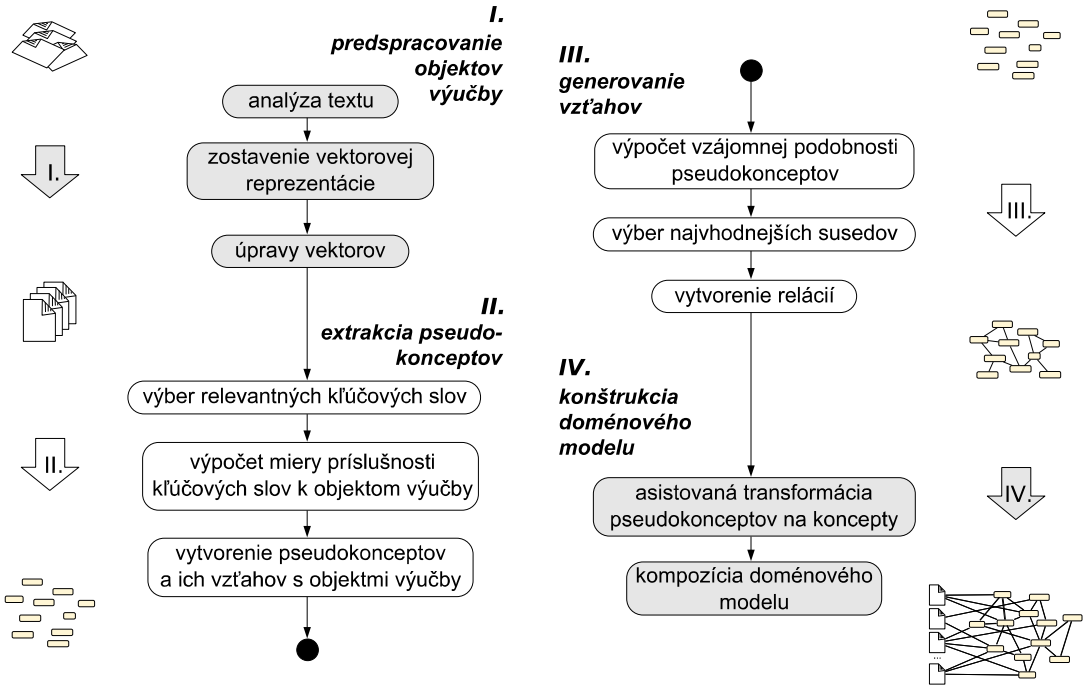
Metóda pozostáva zo štyroch krokov (pozri obrázok 6-1):

I. predspracovanie objektov výučby,
II. extrakcia pseudokonceptov,
III. generovanie vzťahov,
IV. konštrukcia doménového modelu.

V prvom kroku pripravíme objekty výučby do reprezentácie potrebnej pre ďalšie spracovanie: analyzujeme výučbové texty, spracujeme ich obsah do vektorovej reprezentácie, ktorú napokon upravíme v závislosti od formátovania vstupných dát. Využijeme tiež register pojmov, na základe ktorého zvýšime relevanciu jednotlivých slov.

V druhom kroku zo zostrojených vektorov na základe definovaného prahu relevancie vyberieme kľúčové slová. Pre každé kľúčové slovo vyjadríme jeho váhu pomocou metriky *tf-idf*. Kľúčové slová, ktoré prekračujú ďalší definovaný prah (čím eliminujeme príliš často sa vyskytujúce pojmy) označíme ako *pseudokoncepty* – kandidáty súce stať sa konceptmi. Pre každý pseudokoncept sú vytvorené relácie s výučbovými objektmi, ku ktorým prislúchajú.

13 Výučbové materiály sú reprezentované pomocou jazyka DocBook. Detailný opis reprezentácie vstupných dát je uvedený v technickej dokumentácii, ktorá je súčasťou príloh tejto práce.



Obrázok 6-1: Metóda automatizovaného získavania metadát o kurze.

Krok generovania vzťahov (medzi pseudokonceptmi) je kľúčovým krokom celej metódy. V ňom na aktuálny stav doménového modelu aplikujeme vybrané prístupy (techniky získavania znalostí, grafové algoritmy, príp. ich kombinácie), aby sme určili mieru vzájomnej podobnosti pseudokonceptov. Pre každý pseudokoncept potom vyberieme najvhodnejších susedov – pseudokoncepty, ktoré sú mu najviac podobné. Pre každého najvhodnejšieho suseda vytvoríme reláciu podobnosti v doménovom modeli.

Posledným krokom je krok finálnej konštrukcie doménového modelu (jeho časti). V ňom používateľ – učiteľ overuje správnosť vygenerovaného modelu. Stanovuje, ktoré pseudokoncepty sa stanú konceptami a ktoré relácie boli vygenerované správne. Takýmto spôsobom vytvorí špecifikovanú časť doménového modelu, ktorá po spojení s ostatnými časťami (tie sú dostupné pred začiatkom tvorby) môže byť použitá v portáli pre podporu vyučovania.

V nasledujúcich častiach opíšeme metódu detailnejšie.

6.1 Predspracovanie objektov výučby

Na začiatku je objekty výučby potrebné transformovať do podoby, ktorá bude obsahovať relevantné informácie pre ďalšie spracovanie – vytvoríme adekvátnu vnútornú reprezentáciu vstupných dát. Keďže uvažujeme vstupné dáta prevažne textového charakteru, navrhli sme riešenie vychádzajúce z tradičného vektorového modelu (VSM) – tzv. rozšírený vektorový model. Jednotlivé prvky vektora (váhy) sa nevypočítajú z počtu výskytov, ale z tzv. *relevancie*. Relevancia je miera dôležitosti slova (lemy) v dokumente. Ide o rozšírené kvantitatívne ohodnotenie

slova, ktoré zahŕňa okrem kardinality výskytu aj iné charakteristiky slova (pozri sekciu Úprava vektorov ďalej v tejto časti).

Predspracovanie objektov výučby pozostáva z troch krokov:

1. analýza textu,
2. zostavenie vektorovej reprezentácie,
3. úprava vektorov.

Analýza textu

V tomto kroku najprv prebehne *lexikálna analýza* výučbových textov, sú identifikované lexikálne jednotky – tokeny. Každému spracovávanému tokenu priradíme informáciu o jeho vhniezdení v hierarchii značiek jazyka DocBook, ktorú ešte využijeme neskôr pri dodatočných úpravách váh vektora.

Následne odstránime slová, ktoré majú takmer nulovú sémantickú hodnotu – tzv. *stop slová*. V prípade prirodzeného jazyka ide napríklad o spojky, častice, atď., v prípade programovacieho jazyka o operátory, čísla, atď. Každý typ jazyka má vlastný zoznam stop slov.

V ďalšej časti uskutočníme *lematizáciu* slov prirodzeného jazyka. Je to proces priraďovania tzv. lem slovným tvarom (tokenom) v texte (viac informácií sme uviedli v časti 4.1.5). Počas prototypovania vo fáze projektovania diplomovej práce sme experimentovali s viacerými lematizátormi spracúvajúcimi slovenčinu. Na základe výsledkov experimentov sme sa priklonili k použitiu lematizátora vyvinutého Jazykovedným ústavom Ľudovíta Štúra SAV (Garabík, 2006), ktorý dosiahol najlepšiu úspešnosť generovania lem na úrovni 80,3%.

Zostavenie vektorovej reprezentácie

Z lem a kľúčových slov programovacieho jazyka získaných v predchádzajúcom kroku zostavíme vektory, ktoré budú zatiaľ obsahovať len kardinality výskytov jednotlivých slov – v tomto momente máme vytvorený štandardný vektorový model.

Úprava vektorov

Posledným krokom pri predspracovaní objektov výučby je úprava aktuálnych vektorov do rozšíreného vektorového modelu. V ňom zavádzame pojem *relevancie* slova v dokumente – ide o rozšírené kvantitatívne ohodnotenie slova zahŕňajúce dodatočné znalosti o výučbovom texte, ktoré máme v čase spracovania k dispozícii. Účelom tohto kroku je skvalitnenie reprezentácie objektov výučby a vytvorenie lepšieho východiska pre ďalšie kroky metódy automatizovaného získavania metadát o kurze.

Výpočet relevancie prebieha v dvoch úrovniach:

1. spracovaním registra pojmov,
2. spracovaním značkovania.

Register pojmov obsahuje zoznam kľúčových slov doménovej oblasti a môže výrazne napomôcť ich identifikácii v zdrojových textoch. V tomto kroku spracovávaný dokument porovnáme s registrom pojmov a následne zvýšime relevanciu slov, ktoré sú prítomné aj v registri. Koeficient relevancie kľúčového slova v registri bol stanovený experimentálne na hodnotu 3.

Proces *spracovania značkovania* zahŕňa výpočet relevancie slov vo vektorovej reprezentácii dokumentu v závislosti od značkovania, ktoré sa na ne vzťahuje. Na jednotlivé slová dokumentu aplikujeme pravidlá, ktoré upravia hodnoty relevancie podľa hierarchie značiek získanej pri analýze textu. Pravidlá sú úzko spojené s jazykom DocBook, ktorý je využitý pri samotnej tvorbe objektov výučby. Príkladom takého pravidla je pravidlo:

```
if (xmlHierarchy contains „explanation“ and
    xmlHierarchy contains „code“)
then
    relevance(t_i) *= 2.5;
```

Pravidlo vyjadruje fakt, že ak ide o výučbový objekt typu *explanation* (je vhniezdený v značke `explanation`) a zároveň je na slovo t_i je použité značkovanie `code`, jeho relevancia bude vynásobená koeficientom 2.5. Takýmto spôsobom zvýšime relevanciu kľúčových slov programovacieho jazyka ktoré sa nachádzajú v pasážach textu prirodzeného jazyka – pri takýchto slovách predpokladáme, že sú pre text dôležité, keďže boli vybraté mimo kód programu[14]. Ďalšie pravidlá použité pri úprave vektorovej reprezentácie dokumentov pre výučbu programovania sú uvedené v technickej dokumentácii, ktorá je prílohou tejto práce.

6.2 Extrakcia pseudokonceptov

Po vykonaní všetkých krokov predspracovania textu je pripravená taká vnútorná reprezentácia objektov výučby, ktorá je potrebná pre aplikovanie krokov vedúcich k extrakcii pseudokonceptov. Ako sme uviedli skôr, pseudokoncepty nie sú pravé koncepty, ale iba akási abstrakcia, medzikrok medzi kľúčovým slovom a konceptom. Nie každý pseudokoncept bude konceptom doménového modelu.

Extrakcia pseudokonceptov je rozdelená do krokov:

1. výber relevantných kľúčových slov,
2. výpočet miery príslušnosti kľúčových slov k objektom výučby,
3. vytvorenie pseudokonceptov a ich vzťahov s objektmi výučby.

14 Na označenie ucelenej časti kódu programu v jazyku DocBook slúži element `program-listing`. Pri tomto elemente ponechávame relevanciu slova vo výučbovom objekte typu *explanation* na pôvodnej hodnote – predpokladáme, že takého kľúčové slová sú v danom momente študentovi známe z doterajšieho štúdia (inak by boli pomenované v texte (nie kóde)).

Výber relevantných kľúčových slov

V prvom kroku sú z množiny všetkých slov všetkých objektov výučby vybrané tie, ktorých relevancia presahuje určitý prah. Tento prah bol experimentálne nastavený na hodnotu 0.03. Vybrané slová sú označené ako kľúčové slová jednotlivých objektov výučby.

Výpočet miery príslušnosti kľúčových slov k objektom výučby

Výpočet miery príslušnosti kľúčových slov k objektom výučby je východiskovým bodom pre ďalší krok. Využívame tu rozšírenie známej metriky *tf-idf*.

Metrika *tf-idf* vyhodnocuje konečnú dôležitosť slova v danom dokumente vzhľadom na kardinalitu jeho výskytu v korpuse všetkých dokumentov. V našom prípade je jej účelom znížiť mieru príslušnosti kľúčových slov, ktoré sa v objekte výučby vyskytujú príliš často a z pohľadu prínosu znalostí o ňom nie sú relevantné. Vzťahy (9) a (10) sú rozšírením vzťahov (3) a (4), str. 25:

$$tf'(i,j) = \frac{rel(i,j)}{\sum_k rel(k,j)} \tag{9}$$

$$idf'(i) = \log \frac{|LO|}{|\{lo_j : k_i \in lo_j\}|} \tag{10}$$

kde $tf'(i,j)$ je frekvencia kľúčového slova k_i v objekte výučby lo_j a $idf'(i)$ je inverzná frekvencia kľúčového slova k_i v celom kurze. Napokon:

$$w'(i,j) = tf'(i,j) \cdot idf'(i) \tag{11}$$

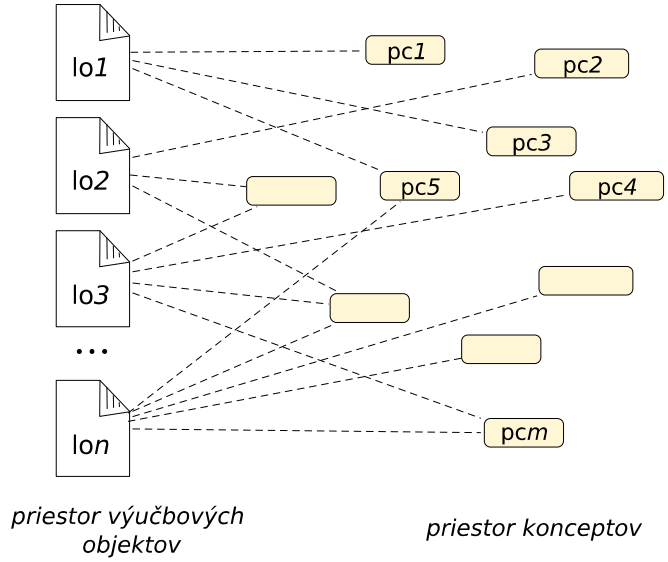
je výsledná miera príslušnosti kľúčového slova k_i k objektu výučby lo_j.

Vytvorenie pseudokonceptov a ich vzťahov s objektami výučby

Na to, aby sme kľúčové slovo objektu výučby povýšili na pseudokoncept, musí presahovať prah minimálnej miery príslušnosti stanovený na hodnotu 0,15. Je to najnižšia možná miera príslušnosti kľúčového slova k objektu výučby, pri ktorej je prepojenie kľúčového slova považované za nezanedbateľné. Eliminujeme tým prípady, kedy by sa k danému objektu výučby priradili pseudokoncepty, ktoré s nim súvisia len okrajovo.

Medzi vygenerovanými pseudokonceptmi a objektami výučby vytvoríme vzťahy – relácie ohodnotené mierou príslušnosti kľúčového slova (reprezentujúceho pseudokoncept) k objektu výučby.

V tomto momente máme vytvorený čiastočný doménový model, v ktorom zatiaľ absentujú vzťahy medzi pseudokonceptmi (pozri obrázok 6-2). Ich vytvoreniu sa venujeme v nasledujúcej časti.



*Obrázok 6-2: Stav doménového modelu po kroku extrakcie konceptov (lo*X *– výučbové objekty, pc*Y *– pseudokoncepty).*

6.3 Generovanie vzťahov

Pri generovaní vzťahov sa snažíme štrukturalizovať priestor konceptov (zatiaľ obsahujúci iba pseudokoncepty) tak, aby doménový model v čase výučby kurzu vytváral čo najväčší potenciál pre efektívnu aplikáciu mechanizmov prispôsobovania.

Generovanie vzťahov pozostáva z nasledujúcich krokov:

1. výpočet vzájomnej podobnosti pseudokonceptov,
2. výber najvhodnejších susedov,
3. vytvorenie vzťahov.

Výpočet vzájomnej podobnosti pseudokonceptov

Výpočet vzájomnej podobnosti pseudokonceptov je jadrom celej metódy automatizovaného získavania metadát o kurze a jeho opisu sa bližšie venujeme v samostatnej časti 6.4 tejto kapitoly.

Účelom tohto kroku je vytvorenie akejsi odľahčenej ontológie aplikačnej domény, ktorá zachytáva hierarchické, relačné, predispozičné a iné vzťahy medzi pseudokonceptmi.

Výber najvhodnejších susedov

Výber najvhodnejších susedov je jedna z najcitlivejších častí celej metódy. Určujeme tu, ktoré pseudokoncepty budeme považovať sa spolusúvisiace a ktoré už nie; najvýraznejšie vplývame na podobu výsledného doménového modelu a tým pádom aj kvalitu celého kurzu.

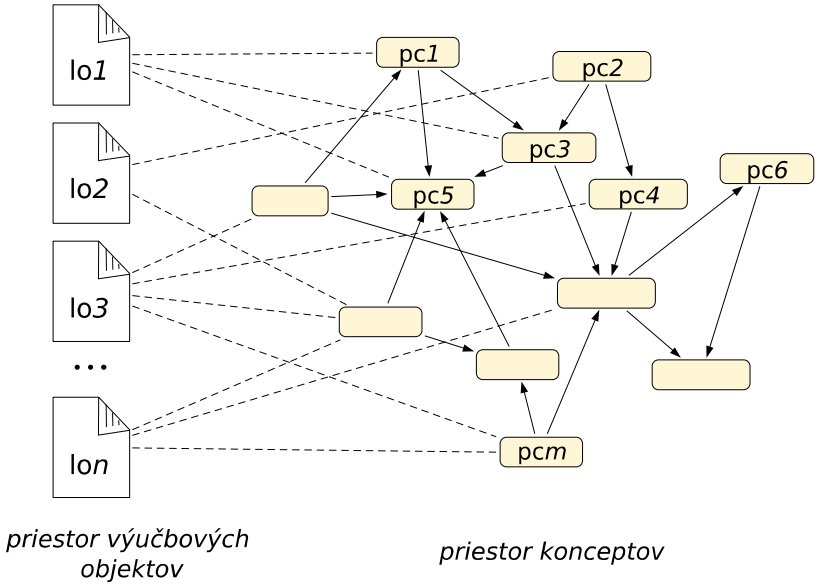
Najvhodnejších susedov vyberáme použitím vzťahu:

$$topN_i = \{pc_j : \sum_j s_{i,j} < r \sum_k s_{i,k}\} \quad (12)$$

kde $topN_i$ je množina najvhodnejších susedov pseudokonceptu pc_i, $s_{i,j}$ je miera podobnosti medzi pseudokonceptmi pc_i a pc_j získaná v predchádzajúcom kroku a zoradená od najvyššej po najnižšiu hodnotu; $r \in <0;1>$ je pomer sumy podobností najvhodnejších susedov ku všetkým susedom pseudokonceptu pc_i. Ak vezmeme napríklad $r = 0.2$, tak suma podobností medzi najvhodnejšími susedmi tvorí 20% z celkovej sumy. Tento postup sa úspešne využíva v kategorizačných systémoch (Diedrich, Balke, 2006).

Vytvorenie vzťahov

V tomto kroku pre každý koncept vytvoríme relácie s najvhodnejšími susedmi. Relácie zatiaľ nemajú priradený typ. Schéma stavu doménového modelu po tomto kroku je zobrazená na obrázku 6-3.



Obrázok 6-3: Doménový model po kroku generovania vzťahov (loX – objekty výučby, pcY – pseudokoncepty).

6.4 Konštrukcia doménového modelu

Tento krok je ukončením celej metódy automatizovaného získavania metadát o kurze. Ide v podstate o kontrolu automaticky generovaných elementov kurzu, kde učiteľ – autor kurzu poupraví podobu doménového modelu tak, aby podliehala jeho špecifikácii (Šimún, 2008).

Konštrukcia doménového modelu pozostáva z:

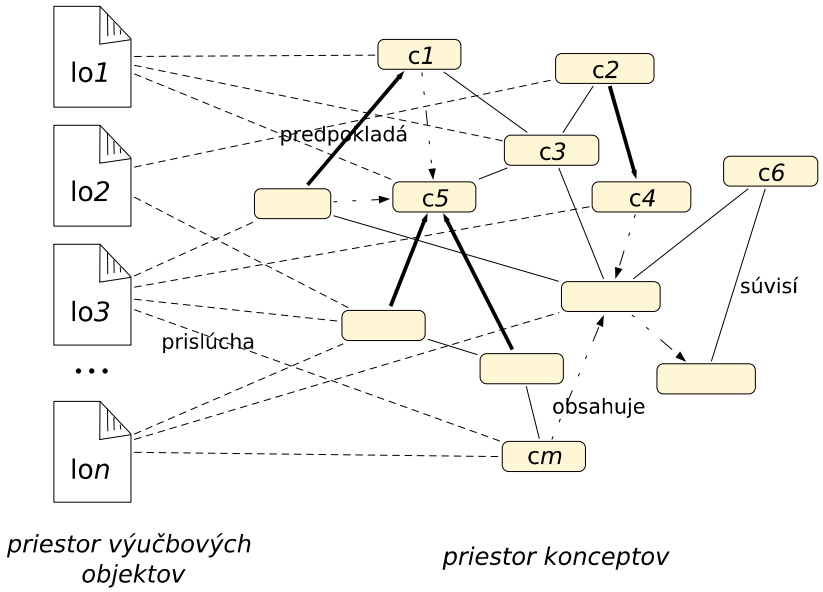
1. asistovanej transformácie pseudokonceptov na koncepty,
2. kompozície doménového modelu.

Asistovaná transformácia pseudokonceptov na koncepty

V tomto kroku autor kurzu spomedzi všetkých automaticky vygenerovaných pseudokonceptov vyselektuje také, ktoré sú relevantné pre kurz – pseudokoncepty sa stanú konceptmi. Kritérium relevancie je na samotnom učiteľovi a jeho odborných a pedagogických znalostiach.

Kompozícia doménového modelu

Na záver autor kurzu identifikuje typy relácií medzi konceptmi, nakoľko navrhnutá metóda tento problém nerieši. Môže tiež zmeniť váhy jednotlivých relácií. Výsledná podoba doménového modelu je zobrazená na obrázku .



Obrázok 6-4: Výsledná podoba doménového modelu po vykonaní všetkých krokov navrhnutej metódy (loX – objekty výučby, pcY – koncepty).

6.5 Výpočet vzájomnej podobnosti pseudokonceptov

Cieľom tohto kroku je kvantitatívne ohodnotenie vzájomnej podobnosti všetkých pseudokonceptov prítomných v kurze. Pri výpočte podobnosti sa snažíme využiť maximum z doteraz získaných znalostí o kurze. V tomto kroku sa doménový model nachádza v stave zobrazenom na obrázku 6-2 (str. 38). Poznáme vektorovú reprezentáciu objektov výučby a relácie medzi pseudokonceptami a objektami výučby vytvorili graf. Na základe týchto poznatkov sme vo fáze projektovania diplomovej práce navrhli viaceré varianty výpočtu vzájomnej podobnosti pseudokonceptov, z ktorých každý využíva špecifický prístup objavovania znalostí. Navrhnutými variantmi sú:

1. vektorový prístup,
2. šírenie aktivácie,

3. analýza založená na algoritme PageRank.

V priebehu experimentovania s navrhnutými variantmi sme dospeli k myšlienke ich kombinácie, ktorá vyústila do vzniku ďalších dvoch variantov:

4. kombinácia variantov 1. a 3.
5. kombinácia variantov 2. a 3.

Každý z uvedených variantov podrobne opíšeme v nasledujúcich častiach.

6.5.1 Vektorový prístup

V prípade tohto variantu využijeme vektorovú reprezentáciu objektov výučby na zostavenie analogickej reprezentácie konceptov. Každý koncept je reprezentovaný váženou sumou vektorov tých objektov výučby, ku ktorým prislúcha:

$$\vec{c}_i = \sum_j w'_{i,j} \overrightarrow{lo}_j \tag{13}$$

kde $w'_{i,j}$ je normalizovaná váha relácie medzi konceptom c_i a objektom výučby lo_j. Normalizovaná váha je proporcionálnym vyjadrením pôvodnej hodnoty (pozri vzťah (11), str. 37) ako percento sumy váh všetkých relácií výučbového objektu.

Po tomto kroku vypočítame mieru podobnosti medzi konceptami pomocou známej metriky kosínusovej vzdialenosti (pozri vzťah (5), str. 27). Nevýhodou tohto variantu je fakt, že získaná matica vzájomnej podobnosti konceptov je symetrická, tj. potenciálne relácie medzi konceptami nebudú orientované[15].

6.5.2 Šírenie aktivácie

Pri tomto variante sa pozeráme na graf doménového modelu ako na kontextovú sieť (pozri obrázok 6-5). V tomto type sietí sa na hľadanie podobných uzlov často využíva metóda šírenia aktivácie (pozri časť 4.2.1, str. 28). Dopytovaný uzol (v našom prípade pseudokoncept) je nabitý energiou E, ktorá sa prešíri k susedným uzlom prostredníctvom hrán (v našom prípade relácií). Tento princíp využijeme pri odhalení vzájomnej podobnosti konceptov; algoritmus spočíva vo vykonaní nasledujúcich krokov:

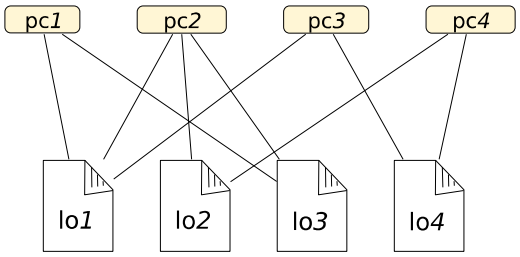
Pre každý koncept c_i:

1. nabi koncept c_i aktivačnou energiou $E_0 = 100$;
2. prešír energiu celým grafom;
3. vypočítaj mieru podobnosti s ostatnými konceptami.

Pri šírení energie v kroku 2 využívame modifikáciu algoritmu uvedeného na str. 29: energiu šírenú medzi medzi susedov nedistribuujeme rovnomerne, ale

15 V skutočnosti však orientované relácie vytvoríme. Pre každú reláciu r_{ij} medzi konceptmi c_i a c_j však bude platiť: $r_{i,j} = r_{j,i}$, čo uberá na znalosti, ktorú o podobnosti konceptov týmto spôsobom dostaneme. Ako ukážeme neskôr, tento nedostatok sa dá odstrániť dodatočným spracovaním získaného grafu konceptov (pozri časť 6.4.4).

proporcionálne podľa pomeru váh jednotlivých relácií na základe miery príslušnosti pseudokonceptu k objektu výučby.



*Obrázok 6-5: Doménový model ako kontextová sieť (pc*X *– pseudokoncepty, lo*Y *– objekty výučby)*

Kľúčovým krokom tohto algoritmu je krok 3. Po prešírení aktivačnej energie celým grafom je každý pseudokoncept nabitý určitou energiou E_i. Podobnosť medzi pseudokonceptom pc_i a ostatnými pseudokonceptami definujeme pomocou vzťahu:

$$saSim_{i,j} = \frac{E_j}{\sum_k E_k} \log(d_{i,j}) \tag{14}$$

kde $saSim_{i,j}$ je podobnosť medzi konceptami pc_i a pc_j, E_j je energia pseudokonceptu pc_j a $d_{i,j}$ je najkratšia cesta medzi uzlami pc_i a pc_j.

Takto získané podobnosti sú predmetom spracovania ďalších krokov generovania vzťahov.

6.5.3 Analýza založená na algoritme PageRank

Grafový charakter doménového modelu využívame aj pri tretej alternatíve výpočtu vzájomnej podobnosti pseudokonceptov. Východiskom pre tento variant boli grafové algoritmy pre odhadovanie relatívnej podobnosti medzi uzlami (White, Smith, 2003).

Tieto algoritmy vypočítavajú relatívne ohodnotenie uzla vzhľadom na definovanú množinu koreňových uzlov. Jedným z týchto algoritmov je aj variácia algoritmu PageRank, ktorý je úspešne používaný na vyhľadávanie na webe (pozri časť 4.2.3).

V našom návrhu tento algoritmus využijeme na odhalenie skrytých súvislostí medzi pseudokonceptami. Aplikujeme pritom nasledujúce kroky:

1. pre každý pseudokoncept pc_i nájdi koncepty, ktoré sú s ním prepojené prostredníctvom práve jedného objektu výučby,
2. pre každú dvojicu pseudokonceptov vypočítaj váhu ich vzájomného výskytu *coocSim*,
3. pre každý pseudokoncept pc_i vyber najvhodnejších susedov pomocou váhy *coocSim*,
4. zostroj graf G_{tmp}, kde uzlami sú pseudokoncepty a hrany sú ohodnotené váhou *coocSim*,
5. pre každý pseudokoncept pc_i v grafe G_{tmp} vypočítaj váhy uzlov aplikovaním algoritmu PageRank s prioritami, pričom za koreňový uzol považuj pseudokoncept pc_i.

V kroku 2 vypočítame váhu vzájomného výskytu *coocSim* pomocou vzťahu:

$$coocSim_{i,j} = \sum_{lo_k \in LO} (w'_{i,k} w'_{j,k}) \quad (15)$$

kde $w'_{i,j}$ je miera príslušnosti pseudokonceptu pc_i k objektu výučby lo_j (pozri vzťah (11), str. 37). Ide o výpočet na základe jednoduchého princípu: ak napríklad pseudokoncept pc_1 súvisí s objektom výučby lo_5 mierou príslušnosti 0.6 a pseudokoncept pc_3 súvisí s tým istým objektom výučby mierou príslušnosti 0.9, váha vzájomnej výskytu pseudokonceptov pc_1 a pc_3 je daná súčinom ich mier príslušností k objektu výučby lo_5: $0.6 \times 0.9 = 0.54$. Ak sú dané koncepty vzájomne zviazané prostredníctvom viacerých objektov výučby, výsledná váha $coocSim_{1,3}$ je daná sumou takýchto čiastkových súčinov.

V kroku 3 využijeme na výber najvhodnejších susedov princíp opísaný v časti 6.3 (pozri vzťah 12, str. 39).

Po aplikovaní uvedenej modifikácie algoritmu PageRank v kroku 5 získame pre každý pseudokoncept graf, v ktorom už budú zahrnuté skryté väzby medzi jednotlivými uzlami. Tieto grafy sú výstupom celého variantu, pretože implicitne obsahujú informáciu o vzájomnej podobnosti konceptov – miera podobnosti je určená tzv. prestížou uzla. Keďže bol aplikovaný algoritmus PageRank s prioritami a koreňovým uzlom bol zvolený pseudokoncept pc_i, prestíž uzla je vyjadrením relatívnej podobnosti ostatných uzlov práve vzhľadom na pseudokoncept pc_i.

Opísaný variant bol čiastočne inšpirovaný prácou Dr. Diedricha a Dr. Balke z univerzity v Hannoveri (Diedrich, Balke, 2006).

6.5.4 Kombinovanie variantov

Ak sa pozrieme lepšie na prvé 4 kroky tretieho variantu, zistíme, že v nich v podstate konštruujeme akýsi dočasný graf – graf s neúplnými informáciami o doménovom modeli, ktorý potom pomocou algoritmu PageRank rozšírime a optimalizujeme do finálnej podoby. Pri tejto príležitosti sa naskytá otázka, čo sa stane v prípade, ak dočasným grafom bude o čosi „kvalitnejší“ graf, napríklad graf pseudokonceptov získaný pomocou prvých dvoch variantov. Kombinovaný prístup by potom vyzeral následovne:

1. zostroj dočasný graf G_{tmp}, v ktorom uzlami budú pseudokoncepty a hranami relácie získané vektorovým prístupom (variant 1) alebo šírením aktivácie (variant 2),
2. pre každý pseudokoncept pc_i v grafe G_{tmp} vypočítaj váhy uzlov aplikovaním algoritmu PageRank s prioritami, pričom za koreňový uzol považuj pseudokoncept pc_i.

Krok 2 je analógiou kroku 5 v treťom navrhovanom variante výpočtu vzájomnej podobnosti konceptov. Slúži na odhalenie skrytých podobností konceptov.

Algoritmus PageRank pri oboch variantoch optimalizuje vlastnosti dočasného grafu, v prípade použitia vektorového prístupu dokonca umožňuje rozšíriť neorientované hrany na orientované a vytvára tak komplexnejší doménový model.

7 Autorský nástroj CourseDesigner

Súčasťou cieľov tejto práce je vytvorenie autorského nástroja, pomocou ktorého bude možné vytvoriť kurz v podobe podporovanej portálom pre výučbu programovania vyvíjanom na Ústave informatiky a softvérového inžinierstva pri FIIT STU.

Základ autorského nástroja staviame na príjemnom a jednoducho ovládateľnom používateľskom prostredí, ktoré odbremení učiteľa od manuálneho definovania štruktúry kurzu, zvýši efektivitu jeho tvorby a prispeje k rozšíreniu používania adaptívnych systémov pre vzdelávanie ako takých.

Nástroj CourseDesigner okrem rozhrania pre manuálnu tvorbu kurzu obsahuje modul, v ktorom je integrovaná metóda automatizovaného získavania metadát o kurze.

7.1 Špecifikácia požiadaviek

Nástroj CourseDesigner pokrýva procesy súvisiace s tvorbou nasledujúcej časti doménového modelu:

- tvorba konceptov,
- tvorba relácií:
  - ◊ medzi objektmi výučby a konceptmi (relácia *belongsTo*),
  - ◊ medzi konceptmi navzájom (relácie *isRelatedTo, contains, prerequsity*).

Uvedené procesy tvorby možno realizovať manuálne alebo automatizovane. Manuálnu tvorbu časti modelu demonštruje nasledujúci scenár použitia:

Krok	Opis činnosti
1	Vytvorenie novej časti doménového modelu.
2	Načítanie objektov výučby s definovanými hierarchickými vzťahmi (relácia *contents*).
3	Vytvorenie konceptov, ktoré sú v objektoch výučby obsiahnuté a definovanie ich základných metadát.
4	Prepojenie konceptov s objektmi výučby pomocou relácií *belongsTo* a definovanie váhy relácií.
5	Definovanie vzťahov medzi konceptmi výučby pomocou relácií *isRelatedTo*, *contains*, *prerequsity* a definuje váhu relácií.
6	Uloženie vytvorenej časti doménového modelu do špecifikovaného súboru.

Automatizovaná tvorba doménového modelu použitím nástroja CourseDesigner využíva integrovanú metódu na získavanie metadát o kurze.

Tvorba kurzu je opísaná prostredníctvom nasledujúceho scenára použitia (v zátvorkách sú uvedené nepovinné kroky):

Krok	Opis činnosti
1	Vytvorenie novej časti doménového modelu.
2	Načítanie objektov výučby s definovanými hierarchickými vzťahmi (relácia *contents*).
3	Automatické predspracovanie objektov výučby (pozri časť 6.1).
(3b)	Kontrola predspracovania objektov výučby a úprave ich vnútornej reprezentácie.
4	Automatická extrakcia konceptov (pozri časť 6.2)
(4b)	Kontrola extrahovaných konceptov, vytvorenie chýbajúcich / úprava existujúcich konceptov.
5	Automatické generovanie vzťahov pomocou zvoleného variantu (pozri časť 6.3).
(5b)	Kontrola generovaných relácií, vytvorenie chýbajúcich / úprava existujúcich.
6	Finálne úpravy na doménovom modeli zahŕňajúce tvorbu / úpravu konceptov a všetkých typov relácií (pozri časť 6.4)
7	Uloženie vytvorenej časti doménového modelu do špecifikovaného súboru.

7.2 Funkcionalita nástroja

Funkcionalitu nástroja môžeme rozdeliť do piatich skupín podľa súvisu s:

- kurzom[16],
- objektmi výučby,
- konceptmi,
- reláciami,
- vedľajšími procesmi.

Jednotlivé prípady stručne opíšeme vymenovaním prípadov použitia, ktoré s danou funkcionalitou súvisia. Kurzívou sú označené prípady, ktoré súvisia s navrhnutou metódou generovania metadát o kurze.

Kurz

- vytvorenie nového kurzu,
- otvorenie existujúceho kurzu,
- uloženie aktuálneho kurzu.

Objekty výučby

- Načítanie objektov výučby,

16 V tejto časti používame pojem kurz na označenie tej časti doménového modelu, ktorá je pomocou nástroja vytváraná.

- *Predspracovanie objektov výučby,*
- Úprava objektov výučby.

Koncepty

- *Extrakcia konceptov,*
- Pridanie konceptu,
- Úprava konceptu,
- Odstránenie konceptu.

Relácie

- *Generovanie relácií,*
- Pridanie relácie,
- Úprava relácie,
- Odstránenie relácie.

Vedľajšie procesy

- Načítanie kľúčových slov doménovej oblasti,
- Úprava kľúčových slov doménovej oblasti,
- Rekonštrukcia grafovej vizualizácie kurzu.

7.3 Používateľské rozhranie

Táto časť poskytuje základný náhľad na používateľské rozhranie nástroja CourseDesigner. Detaily používania nástroja sú opísané v používateľskej príručke (ktorá je súčasťou príloh tejto práce).

Jadro používateľského rozhrania pozostáva z hlavného okna a troch základných dialógových okien.

Hlavné okno

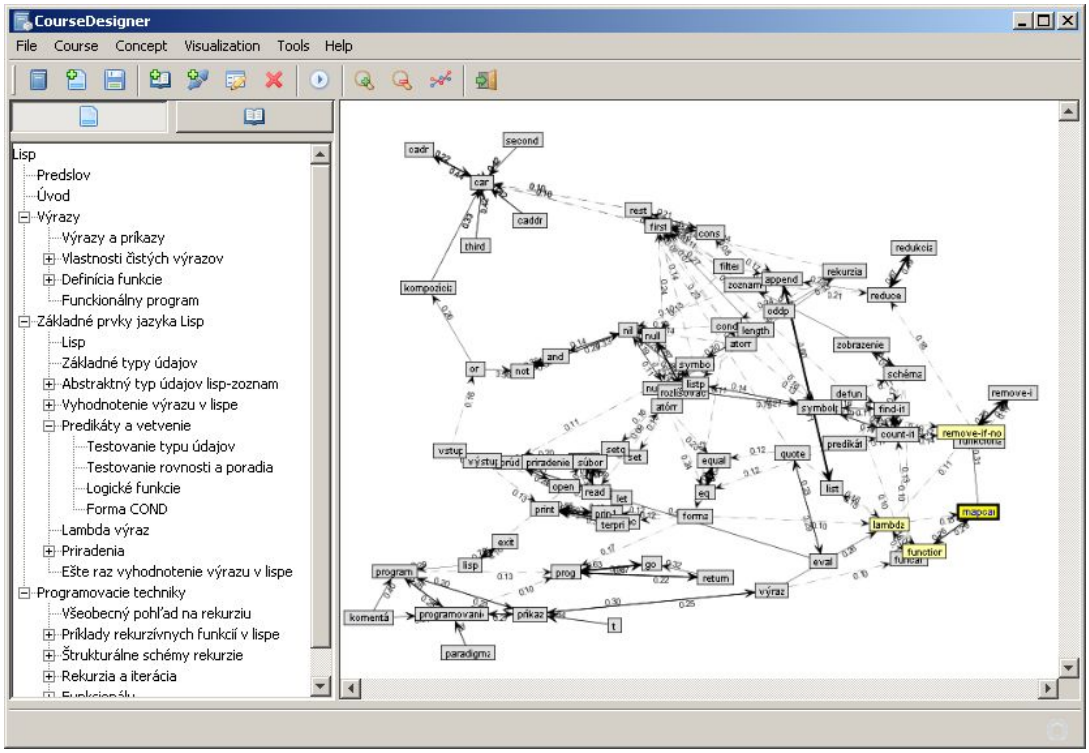
Hlavné okno používateľského rozhrania obsahuje štyri základné časti (pozri obr. 7-1):

- hlavné menu,
- lišta nástrojov,
- panel objektov výučby a konceptov,
- interaktívny graf kurzu.

Hlavné menu a *lišta nástrojov* zjednodušuje prístup k najčastejšie používaným funkciám celej aplikácie prostredníctvom prehľadných ikon.

Panel objektov výučby a konceptov umožňuje prepínanie medzi zobrazením hierarchickej stromovej štruktúry výučbových objektov a zoznamom konceptov

v kurze. Cez tento panel je po označení konkrétneho prvku možné odstraňovať a upravovať vlastnosti výučbových objektov, resp. konceptov.



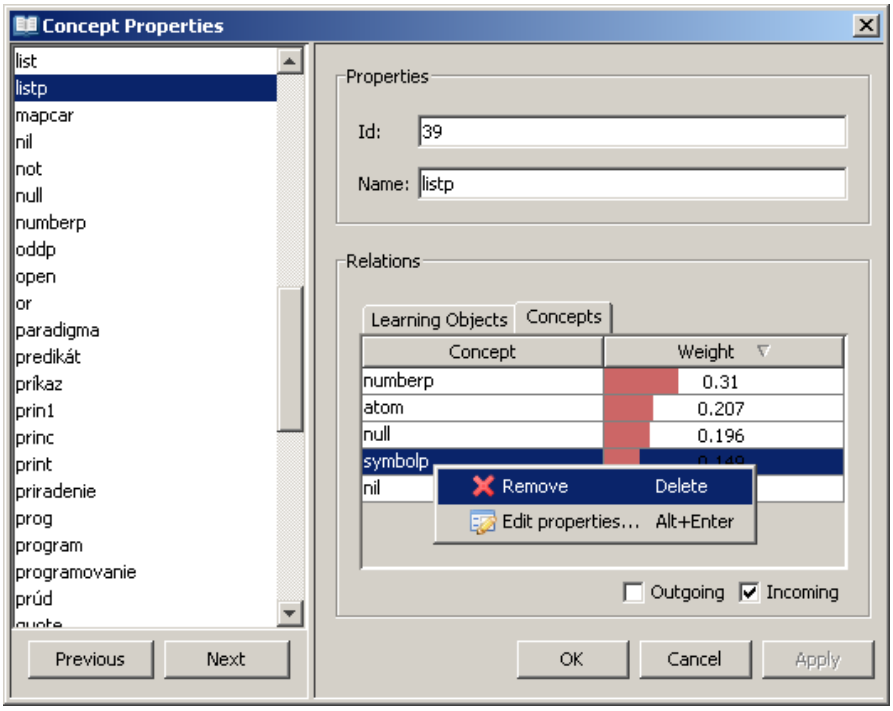
Obrázok 7-1: Hlavné okno nástroja CourseDesigner

Štruktúra konceptov je vizualizovaná prostredníctvom *interaktívneho grafu*. Graf umožňuje prácu s konceptmi a reláciami: ich pridávanie, odstraňovanie a úpravu vlastností. Vykresľovanie grafu je realizované dynamicky, graf sa stále prispôsobuje aktuálnemu stavu kurzu. Pohodlné používateľské rozhranie je zaručené možnosťou transfokácie grafu podľa veľkosti kurzu (zoom), premiestňovania jednotlivých uzlov, rekonštrukcie aktuálneho rozmiestnenia uzlov a ďalšími funkciami.

Základné dialógové okná

Dialógové okná jednotlivých elementov kurzu (výučbové objekty, koncepty, relácie) slúžia na interaktívnu prácu so štruktúrou kurzu. Na obrázku 7-2 je zobrazená ukážka dialógového okna pre koncepty. Okno pozostáva z troch základných častí:

- zoznam konceptov (vľavo),
- metadáta o koncepte (vpravo hore),
- relácie konceptu (vpravo dole).



Obrázok 7-2: Dialógové okno konpcetu

V tomto okne je možné upravovať vlastnosti konceptu vrátane úprav jeho relácií či už s objektmi výučby alebo ostatnými konceptami. Ak to používateľovi nestačí, tak pomocou vyskakovacieho menu dostupného v oblasti tabuliek je možné vyvolať ďalšie dialógové okná pre výučbové objekty, príp. relácie.

7.4 Technické detaily

Nástroj bol implementovaný na platforme Java vo verzii 6 využitím knižníc rámca Swing Application Framework pre tvorbu aplikácií s bohatým GUI. Voľba prostredia Java zabezpečuje platformovú nezávislosť nástroja. Nástroj využíva niekoľko externých závislostí. Vizualizačná časť je vytvorená použitím knižnice JUNG (Java Universal Network/Graph framework). V procese lematizácie je využitá knižnica sg-cdb (strange gizmo constant database) pre prístup do konštantnej databázy slovných tvarov slovenského jazyka.

Návrh systému bol koncipovaný tak, aby bol systém ľahko rozšíriteľné. Systém definuje niekoľko rozhraní, pomocou ktorých je možné rozšíriť, príp. doplniť komponenty, akými sú napríklad lematizátory (v súčasnosti sú podporované lematizátory slovenčiny, modulárna architektúra systému však umožňuje pridať ďalšie). Pri návrhu bolo využitých niekoľko známych návrhových vzorov, čím je nástroj jednoducho rozšíriteľný o nové stratégie generovania vzťahov medzi konceptmi.

Nástroj slúži okrem prostredia pre tvorbu kurzov aj ako prostredie pre jednoduché overenie navrhnutých variantov metódy. V procese tvorby kurzu je

používateľ vyzvaný na výber variantu, príp. modifikáciu vybraných parametrov. Overovanie je navrhnuté ako modulárne súčasť nástroja, keďže finálna podoba nepredpokladá využitie všetkých variantov generovania vzťahov medzi konceptmi. Modulárna architektúra umožňuje vykonať iba minimálne množstvo zmien, aby nástroj mohol používať aj konečný používateľ – tvorca kurzu.

7.4.1 Architektúra systému

Systém môžeme rozdeliť do niekoľkých funkčných blokov:

- vstupno výstupné operácie:
  - ◇ parsovanie, zapisovanie XML,
  - ◇ načítavanie textových súborov,
- spracovanie prirodzeného jazyka,
  - ◇ lematizácia,
- aplikačná logika metódy,
- grafové algoritmy a vizualizácia grafu,
- grafické rozhranie nástroja.

Z pohľadu konkrétnych komponentov nástroj pozostáva z balíkov:

- `core` – pokrýva jadro aplikačnej logiky, implementuje jednotlivé varianty generovania vzťahov medzi konceptmi,
- `parser` – zabezpečuje spracovanie súborov,
- `lemmatizer` – implementuje jednotlivé algoritmy lematizácie,
- `gui` – pokrýva grafické rozhranie nástroja.

8 Overenie riešenia v doméne výučby programovania

8.1 Sledované ciele

Pre overenie navrhnutej metódy sme stanovili nasledujúce ciele:

- vyhodnotenie úspešnosti generovania časti doménového modelu pre každý navrhnutý variant a ich vzájomné porovnanie;
- experimentálne nastavenie najvhodnejších hodnôt parametrov metódy pre jednotlivé varianty metódy;
- čiastočné otestovanie nástroja CourseDesigner.

Pri overovaní sme sa nezaoberali komplexným testovaním autorského nástroja CourseDesigner, nakoľko cieľom projektu nebolo vytvoriť finálny produkt. Čiastočným testovaním nástroj prešiel v procese plnenia predošlých dvoch cieľov – pri vyhodnocovaní úspešnosti a experimentmi s metódou, ktorá do neho bola implementovaná.

8.2 Testovacie dáta

Navrhnutú metódu automatizovaného získavania metadát o kurze sme overovali v doméne výučby programovania. V rámci projektu PeWePro boli vytvorené desiatky výučbových textov (objektov výučby), ktoré pokrývajú niekoľko kurzov prebiehajúcich na Fakulte informatiky a informačných technológií. Jedným z nich je aj kurz funkcionálneho a logického programovania (FLP) pod vedením prof. Bielikovej. Vybranými testovacími dátami je jeho časť – *funkcionálne programovanie* rozdelené do 70 výučbových objektov, ktoré sú vytvorené podľa učebnice Funkcionálne a logické programovanie (Bieliková, Návrat, 2000).

Referenčný kurz

Výstupy získané zo vstupných dát budú vyhodnocované voči manuálne zostavenej referenčnej štruktúre kurzu funkcionálneho programovania. Na tvorbe referenčnej štruktúry sa podieľala aj prof. Bieliková. Keďže ohodnotenie každej relácie v kurze váhou z intervalu <0; 1> je netriviálna úloha, priraďovali sme iba váhy:

- 1 – ak koncepty spolu veľmi súvisia (určite súvisia),
- 0.5 – ak koncepty spolu trochu súvisia (možno súvisia),
- 0 – ak koncepty spolu veľmi málo súvisia (určite nesúvisia).

8.3 Návrh experimentov

Zvoliť správne kritérium vyhodnotenia úspešnosti kurzu (resp. jeho časti) nie je také ľahké, ako by sa na prvý pohľad mohlo zdať. V rôznych príbuzných doménach sa využívajú rôzne spôsoby vyhodnocovania, niekedy príliš málo prísne. My pri overovaní metódy automatického získavania metadát použijeme metriky *úplnosť* (angl. recall) a *presnosť* (angl. precision) známe zo získavania znalostí. Budeme vyhodnocovať úspešnosť vygenerovaných relácií, konkrétne:

- pomer správne vytvorených ku všetkým vytvoreným (presnosť) a
- pomer správne vytvorených ku všetkým relevantným, tj. takým, ktoré sú v kurze žiaduce (úplnosť).

Použité metriky sú definované ako:

$$recall = \frac{retrieved \cap relevant}{relevant} \tag{16}$$

$$precision = \frac{retrieved \cap relevant}{retrieved} \tag{17}$$

kde *relevant* je množina všetkých relevantných relácií a *retrieved* je množina všetkých (nami) vytvorených relácií. Na výpočet harmonického priemeru oboch metrík je definovaná tzv. *F-metrika* (angl. F-measure*)*:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{18}$$

Z praxe je známe, že metriky úplnosť a presnosť sú zvyčajne nepriamo úmerné – ak klesá presnosť, tak rastie úplnosť, a naopak. F-metrika sa javí vhodná, pretože z pohľadu tvorcu kurzu nie je žiadúce, ak sa automatizáciou kurzu vytvorilo primálo relácií (hoci správnych), pretože ostatné bude musieť vytvoriť manuálne. Vtedy je dosiahnutá vysoká presnosť, ale nízka úplnosť). Tak isto nechceme, aby popri veľkom množstve správnych relácií vzniklo veľa nesprávnych (automatizácia kurzu je kontraproduktívna – je potrebné dodatočné úsilie na mazanie nesprávnych relácií). Táto možnosť je charakterizovaná vysokou úplnosťou pri nízkej presnosti. F-metrika je akousi zlatou strednou cestou medzi oboma prípadmi.

Pre lepšie vyhodnotenie výsledkov voči dostupnej referenčnej štruktúre kurzu sme ešte sledovali pozmenenú metriku úplnosti – *recall**. Túto metriku sme definovali ako:

$$recall* = \frac{retrieved \cap (relevantA \cup relevantB)}{relevantA \cup (relevantB \cap retrieved)} \quad (19)$$

Kde *relevantA* je množina relevantných relácií referenčného kurzu ohodnotených váhou 1 a *relevantB* je množina relevantných relácií ohodnotených váhou 0.5. Uvedený vzťah potom určuje pomer všetkých správne vytvorených relácií k reláciám, ktoré v kurze „musia“ byť (koncepty spolu *určite* súvisia), pričom sa mierne zanedbajú relácie, ktoré v kurze byť „môžu“ (koncepty spolu *možno* súvisia).

8.4 Realizácia experimentov a výsledky

Pre realizáciu experimentov bola použitá implementácia metódy v autorskom nástroji CourseDesigner. Priebeh experimentov bol automatický, štruktúra kurzu nebola v jednotlivých medzikrokoch metódy nijako manuálne menená.

Testovacie dáta pozostávali zo 70 objektov výučby. Po krokoch predspracovania objektov výučby a extrakcie pseudokonceptov bolo vytvorených 77 pseudokonceptov. Medzi pseudokonceptmi a výučbovými objektmi boli vytvorené relácie ováhované pomocou metriky *tf-idf*. Úspešnosť lematizácie slovenčiny v týchto procesoch bola 80,3 %. Po týchto krokoch sme postupne vyhodnocovali všetkých päť navrhnutých variantov, pričom sme vykonali niekoľko desiatok experimentov, aby sme nastavili najlepšie hodnoty pre špecifické parametre variantov.

Výsledky experimentov sú zobrazené v tabuľke 8-1.

Tabuľka 8-1: Výsledky experimentov (v %)

Variant	R	P	F	R*	F*
Vektorový prístup	0.594	0.509	0.549	0.793	0.620
Šírenie aktivácie	0.544	0.443	0.488	0.784	0.566
PageRank' analýza	0.500	0.569	0.532	0.741	0.652
Vektorový prístupu + PageRank' analýza	0.567	0.528	0.547	0.803	0.638
Šírenia aktivácie + PageRank' analýza	0.572	0.453	0.506	0.824	0.585

Legenda: *R – úplnosť (recall)* *R* – modifikácia úplnosti*
P – presnosť (precision) *F* – F-metrika po použití modifikovanej metriky úplnosti*
F – F-metrika (F-measure)

8.5 Interpretácia výsledkov a diskusia

Je veľmi ťažké vyjadriť sa k úspešnosti navrhnutej metódy ako celku, nakoľko výsledky nie je s čím porovnať. Určite by sme si želali vyššie čísla, kde je však hranica, ktorú môžeme dosiahnuť automatizovaným počítačovým spracovaním? Určite to nebude 100%, koľko teda?

Najvyššia úspešnosť podľa tradičnej F-metriky dosiahla hodnotu takmer 55%. To znamená, že viac ako polovica doménového modelu je vytvorená správne. Pri menej prísnom kritériu, keď sa zameriame len na naozaj dôležité relácie a „oželieme“ tie menej dôležité (to však ešte nemusí nutne znamenať, že tieto väzby nevzniknú na úrovni odporúčania výučbového materiálu pomocou modelu prispôsobovania!) máme správne vytvorené takmer dve tretiny doménového modelu. Je toto percento postačujúce na to, aby učitelia tvorili kurzy naozaj efektívnejšie? Na to bez komplexnejšieho a dlhodobejšieho procesu vyhodnotenia odpovedať nevieme.

Správne vytvorenie doménového modelu v našom prípade znamená podobajúce sa referenčnej štruktúre kurzu. Nakoľko však musí skutočná podoba doménového modelu odrážať štruktúru kurzu, ktorú sme definovali? Neobsahuje nami definovaná štruktúra len informácie o tom, čo chceme mať pri kontakte s danými konceptmi odporučené adaptívnym systémom? Do akej miery je naša predstava o správne zostrojenom doménovom modeli relevantná pre mechanizmy prispôsobovania?

Vyhodnotenie variantov

Na základe porovnaní výsledných hodnôt F*-metriky môžeme skonštatovať, že úspešnosť jednotlivých variantov je približne rovnaká – rozdiel medzi najlepším a najhorším výsledkom nečiní ani 10%. To možno napovedá o miere vplyvu východiskového bodu, z ktorého všetky varianty vychádzali. Zostrojenie vhodnej reprezentácie výučbových objektov možno až príliš „obmedzuje“ aplikáciu ďalších metód. To dokazuje aj fakt, že veľmi dobre (vzhľadom na ostatné varianty) dopadlo aj použitie vektorového prístupu, ktorý najviac zakladal práve na zostrojenej reprezentácii objektov výučby.

Najvyššiu úspešnosť pri menej prísnom kritériu hodnotenia úspešnosti sme dosiahli použitím analýzy pomocou algoritmu PageRank s prioritami. Stačilo poznať jednoduché väzby medzi objektmi výučby a konceptmi (získané spracovaním textu na základe metriky *tf-idf*) a z nich odvodiť východiskový graf pre algoritmus. Výsledky prekonali aj kombinácie variantov, ktoré vychádzali zo sofistikovanejšie skonštruovaných grafov. V oboch prípadoch sme očakávali výraznejšie zlepšenie úspešnosti ako bez použitia optimalizácie algoritmom PageRank. Rozdiel úspešnosti medzi pôvodným variantom a jeho kombináciou s algoritmom PageRank (uvažujúc metriky F a F*) je v najlepšom prípade iba 2 %. Aj keď algoritmus poupravil váhy niektorých relácií, zmeny boli primalé na to, aby

ovplyvnili výber najlepších susedov. Pri pokuse o výraznejšiu zmenu váh boli z aktuálnej štruktúry dočasného grafu generované nekorektné prepojenia. Opäť argument pre hypotézu o potrebe zostrojenia „precíznejšej“ reprezentácie.

Celkovo najhoršie spomedzi ostatných dopadol variant šírenia aktivácie. Ukázalo sa, že energia šírená medzi jednotlivými uzlami doménového modelu neodhalí skryté vzťahy medzi konceptami. Možno je príčina v nevhodnej optimalizácii naakumulovaných hodnôt energie na konci algoritmu. Túto oblasť by bolo potrebné ešte preskúmať, nakoľko aplikácie algoritmu šírenia aktivácie dosahujú vo všeobecnosti lepšie výsledky.

Faktory vplyvu na dosiahnuté výsledky

Skúsme teraz zhrnúť faktory, ktoré mohli ovplyvniť úspešnosť či neúspešnosť dosiahnutých výsledkov.

1. Kvalita objektov výučby. Niektoré sú koncipované príliš všeobecne a obsahujú priveľa informácií. Iné zasa obsahujú príliš málo textu a nie sú dostatočne naviazané na zvyšok kurzu. Je potrebné dbať na to, aby boli objekty výučby vytvárané ako znovupoužiteľné celky, nezviazané s konkrétnym kurzom.
2. Obmedzenia spracovania prirodzeného jazyka. Schopnosť 80,3%-nej lematizácie slovenčiny ubrala metóde určitý podiel znalostí o kurze, ktorý mohol byť využitý v prospech odhalenia vzťahov medzi konceptmi.
3. Správnosť referenčnej štruktúry kurzu. Nie je možné jednoznačne povedať, že nepodobnosť vytvoreného kurzu voči referenčnej štruktúre je zlým výsledkom.
4. Kritérium hodnotenia úspešnosti kurzu. Zvolené kritérium hodnotenia nemusí predstavovať tú charakteristiku, ktorú treba pri vyhodnocovaní úspešnosti tvorby doménového modelu sledovať.

Najvážnejšie problémy a ďalšie východiská

Pri preskúmaní úspešnosti generovania vzťahov pre jednotlivé koncepty sme identifikovali nasledujúce problémy:

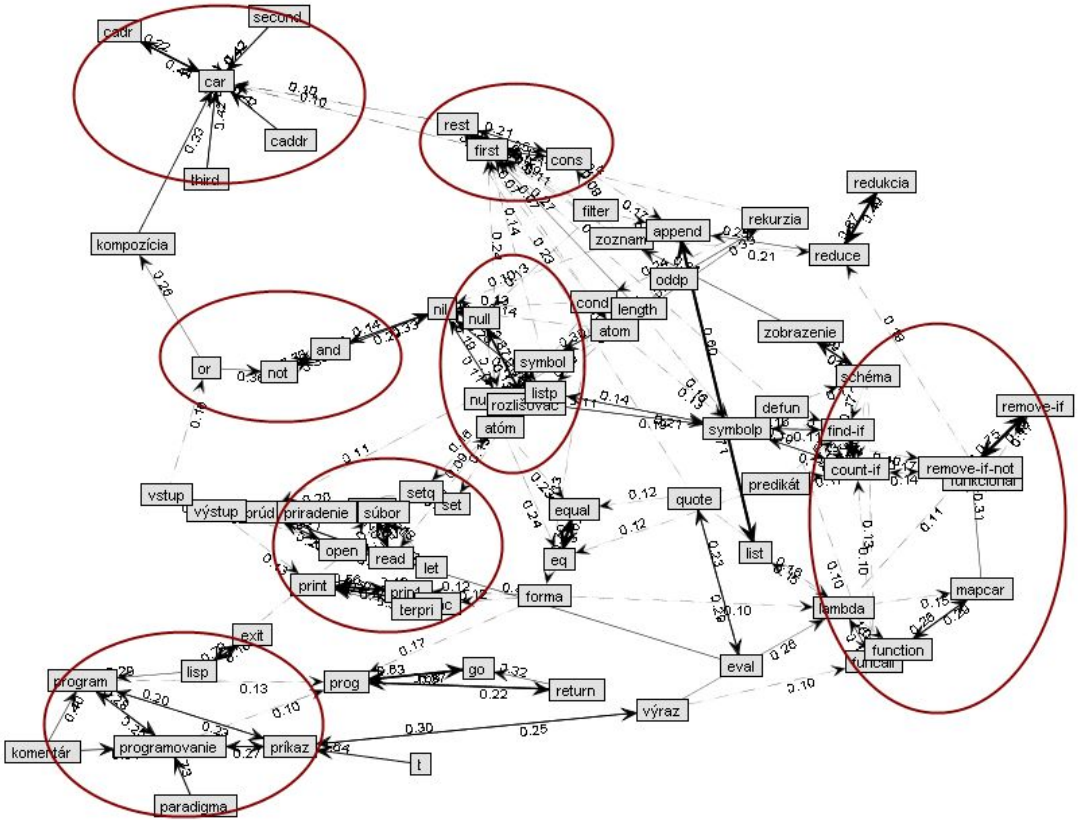
1. Nízku úspešnosť dosiahli koncepty reprezentované slovami, ktoré sú síce kľúčovými slovami, ale sú rovnomerne rozptýlené po celom texte (v tom horšom prípade sa používajú v rôznom kontexte). Nemusia sa vyskytovať ani príliš často a zapríčinia, že sú definované ako súvisiace s veľkým množstvom ďalších slov, čo negatívne vplýva na aplikáciu prístupov zakladajúcich napr. na šírené energie. V testovacej vzorke išlo napríklad o kľúčové slová *rekurzia*, *kompozícia*.
2. Problém s identifikáciou podobných konceptov majú koncepty reprezentované slovami sústredenými v malom počte malých (rozsahom) objektov výučby. Tieto kľúčové slová majú príliš silné väzby s ostatnými kľúčovými slovami nachádzajúcimi sa v daných objektoch výučby a je takmer nemožné dať ich do súvisu s inými pojmami. V testovacej vzorke išlo napríklad o skupinu kľúčových slov *CxR*. Tie vytvorili prijateľné väzby medzi sebou, avšak nedokázali nadviazať na ostatné kľúčové slová kurzu.

3. Najhoršie hodnoty testovaných metrík dosiahli koncepty reprezentované slovami, ktoré síce sú termínmi doménovej oblasti, ale sú príliš elementárne, tj. sú používané skoro všade. Ich relevanciu vo výučbových textoch vzhľadom na množstvo ich výskytu nedokázalo efektívne ovplyvniť ani využitie princípu inverznej frekvencie. V testovacej vzorke takými slovami boli napr. *COND*, *DEFUN*, *T*.

Východiská pre zlepšenie úspešnosti navrhnutej metódy vidíme hlavne v odstránení uvedených problémov. Je potrebné experimentovať s inými metrikami pre výpočet váh slov v texte alebo pouvažovať nad možnosťou využitia inej reprezentácie.

Problém súvisiaci s lematizáciou doménovo-špecifických termínov je možné čiastočne vyriešiť definovaním malého slovníka slov, ktoré sú dôležité pre doménovú oblasť a ten zakomponovať do procesu lematizácie.

Ďalšie možnosti vylepšenia sa ponúkajú spolu s využitím ďalších grafových algoritmov, akým je napríklad algoritmus markovskej centrality (White, Smith, 2003), alebo využitím niektorého z množstva zhlukovacích algoritmov. Už zo samotnej vizualizácie kurzu, ktorá je postavená na samorganizujúcich sa mapách, je zrejmé, že výskum v tejto oblasti by mohol priniesť nejaké ovocie (pozri obr. 8-1).



Obrázok 8-1: Automaticky vytvorený doménový model. Na obrázku sú zreteľne viditeľné zhluky podobných konceptov.

9 Zhodnotenie

Motiváciou tejto práce bola podpora procesu tvorby kurzov pre adaptívne systémy pre vzdelávanie, ktoré ako prezentačné médium využívajú web. Na začiatku sme analyzovali vlastnosti takýchto systémov a ich architektúru. Dôraz sme kládli na doménový model, ktorým sú reprezentované metadáta o kurze. Preskúmali sme spôsoby jeho modelovania vo viacerých adaptívnych systémoch pre vzdelávanie a odhalili súvislosti modelu s rozsahom a kvalitou prispôsobovania.

V ďalšej kapitole sme sa venovali už samotnému procesu tvorby kurzov. Súčasný stav sme zmapovali na príkladoch prístupov k podpore tvorby kurzov vo vybraných adaptívnych systémoch pre vzdelávanie. V takýchto systémoch najväčším problémom tvorby kurzov nie je napĺňanie výučbovým materiálom, ale tvorba „inteligentného obsahu“ – metadát, ktoré definujú štruktúru kurzu. Zistili sme, že aj napriek identifikovanému problému len málo z preskúmaných systémov využíva potenciál získavania znalostí a dolovania v dátach.

Po preskúmaní problematiky spracovania textu – od spôsobov reprezentácie dokumentov až po pokročilé jazykovo-špecifické prístupy – sme, kvôli nadväznosti na doménový model, analyzovali niekoľko grafových algoritmov. Z výsledkov analýzy vzišli ciele práce, z ktorých nosnou časťou bola potreba návrhu a overenia metódy automatizovaného vytvárania časti doménového modelu pomocou odhaľovania prepojení konceptov, ktoré reprezentujú prezentovaný obsah.

Navrhli sme riešenie, v ktorom sme sa snažili uplatniť čo najviac nadobudnutých poznatkov. Výsledkom je metóda automatizovaného získavania metadát o kurze, ktorej hlavnými krokmi sú:

1. predspracovanie objektov výučby,
2. extrakcia pseudokonceptov,
3. generovanie vzťahov a
4. konštrukcia doménového modelu.

Najväčším prínosom práce je rozpracovanie kroku *generovania vzťahov* do viacerých variantov, ktoré aplikujú rozličné prístupy (a ich kombinácie) k objavovaniu znalostí zviazaných s výučbovým obsahom kurzu.

Jednotlivé varianty sme overili v doméne výučby programovanie pomocou príkladov. Vykonali sme niekoľko experimentov, v ktorých sme sledovali vybrané charakteristiky vygenerovanej časti doménového modelu. Tú sme porovnali s manuálne zostrojenou referenčnou štruktúrou testovaného kurzu. Na základe výsledkov experimentov sme identifikovali príčiny, ktoré ovplyvnili úspešnosť

odhaľovania vzťahov v obsahu kurzu. Načrtli sme smery, ktorými je možné sa uberať pri snahe o zvýšenie kvality generovaného doménového modelu.

Ďalším z výstupov práce je autorský nástroj CourseDesigner, prostredníctvom ktorého je metóda integrovaná do prostredia portálu pre podporu výučby programovania na Fakulte informatiky a informačných technológií v Bratislave. Pomocou nástroja je možné vytvárať časť doménového modelu obsahujúcu metadáta o kurze, ktoré sa využívajú pri prispôsobovaní. CourseDesigner bol navrhnutý tak, aby zefektívnil tvorbu kurzov: obsahuje prívetivé používateľské prostredie, ktoré pokrýva všetky kroky tvorby štruktúry doménového modelu (nie však tvorbu samotných objektov výučby).

Z výsledkov experimentov, ako aj z poznatkov nadobudnutých štúdiom problémovej oblasti vzniklo niekoľko ideí, ako by sme mohli metódu vylepšiť. Možné vylepšenia sa dajú rozdeliť do troch oblastí. Prvou je optimalizácia predspracovania objektov výučby, kde by bolo potrebné preskúmať možnosti ich alternatívnej reprezentácie a vyskúšať riešenia, ktoré sa spočiatku zdali nevhodné (napr. latentná sémantická analýza). Druhá oblasť vylepšenia súvisí s integrovaním ďalších grafových algoritmov, akým je napríklad algoritmus markovskej centrality. Najväčší potenciál však tkvie v zhlukovacích algoritmoch, ktoré by bolo možné aplikovať na grafy zostrojené súčasnými variantmi metódy.

Aj keď výsledkom práce nie je objav prevratného algoritmu, prispeli sme k aktuálnemu stavu poznania v oblasti adaptívneho vzdelávania a tvorby kurzov. A možno sme pootvorili dvierka iným, efektívnejším prístupom.

Použitá literatúra

ALLAN, J., BALLESTEROS, L., et al. 1995. Recent Experiments with INQUERY. In *Fourth Text Retrieval Conference (TREC-4)*, 1995.

AMATI, G., VAN RIJSBERGEN, C. 2002. Probabilistic Models of Information Retrieval Based on Measuring the Divergence from Randomness. In *ACM Transactions on Information Systems*, Vol. 20(4), New York, October 2002, pp. 357-389.

BEAUMONT, I., BRUSILOVSKY, P. 1995. Adaptive educational hypermedia: From ideas to real systems. In MAURER, H. (Ed.). *Proceedings of ED-MEDIA'95 – World conference on educational multimedia and hypermedia*. Graz, Austria, 1995, pp. 93-98.

BIELIKOVÁ, M., NÁVRAT, P. Funkcionálne a logické programovanie. Vydavateľstvo STU: Bratislava, 2000. ISBN 80-227-1459-3. 281s.

BRIN, S., PAGE, L. 1998. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, pp. 107–117.

BRUSILOVSKY, P. 1992. The Intelligent Tutor, Environment and Manual for Introductory Programming. In *Educational Technology and Training International*, 1992, 29, vol. 1, pp.26-34.

BRUSILOVSKY, P. 1996. Methods and techniques of adaptive hypermedia. In *User Modeling and User-Adapted Interaction.* 1996, vol. 6, no. 2-3, pp. 87-129.

BRUSILOVSKY, P. SCHWARZ, E. 1997. Concept-based navigation in educational hypermedia and its implementation on WWW. In MÜLDNER, T., REEVES, T. C. (eds.) *Proceedings of ED-MEDIA/ED-TELECOM'97 - World Conference on Educational Multimedia/Hypermedia and World Conference on Educational Telecommunications.* Calgary, 1997, pp. 112-117.

BRUSILOVSKY, P. 2001. WebEx: Learning from examples in a programming course. In FOWLER, W. and HASEBROOK, J. (eds.) *Proc. of WebNet'2001, World Conference of the WWW and Internet*, Orlando, FL, AACE, pp. 124-129.

BRUSILOVSKY, P. 2003. Developing adaptive educational hypermedia systems: From design models to authoring tools. In MURRAY, T., BLESSING, S., AINSWORTH, S. (eds.): *Authoring Tools for Advanced Technology Learning Environment*. Dordrecht: Kluwer Academic Publishers, 2003, p. 377-409.

BRUSILOVSKY, P, et al. 2005. Interactive Authoring Support for Adaptive Educational Systems. In LOOI, et al. (eds.) *Proceedings of 12th International Conference on Artificial Intelligence in Education*, AIED'2005 (Amsterdam, July 18-22, 2005). Amsterdam: IOS Press, pp. 96-103.

BRUSILOVSKY, P., KNAPP, J., GAMPER, J. 2006. Supporting teachers as content authors in intelligent educational systems. In *International Journal of Knowledge and Learning,* 2006, vol. 2, no. 3/4, pp. 191–215.

CEGLOVSKY, M., COBURN, A., CUADRADO, J. *Semantic Search of Unstructured Data using Contextual Network Graphs*. 2003.

CRISTEA, A. I., OKAMOTO, T. 2001. Object-oriented Collaborative Course Authoring Environment supported by Concept Mapping in MyEnglishTeacher. In Educational Technology & Society. ISSN 1436-4522. Vol. 4, no. 2, 2001.

CRISTEA, A. I., AROYO, L. 2002. Adaptive Authoring of Adaptive Educational Hypermedia, In *Proceedings of AH 2002, Adaptive Hypermedia and Adaptive Web-Based Systems*, LNCS 2347, Springer, pp. 122-132.

CRISTEA, A. I., DE MOOIJ, A. 2003. Adaptive Course Authoring: My Online Teacher. In Proc. of 10th Int. *Conference on Telecommunications, ICT 2003*. Vol. 2, pp. 1762-1769.

CRISTEA, A. I., DE MOOIJ, A. 2003b. LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators. In *Proceedings of WWW'03,* Budapest, Hungary, ACM, IW3C2.

CRISTEA, A. I., DE MOOIJ, A. 2003c. Designer Adaptation in Adaptive Hypermedia Authoring. In *Proceedings of the International Conference on Information Technology: Computers and Communications ITCC'03*. Las Vegas,US, IEEE Computer Science, 2003, pp. 444-448.

CRISTEA, A. I. 2004. Is Semi-automatic authoring of adaptive educational hypermedia possible? In *Special Issue on "Innovations in Advanced Technology for Learning: Authoring for Adaptive Educational Hypermedia"*, *Journal of Advanced Technology for Learning*, vol. 1, 2004.

CRISTEA, A.I. 2004b. Authoring of adaptive and adaptable educational hypermedia: where are we now and where are we going? In *Proceedings of the IASTED International Conference Web-Based Education*, February 16-18, 2004, Insbruck, Austria.

CRISTEA, A., et al. 2005. *Evaluation of Interoperability of Adaptive Hypermedia Systems: testing the MOT to WHURLE conversion in a classroom setting*. Workshop paper at A3EH, AEID, Amsterdam, Netherlands, July 19, 2005.

CRISTEA, A., SMITS, D., DE BRA, P. 2005. *Writing MOT, Reading AHA! - converting between an authoring and a delivery system for adaptive educational hypermedia*. Workshop paper at A3EH, AEID, Amsterdam, Netherlands, July 19, 2005.

DE BRA, P., CALVI, L. 1998. AHA: a Generic Adaptive Hypermedia System. In *Proc. of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98*. Pittsburgh, USA.

DE BRA, P., et al. 2007. Authoring and Management Tools for Adaptive Educational Hypermedia Systems: The AHA! Case Study. In *Studies in Computational Intelligence (SCI),* no. 62, pp. 285-308, Springer Verlag, 2007.

GARABÍK, R. 2006. Slovak morphology analyzer based on levenshtein edit operations. In *Proceedings of 1st Workshop on Inteligent and Knowledge Oriented Technologies*, WIKT 2006, Bratislava, Slovakia, 2006, pp. 2-5.

GRIGORIADOU, M., et al. 2001. INSPIRE: an intelligent system for personalized instruction in a remote environment. In *Proceedings of 3rd Workshop on Adaptive Hypertext and Hypermedia* (Sonthofen, Germany, 2001), pp. 13-24.

GRIGORIADOU, M., PAPANIKOLAOU, K. 2006. Authoring Personalised Interactive Content. In *First International Workshop on Semantic Media Adaptation and Personalization* (SMAP'06), 2006, pp. 80-85.

KNAPP, J., GAMPER, J., BRUSILOVSKY, P. 2004. Reuse of lexicographic examples in a web-based learners' dictionary. In NALL, J., ROBSON, R. (eds.) *Proceedings of World Conference on E-Learning, E-Learn 2004*, Washington, DC, USA, pp. 776–783.

KRAJČI, S., NOVOTNÝ, R. 2006. Hľadanie základného tvaru slovenského slova na základe spoločného konca slov. In *Proceedings of 1st Workshop on Inteligent and Knowledge Oriented Technologies*, WIKT 2006, Bratislava, Slovakia, 2006, pp. 99-101.

KUROPKA, D. 2004. *Modelle zur Repräsentation natürlichsprachlicher Dokumente: Ontologie-basiertes Information-Filtering und -Retrieval mit relationalen Datenbanken*. Logos Verlag: Berlin, 2004. 264 s. ISBN 978-3-8325-0514-1.

MORAVČÍK, M. 2006. *Modelovanie adaptívnych webových systémov.* Bratislava, 2006. 74 s. Diplomová práca na FIIT STU v Bratislave. Vedúca práce prof. Mária Bieliková.

PÁLEŠ, E. 1994. *Sapfo: parafrázovač slovenčiny.* Bratislava: Veda, 1994. 305 s. ISBN 80-224-0209-9.

PORTER, M. F. 1980. An Algorithm for Suffix Stripping. In *Program*, vol. 14, n.3, july 1980, pp. 130-137.

SALTON, G., WONG, A., YANG, C. S. 1975. A Vector Space Model for Automatic Indexing. In *Communications of the ACM*, 1975, vol. 18, n. 11, pp. 613–620.

SALTON, G., BUCKLEY, C. 1988. Term weighting approaches in automatic text retrieval. In *Information Processing and Management*, vol. 24(5). Pergamon Press, Inc., Tarrytown, NY, USA, 1988, pp. 513-523.

SOSNOVSKY, S., BRUSILOVSKY, P., YUDELSON, M. 2004. Supporting Adaptive Hypermedia Authors with Automated Content Indexing. In *Proc. of Second Int. Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia at the Third Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems (AH'2004),* Eindhoven, Netherlands, 2004.

ŠIMÚN, M. 2008. *Personalizovaný prístup k používateľovi v prostredí e-vzdelávania*. Bratislava, 2008. 58 s. Diplomová práca na FIIT STU v Bratislave. Vedúci práce Ing. Anton Andrejko.

WHITE, S., SMITH, P. 2003. Algorithms for estimating relative importance in networks. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 2003, pp. 266 - 275.

Príloha A: Technická dokumentácia

A.1 Špecifikácia vstupných dát projektu PeWePro

Kurz projektu PeWePro je definovaný adresárom s nasledujúcimi súbormi:

```
authors.xml   .............   definícia autorov
                              participujúcich na tvorbe kurzu

concepts.xml   ............   definícia konceptov, ktoré sú
                              súčasťou kurzu

conceptsRelations.xml   ....  definícia vzťahov medzi
                              konceptmi

courses.xml   .............   definícia kurzu

learningObjects.xml   ......  definícia a obsah výučbových
                              objektov kurzu

learningObjectsRelations.xml  definícia vzťahov medzi
                              výučbovými objektmi

structures.xml  ............  definícia štruktúry kurzu
```

Z nášho pohľadu je zaujímavý súbor `learningObjects.xml` obsahujúci výučbové objekty ktoré sú predmetom spracovania navrhnutou metódou. Špecifikácia ostatných súborov (vrátane súboru `learningObjects.xml`) je dostupná v podobe XML schém na priloženom dátovom nosiči.

`learningObjects.xml`

Súbor pozostáva z koreňového elementu `learningObjects`, ktorý môže obsahovať nasledujúce elementy podľa typu objektu výučby:

- `explanation` – obsahujúce vysvetlenie učiva;
- `excercise` – obsahujúce cvičenia;
- `simple` – obsahujúce sprievodný text ku kurzu;
- `template` – obsahujúci programové schémy (už nepoužívaný).

Ako potomkovia uvedených elementov sú v hierarchii XML definované elementy:

- `definition` – obsahuje nosnú textovú časť výučbového objektu, príp. definíciu problému,
- `hint` – obsahuje pomôcku k vyriešeniu problému (je definový len pre výučbový objekt typu `exercise`),
- `solution` – obsahuje riešenie problému (je definový len pre výučbový objekt typu `exercise`).

Na špecifikáciu potomkov týchto elementov je využitá notácia jazyka DocBook[17]. Ukážka výučbového objektu je zobrazená na obr. 8-1.

```
<explanation name="Funkcia FIRST" id="flpbook-2.2.1">
   <definition>
      <para xmlns="http://docbook.org/ns/docbook">
         Funkcia <code>FIRST</code> vráti prvý prvok zoznamu.
         V príkladoch, pri zápise zoznamu ako argumentu funkcie,
         niekedy budeme používať apostrof. Vzhľadom na to, že
         vysvetlenie by na tomto mieste bolo veľmi ťažkopádne
         a nevhodné, nateraz si to nevšímajte. K vysvetleniu sa
         vrátime neskôr v tejto kapitole.
      </para>
      <para xmlns="http://docbook.org/ns/docbook">
         Tu je niekoľko príkladov použitia funkcie FIRST:
         <informalexample>
            <programlisting>
                * (FIRST '(7 2 14))   ;prvý prvok zoznamu (7 2 14)
                7
                * (FIRST '((hong kong) (ping pong) bang))
                (hong kong)
            </programlisting>
         </informalexample> Nemá zmysel aplikovať funkciu
         <code>FIRST</code> na prázdny zoznam alebo na atóm. Takže
         výrazy <code>(FIRST ())</code> a <code>FIRST</code> s
         argumentom typu atóm, napr. <code>(FIRST 18)</code>, nie
         sú definované.
      </para>
   </definition>
</explanation>
```

Obrázok A-1: Ukážka vstupých dát

Dôraz pri vytváraní objektov výučby je potrené klásť na oddelenie častí kódu od častí prirodzeného jazyka. Predspracovanie objektov výučby je potom efektívnejšie, pretože môžeme získať detailnejšie znalosti o spracovávanom obsahu.

17 http://www.docbook.org/

A.2 Heuristiky predspracovania objektov výučby

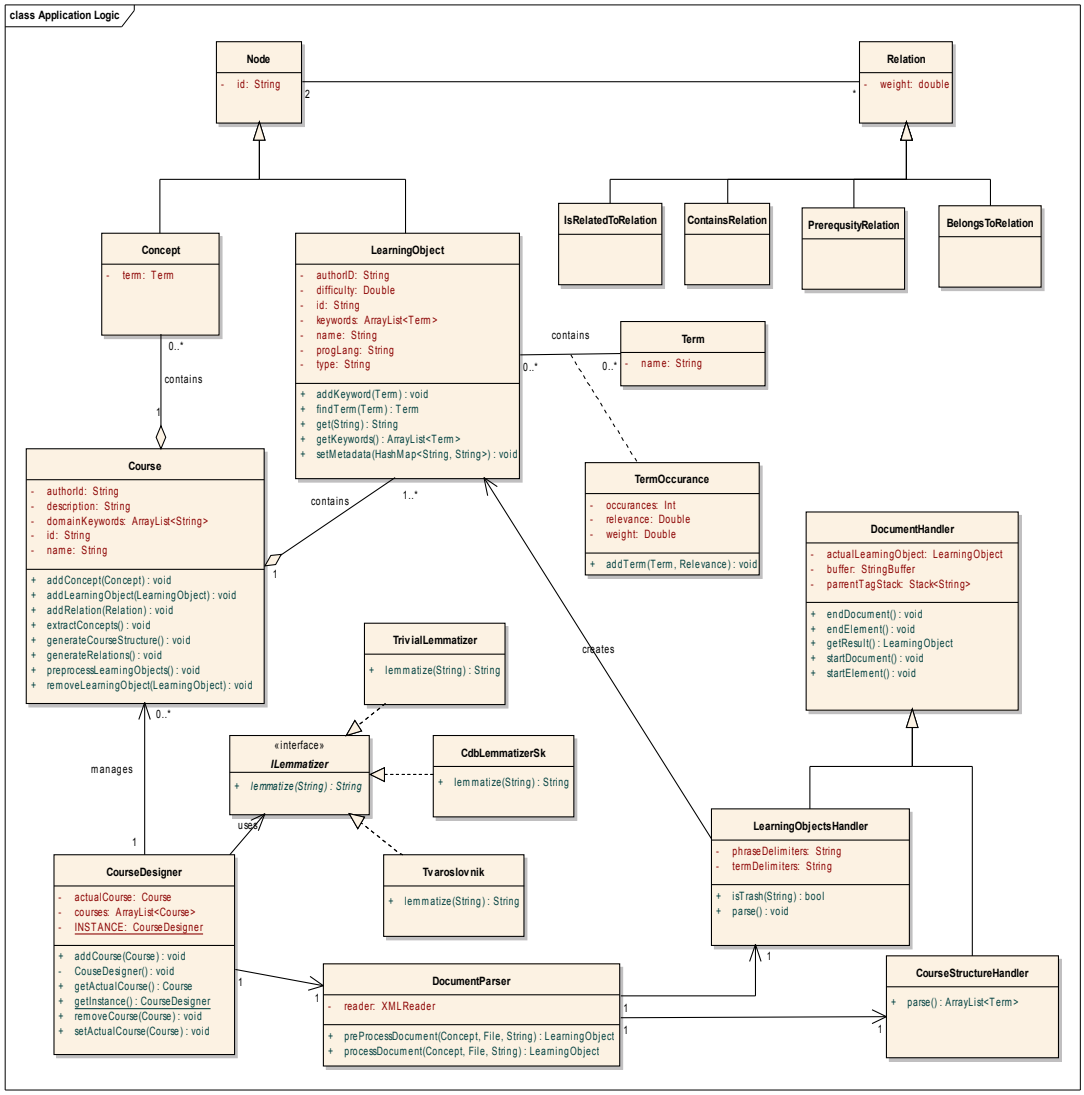
```
if (ti isDomainKeyword)
then
    relevance(ti) *= 5;

if (xmlHierarchy contains „explanation“ and
    xmlHierarchy contains „code“)
then
    relevance(ti) *= 2.5;


if (xmlHierarchy contains „explanation“ and
    xmlHierarchy contains „programlisting“)
then
    relevance(ti) *= 1.1;


if (xmlHierarchy contains „simple“ and
    xmlHierarchy contains „code“)
then
    relevance(ti) *= 3;


if (xmlHierarchy contains „simple“ and
    xmlHierarchy contains „programlisting“)
then
    relevance(ti) *= 1.5;


if (xmlHierarchy contains „excercise“ and
    xmlHierarchy contains „code“)
then
    relevance(ti) *= 1.2;


if (xmlHierarchy contains „exercise“ and
    xmlHierarchy contains „programlisting“)
then
    relevance(ti) *= 0.5;


if (xmlHierarchy contains „emphasis“)
then
    relevance(ti) *= 2;
```

```
if (xmlHierarchy contains „title“)
then
    relevance(t_i) *= 2.5;
```

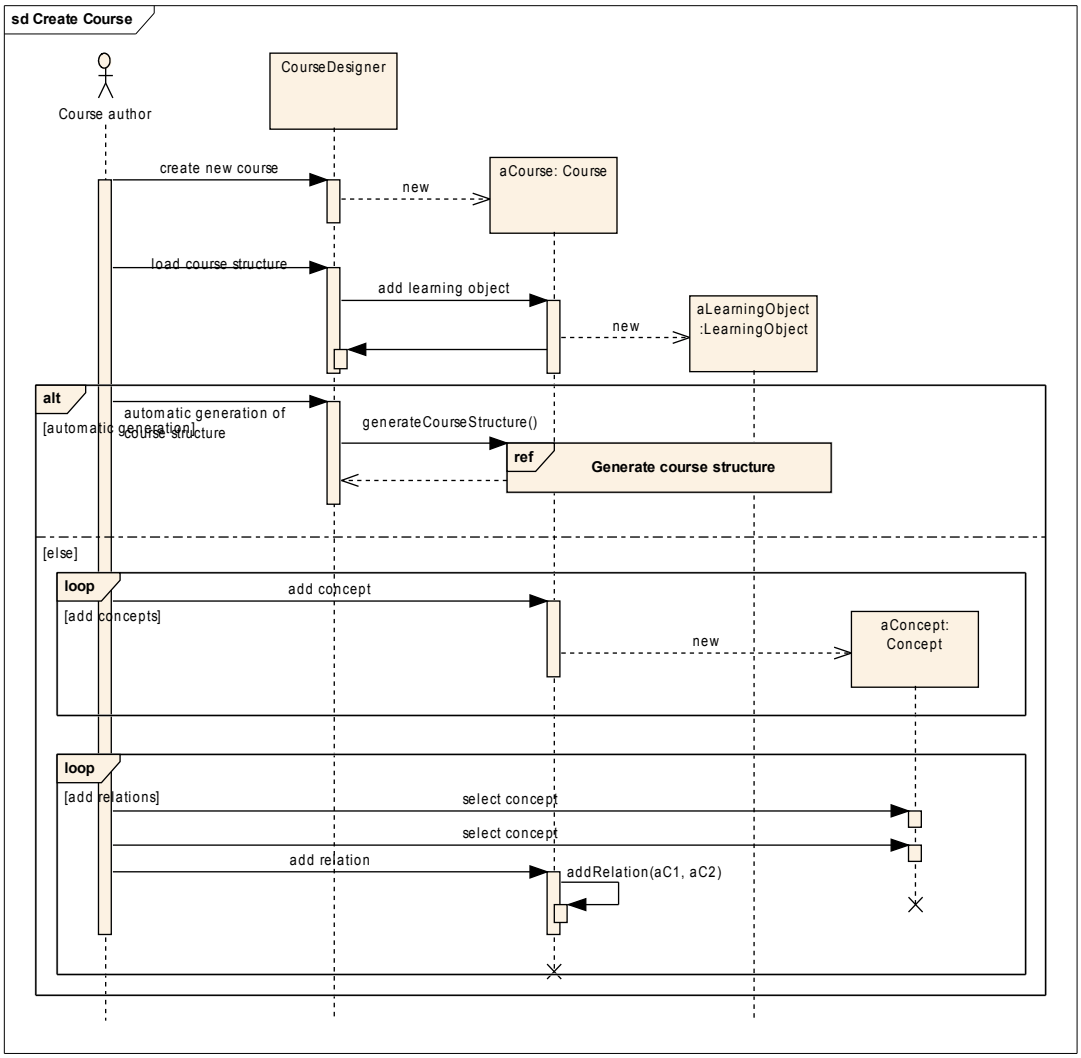
```
if (xmlHierarchy contains „listitem“)
then
    relevance(t_i) *= 1.1;
```

A.3 Návrh nástroja CourseDesigner

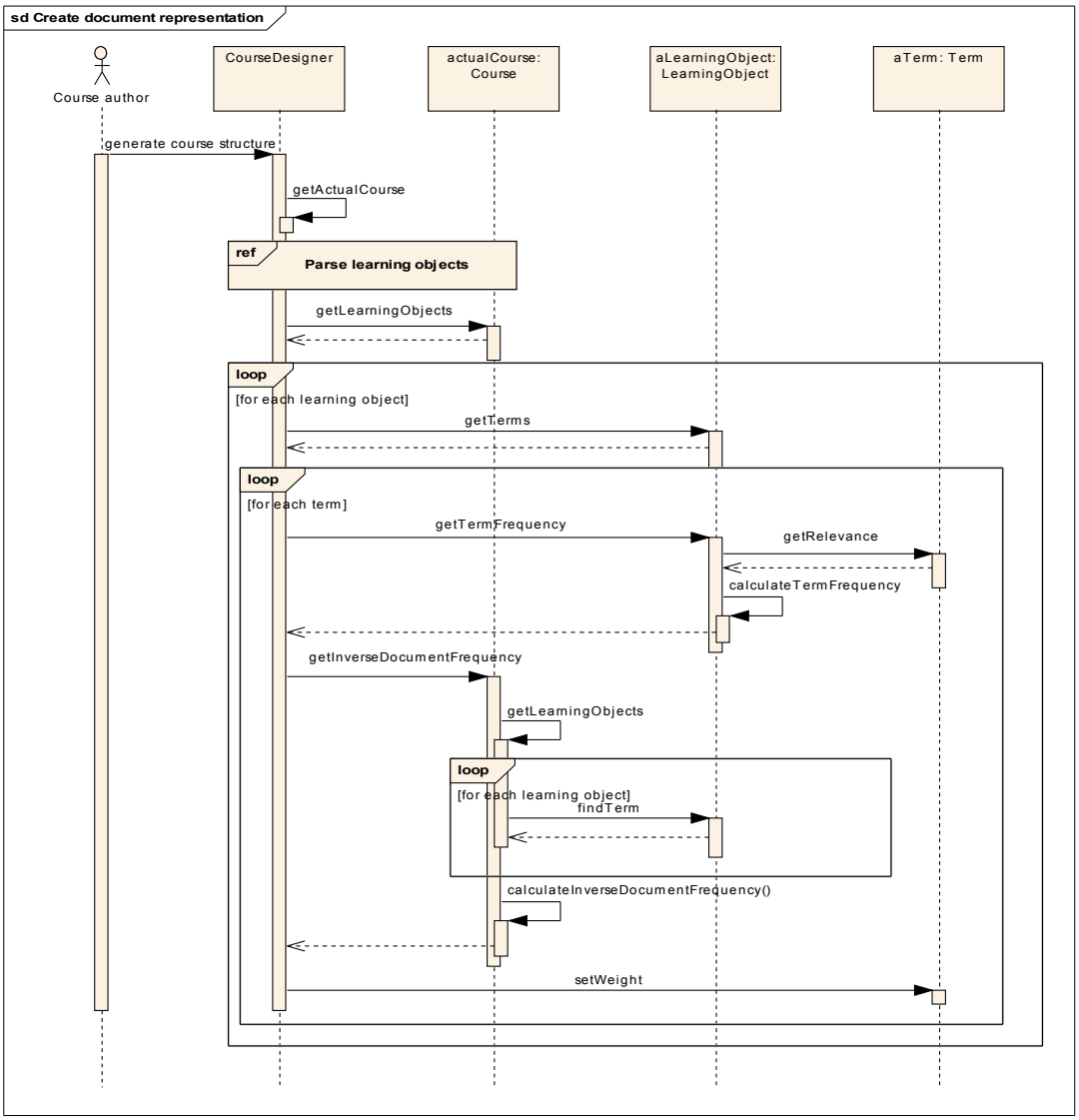
V tejto časti uvádzané základné UML diagramy opisujúce vybrané vlastnosti vytvoreného nástroja.



Obrázok A-2: Diagram tried základnej aplikačnej logiky nástroja CourseDesigner



Obrázok A-3: Sekvenčný diagram vytvárania kurzu.



Obrázok A-4: Sekvenčný diagram pre tvorbu reprezentácie objektov výučby.

Príloha B: CourseDesigner – používateľská príručka

B.1 Požiadavky na systém

Softvérové požiadavky

Nástroj CourseDesigner je jednoduchá „desktopová“ aplikácia, ktorá nemá zvýšené softvérové nároky. Keďže bola vyvinutá na platforme Java, je potrebné mať nainštalované prostredie pre spúšťanie aplikácií (JRE; Java Runtime Environment) vo verzii 6.

Hardvérové požiadavky

Odporúčané hardvérové požiadavky sú uvedené v nasledujúcej tabuľke:

Miesto na HDD	50 MB
Veľkosť RAM	1024 MB
Frekvencia CPU	2 GHz

Mierne zvýšený nárok na diskovú kapacitu je spôsobený prítomnosťou konštantnej databázy pre lematizáciu slovenského jazyka. Pre plynulé vykresľovanie grafu (najmä pre rozsiahle kurzy) odporúčame procesor pracujúci na frekvencii minimálne 2 GHz.

B.2 Inštalácia a spustenie

Nástroj CourseDesigner je distrubuovaný v archíve Zip s názvom CourseDesigner-*xxx*.zip, ktorý je potrebný dekomprimovať na želané miesto na disku. Časť *xxx* označuje verziu nástroja. Štruktúra archívu je nasledovná:

```
..
lib
courses
misc
CourseDesigner.jar
Readme.txt
```

V adresári `lib` sa nachádzajú externé knižnice potrebné pre fungovanie nástroja. Adresár `courses` slúži na uchovávanie kurzov (nástroj v základnej

konfigurácii obsahuje súborovú štruktúru kurzu funkcionálneho programovania). V adresári `misc` sú ostatné dáta potrebné pre beh programu: lematizátory, zoznamy stop slov, a pod. Obsah adrasárov `lib` a `misc` by používateľ nemal meniť.

Súbor CourseDesigner.jar je spustiteľný Java archív celej aplikácie. Spustenie vykonáme v konzole pomocou príkazu:

```
>java -jar "CourseDesigner.jar"
```

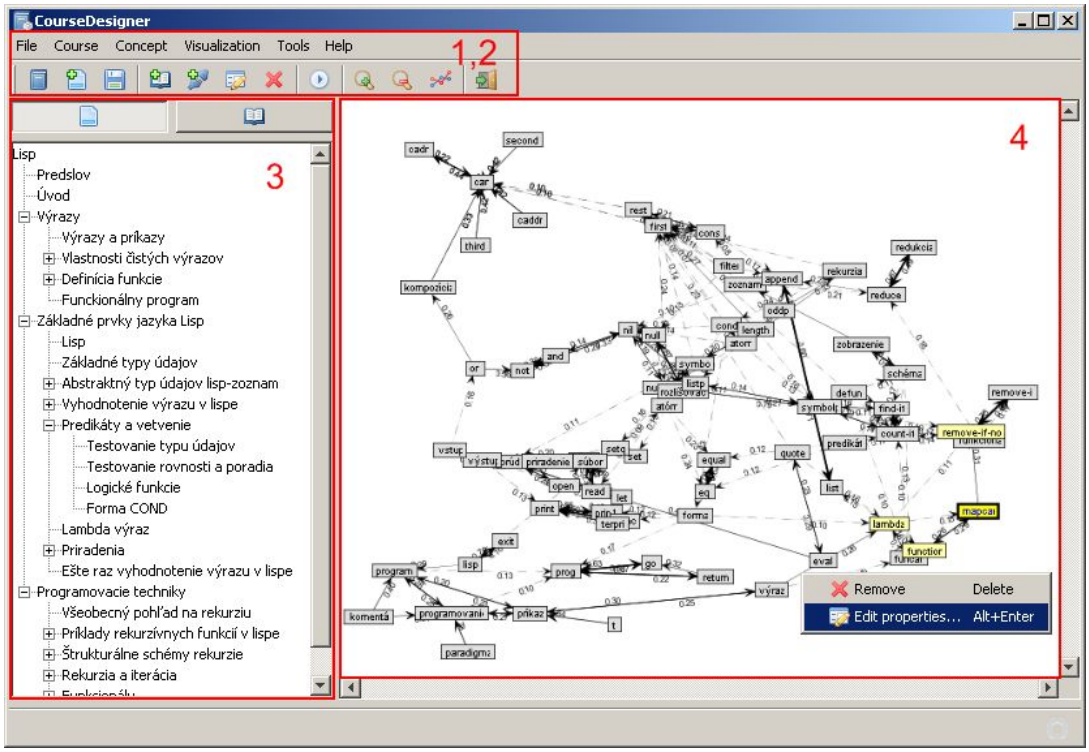
Za predpokladu, že je v systéme korektne nainštalované prostredie java, zobrazí sa hlavné okno nástroja CourseDesigner.

Súbor Readme.txt obsahuje základné informácie o nástroji, ako aj informácie o spustení.

B.3 Hlavné okno a rozmiestnenie ovládacích prvkov

Hlavné okno nástroja pozostáva zo štyroch základných častí (pozri obr. B-1):

- hlavné menu (1),
- lišta nástrojov (2),
- panel objektov výučby a konceptov (3),
- interaktívna vizualizácia kurzu (4).



Obrázok B-1: Hlavné okno nástroja CourseDesigner.

Hlavné menu

Lišta s hlavným menu obsahuje tieto menu:

1. File
2. Course
3. Concept/Relation
4. Visualization
5. Tools
6. Help

Jednotlivé položky uvedených menu sú opísané v tabuľkách B-1 až B-5.

Tabuľka B-1: Menu File

Ikona	Názov položky	Preklad	Účel
	New course	Nový kurz	Vytvorenie nového prázdneho kurzu[18]
	Add learning objects	Pridanie objektov výučby	Načítanie objektov výučby a ich hierarchickej štruktúry
	Save course	Uloženie kurzu	Uloženie kurzu do špecifikovaných súborov
	Exit	Ukončenie práce	Ukončenie práce a opustenie programu

Tabuľka B-2: Menu Course

Ikona	Názov položky	Preklad	Účel
	Add concept	Pridanie konceptu	Pridanie konceptu do kurzu
	Add relation	Pridanie relácie	Pridanie relácie do kurzu
	Generate course structure	Generovanie štruktúry kurzu	Spustenie aktuálne povoleného kroku generovania štruktúry kurzu. Povolený krok je určený stavom, v ktorom sa kurz nachádza. Možné kroky generovania sú prístupné aj samostatne prostredníctvom ďalších položiek menu
–	Preprocess learning objects	Predspracovanie objektov výučby	Predspracovanie objektov výučby a vytvorenie vnútornej reprezentácie
–	Extract concepts	Extrakcia konceptov	Extrakcia konceptov z predspracovaných výučbových objektov
–	Generate relations	Generovanie vzťahov	Generovanie vzťahov medzi konceptmi podľa zvoleného variantu

18 V tejto časti používame pojem kurz na označenie tej časti doménového modelu, ktorá je pomocou nástroja vytváraná.

Tabuľka B-3: Menu Concept / Relation

Ikona	Názov položky	Preklad	Účel
	Remove	Odstránenie	Odstránenie označeného konceptu / relácie (výber na základe označenia)
	Edit properties	Úprava vlastností	Vyvolanie dialógového okna pre úpravu vlastností konceptu / (výber na základe označenia)

Tabuľka B-4: Menu Visualization

Ikona	Názov položky	Preklad	Účel
	Reset graph layout	Rekonštrukcia rozloženia uzlov	Rekonštrukcia vykreslenia grafu kurzu (v prípade, že sa uzly prekrývajú a pod.)
	Zoom in	Zväčšenie	Zväčšenie grafu, priblíženie
	Zoom out	Zmenšenie	Zmenšenie grafu, oddialenie

Tabuľka B-5: Menu Tools

Ikona	Názov položky	Preklad	Účel
–	Load domain keywords	Načítanie kľúčových slov doménovej oblasti	Načítanie kľúčových slov doménovej oblasti z externého súboru
–	Edit domain keywords	Úprava kľúčových slov doménovej oblasti	Vyvolanie dialógového okna pre úpravu kľúčových slov doménovej oblasti
–	Configure	Konfigurácia	Konfigurácia nástroja

Lišta s nástrojmi

Lišta nástrojov zjednodušuje prístup k najčastejšie používaným funkciám celej aplikácie prostredníctvom prehľadných ikon. Je rozdelená do piatich funkčných častí, kde jednotlivé časti pokrývajú (pozri obr. B-2):

1. vstupno-výstupné procesy,
2. prácu s konceptmi a reláciami,
3. automatizáciu tvorby kurzu,
4. vizualizáciu kurzu,
5. ukončenie programu.

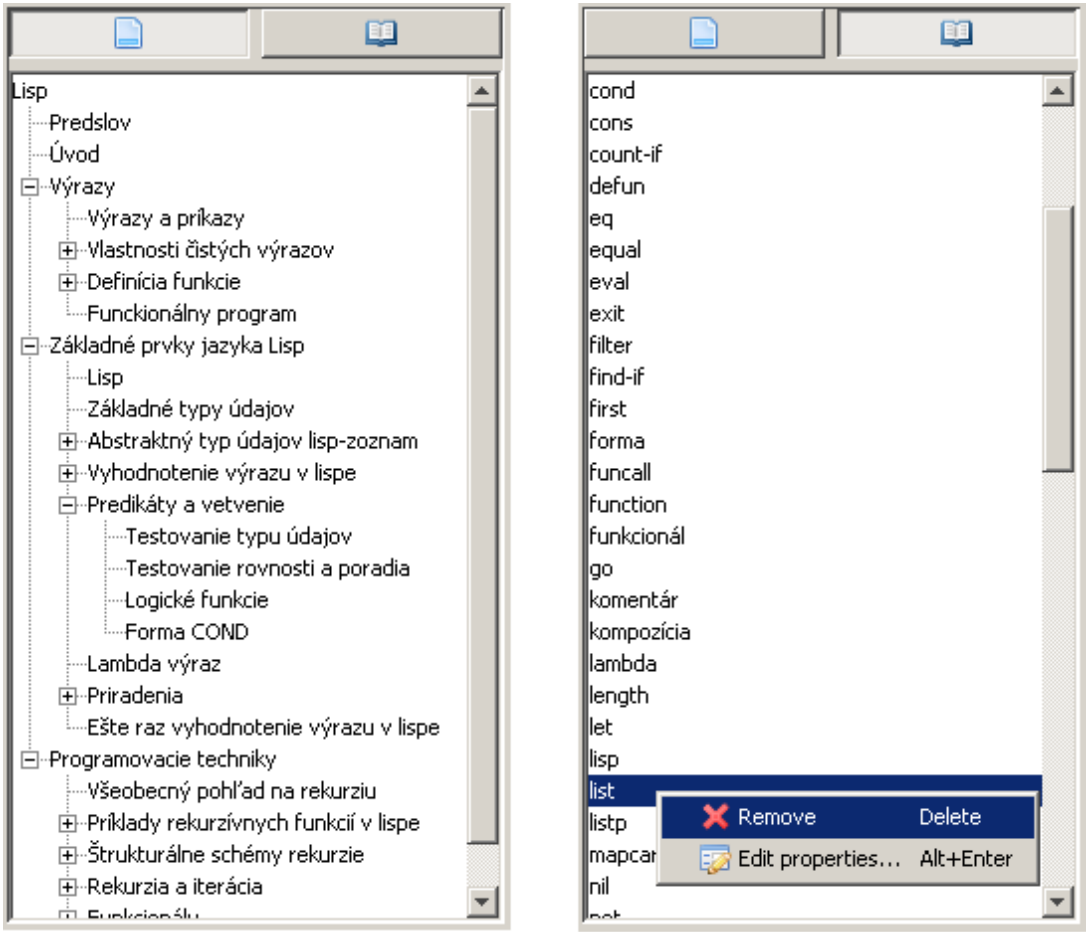
Panel objektov výučby a konceptov

V paneli objektov výučby a konceptov je možne prepínať medzi zobrazením hierarchickej štruktúry kurzu a zoznamom konceptov (pozri obr. B-3).

Cez tento panel pristupujeme k základným elementom kurzu: výučbovým objektom a konceptom. Po označení elementu môžeme kliknutím pravého tlačidla myši vyvolať vyskakovacie menu, ktoré ponúka možnosti odstránenia alebo úpravy vlastností výučbového objektu alebo konceptu.

Obrázok B-2: Lišta nástrojov.

Panel objektov výučby je plne synchronizovaný s grafovou vizualizáciou kurzu. Po označení objektu výučby sa v grafe zvýraznia všetky koncepty, pre ktoré existuje relácia s označeným objektom výučby. Po označení konceptu sa vo vizualizácii zvýrazní samotný koncept a tiež jeho susedia (pozri obr. B-11).



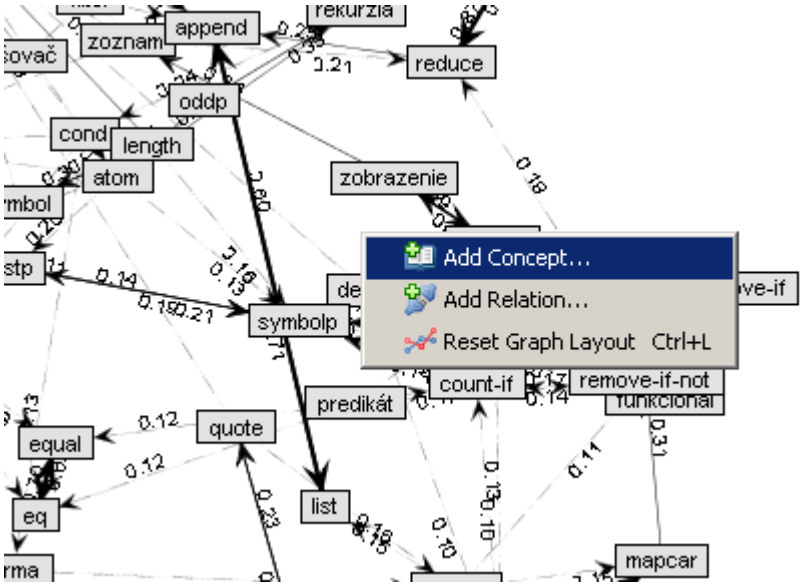
Obrázok B-3: Panel výučbových objektov a koncpetov.

Vizualizácia kurzu

Kurz je vizualizovaný interaktívnym grafom, ktorý prostredníctvom myši a vyskakovacích okien poskytuje nasledujúcu funkcionalitu:

- súvisiacu s kurzom:
  - ◇ pridanie, odstránenie, úprava vlastností konceptov;
  - ◇ pridanie, odstránenie, úprava vlastností relácií;
- súvisiacu s vizualizáciou:
  - ◇ presúvanie konceptov;
  - ◇ zväčšenie / zmenšenie grafu;
  - ◇ rekonštrukcia rozloženia konceptov;

Funkcie súvisiace *odstraňovaním* a *úpravou vlastností* konceptov, resp. relácií sú realizované pomocou vyskakovacích okien, ktoré pracujú nad označeným elementom v grafe. Ak myšou klikneme na prázdne miesto, vyvoláme vyskakovacie okno s ponukou *pridania* konceptu alebo relácie (pozri obr. B-4).



Obrázok B-4: Pridanie nového konceptu pomocou vyskakovacieho okna.

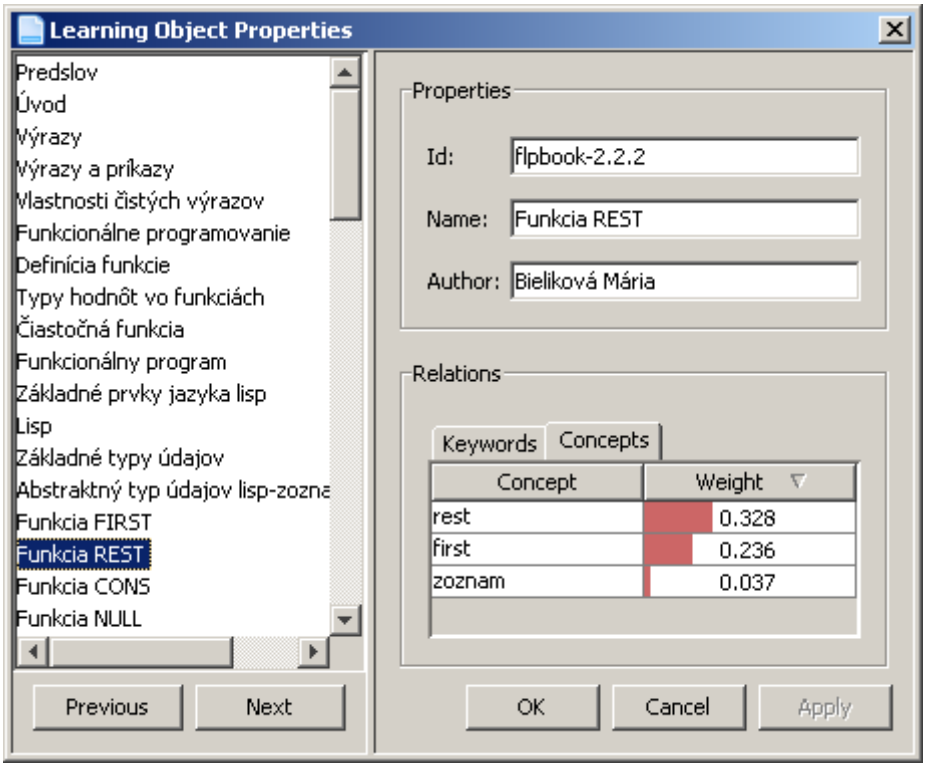
Opis funkcií súvisiacich s vizualizáciou kurzu je v časti B.6.

B.4 Dialógové okná jednotlivých elementov kurzu

V kurze pracujeme s tromi elementmi kurzu: výučbovými objektmi, konceptmi a reláciami. Pre každý z nich existuje dialógové okno pre úpravu ich vlastností.

Dialógové okno objektov výučby

Dialógové okno objektov výučby umožňuje zobraziť a zmeniť vlastnosti objektov výučby (pozri obr. B-5).



Obrázok B-5: Dialógové okno pre úpravu vlastností výučbového objektu.

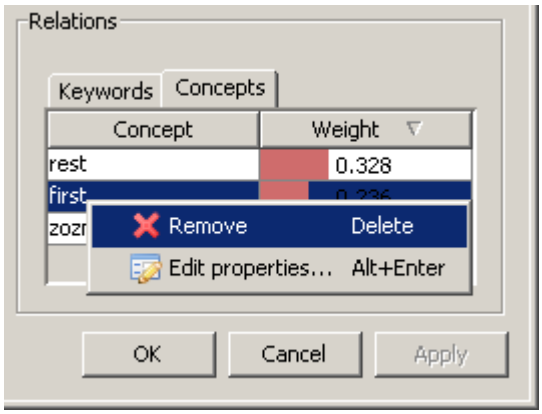
Dialógové okno sa skladá z troch základných častí:

- zoznam objektov výučby (vľavo),
- metadáta o objekte výučby (vpravo hore),
- relácie objektu výučby (vpravo dole).

V tomto okne je možné upravovať vlastnosti výučbových objektov a relácií medzi označeným objektom výučby a susediacimi konceptmi. Zmena váh relácií je realizovaná pomocou interaktívneho komponentu, v ktorom je váha zobrazená percentuálne ako farebný obdĺžnik z celého poľa tabuľku. Aby chceme zmeniť váhu relácie, stačí kliknúť myšou na požadované miesto v komponente a váha sa prispôsobí bez toho, aby sme museli vypĺňať číslo.

Vykonané zmeny v dialógovom okne sú synchronizované s kurzom až po potvrdení ich uloženia do kurzu pomocou tlačidiel *OK*, resp. *Apply*.

Pre dosiahnutia pohodlia pri úprave kurzu je možné z tohto okna vyvolať dialógové okno pre úpravu konceptov a relácií (pozri obr. B-6).

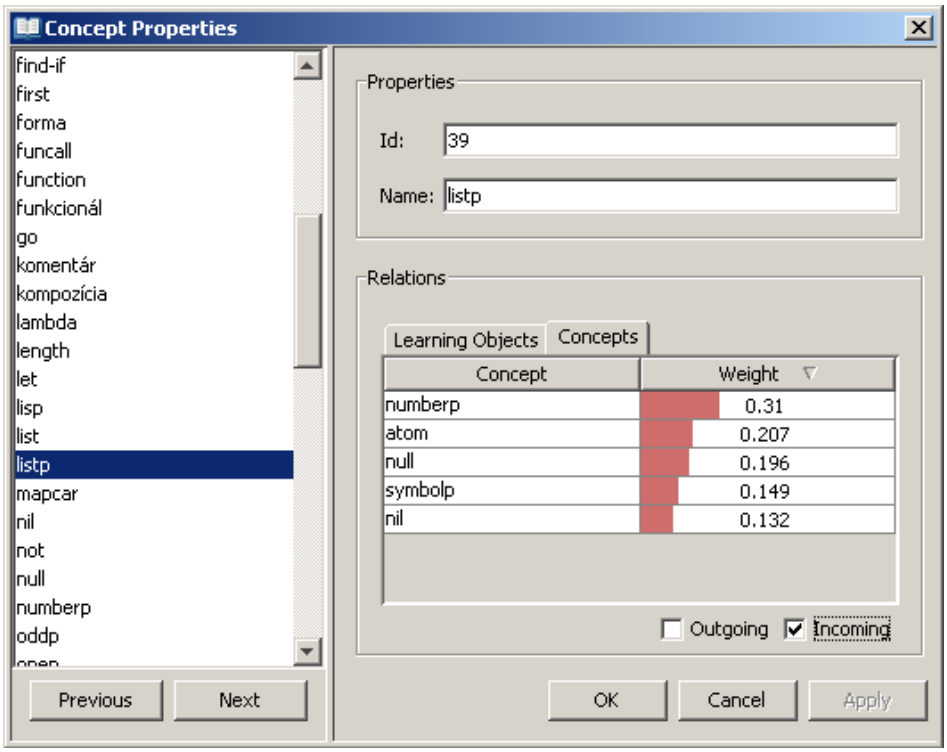


Obrázok B-6: Vyskakovanie menu pre označenú reláciu.

V časti relácií výučbového objektu je možné zobraziť nielen relácie s konceptmi, ale tiež relácie k kľúčovými slovami, ktoré vznikli pri predspracovania výučbových objektov v procese automatizovanej tvorby kurzu.

Dialógové okno konceptov

Dialógové okno pre úpravu vlastností konceptov je koncipované v podobnom duchu, ako jeho predchodca (obr. B-7).



Obrázok B-7: Dialógové okno pre úpravu vlastností konceptu.

Dialógové okno sa skladá z troch základných častí:

- zoznam konceptov (vľavo),
- metadáta o koncepte (vpravo hore),
- relácie konceptu (vpravo dole).

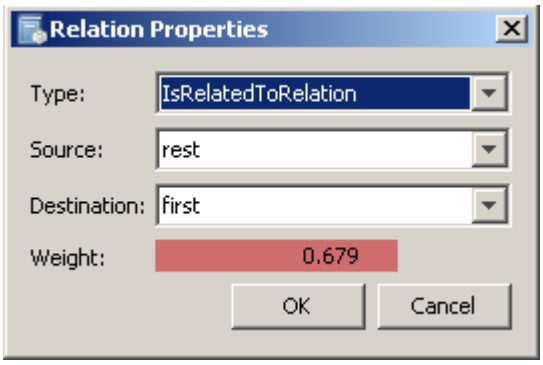
V tomto okne je možné upravovať vlastnosti konceptov a relácií medzi nimi. Zmena váh relácií je opäť realizovaná pomocou interaktívneho komponentu, v ktorom len stačí kliknúť myšou na požadované miesto v komponente a váha sa prispôsobí bez toho, aby sme museli vypĺňať číslo.

Na rozdiel od predchádzajúceho dialógové okna, tu môžeme zvoliť aj orientovanosť relácií, ktoré chceme zobraziť (zaškrtávacie políčka *Outgoing, Incoming*).

Vykonané zmeny v dialógovom okne sú synchronizované s kurzom až po potvrdení ich uloženia do kurzu pomocou tlačidiel *OK*, resp. *Apply*.

Dialógové okno relácií

Dialógové okno pre zmenu vlastností relácie je jednoduchšie ako v predchádzajúcich prípadoch. Umožňuje zmeniť zdroj relácie, cieľ relácie, váhu relácie a jej typ (pozri obr. B-8).



Obrázok B-8: Dialógové okno pre úpravu vlastností relácie.

Zoznam dostupných zdrojových a cieľových elementov (konceptov, príp. objektov výučby) sa dynamicky mení s výberom typu relácie. Je možné vybrať zo štyroch typov relácií:

- IsRelatedToRelation
- BelongsToRelation
- ContainsRelation
- Prerequisity

Z tohto okna je možné vyvolať dialógové okná pre zobrazenie vlastností konceptov, príp. objektov výučby kliknutím pravého tlačidla myši na „combobox“.

B.5 Práca s kurzom

V tejto časti si vysvetlíme jednotlivé funkcie nástroja CourseDesigner na príklade tvorby kurzu. Najprv načítame objekty výučby a potom sa môžeme rozhodnúť, či kurz chceme vytvárať manuálne alebo automatizovane.

Načítanie objektov výučby

Načítanie objektov výučby zrealizujeme pomocou príkazu *Add learning objects*. Tento príkaz môžeme zadať z:

- hlavného menu: *File -> Add learning objects*
- lišty nástrojov kliknutím na ikonu 
- pomocou klávesovej skratky *Ctrl + O*.

Pri načítaní objektov výučby sme vyzvaní k voľbe adresára, ktorý obsahuje špecifickú súborovú štruktúru. Z tohto adresára je načítaný súbor s výučbovými objektmi.

Načítanie objektov výučby je možné len na začiatku tvorby kurzu. Kým nevytvoríme prvý koncept (manuálna tvorba kurzu) alebo nespustíme predspracovanie objektov výučby (automatizovaná tvorba kurzu).

Odstraňovanie objektov výučby ani práca s ich obsahom nie je možná, nakoľko to nie je cieľom nástroja CourseDesigner.

Manuálna tvorba kurzu

Akonáhle začneme kurz vytvárať manuálne, nie je možné automatizovať žiaden proces tvorby kurzu, aby nebola narušená integrita kurzu. Naopak to neplatí.

Pri manuálnej tvorbe kurzu vytvárame pre načítané výučbové objekty koncepty a prepájame ich reláciou typu *belongsTo*. Po vytvorení konceptov ich prepojíme navzájom pomocou relácií typu *contains*, *isRelatedTo*, *prerequisity*.

Práca s konceptmi

Tvorba konceptov je možná pomocou:

- hlavného menu: *Course -> Add concept*
- lišty nástrojov kliknutím na ikonu 
- pomocou vyskakovacieho menu vo vizualizačnej časti nástroja a výberu príslušnej položky v menu (vyskakovacie menu vyvoláme na „prázdnom“ mieste

Odstraňovanie a úpravy vlastností konceptov je možné pomocou:

- hlavného menu: *Concept -> Remove*, *Concept -> Edit properties*
- lišty nástrojov kliknutím na ikonu  , resp. 
  (koncept musí byť označený v paneli konceptov alebo v grafe)

- pomocou vyskakovacieho menu vo vizualizačnej časti nástroja a výberu príslušnej položky v menu (vyskakovacie menu vyvoláme nad označeným konceptom)

Práca s reláciami

Tvorba relácií je možná pomocou:

- hlavného menu: *Course -> Add relation*
- lišty nástrojov kliknutím na ikonu
- pomocou vyskakovacieho menu vo vizualizačnej časti nástroja a výberu príslušnej položky v menu (vyskakovacie menu vyvoláme na „prázdnom“ mieste)

Odstraňovanie a úpravy vlastností relácií je možné pomocou:

- hlavného menu: *Relation -> Remove*, *Relation -> Edit properties*
- lišty nástrojov kliknutím na ikonu , resp. 
  (relácia musí byť označená v paneli konceptov alebo v grafe)
- pomocou vyskakovacieho menu vo vizualizačnej časti nástroja a výberu príslušnej položky v menu (vyskakovacie menu vyvoláme nad označenou reláciou)

Automatizovaná tvorba kurzu

Ak chceme kurz vytvárať automatizovane, na vykonanie určitej rutinnej práce máme k dispozícii funkciu automatizovaného generovania štruktúry kurzu, ktorá je spúšťaná pomocou:

- hlavného menu: Course -> Generate course structure
- lišty nástrojov kliknutím na ikonu 

Táto činnosť vyvolá vždy jeden z krokov metódy automatizovaného získavania metadát o kurze – podľa stavu, k akom sa kurz nachádza:

1. ak sú načítané objekty výučby a nebol vytvorený žiaden koncept, vykoná sa predspracovanie objektov výučby (tento krok je možné spustiť aj samostatne pomocou klávesovej skratky Ctrl+F).
2. ak boli objekty výučby predspracované a nebol manuálne pridaný žiaden koncept, vykoná sa extrahovanie konceptov (tento krok je možné spustiť aj samostatne pomocou klávesovej skratky Ctrl+G).
3. ak boli extrahované koncepty a nebola zatiaľ manuálne vytvorená žiadna relácia, vykoná sa krok generovania vzťahov (tento krok je možné spustiť aj samostatne pomocou klávesovej skratky Ctrl+H).

Uloženie štruktúry kurzu

Ak sme skončili prácu na vytváraní doménového modelu kurzu, je potrebné kurz uložiť.

Uloženie kurzu je možné vykonať pomocou:

- hlavného menu: *File -> Save course*
- lišty nástrojov kliknutím na ikonu 

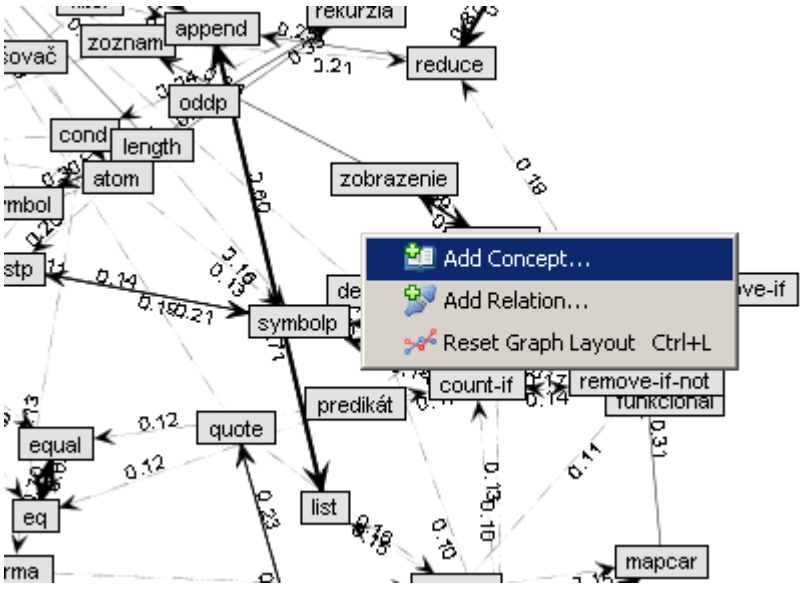
Kurz bude uložený do adresára, z ktorého boli načítané objekty výučby. Vytvoria sa súbory podľa špecifikácie doménového modelu.

B.6 Vizualizácia kurzu

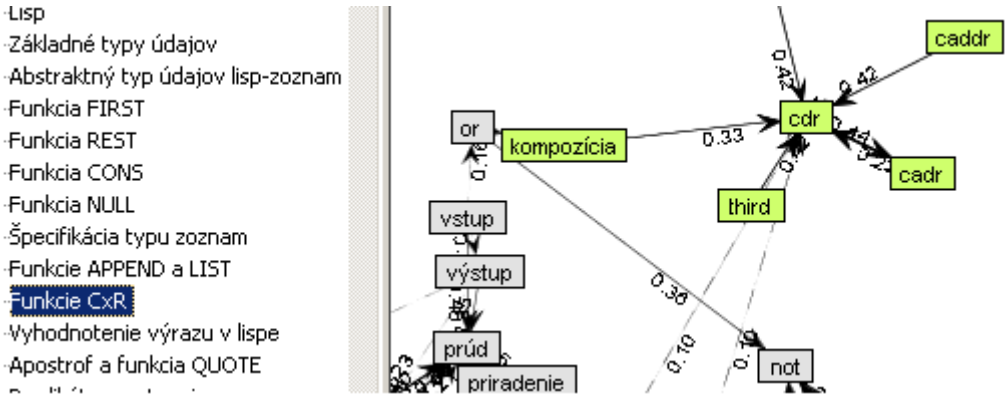
Úlohou vizualizačného komponentu nástroja CourseDesigner je zjednodušiť a sprehľadniť prácu používateľa so zložitým a rozsiahlym doménovým modelom.

Kurz je vizualizovaný ako graf s konceptmi ako uzlami a reláciami ako hranami. Hrany sú ohodnotené podľa váhy relácie. Interaktivita s používateľom prebieha pomocou nasledujúcich funkcií:

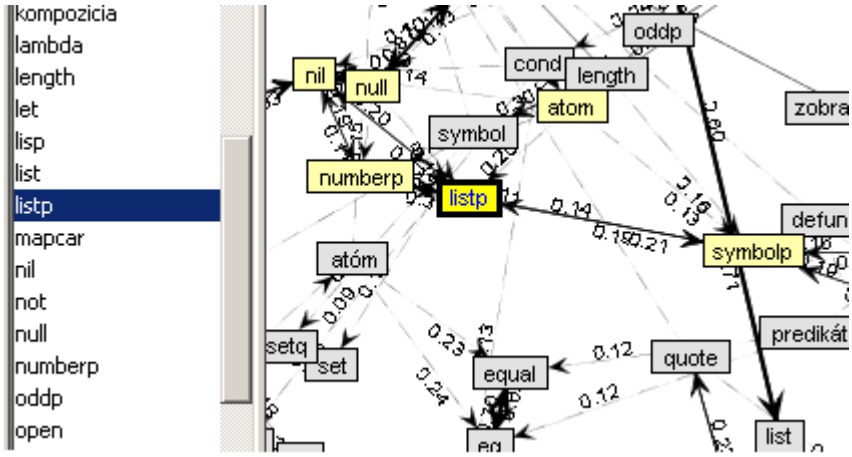
- presúvanie konceptov pomocou myši;
- interaktívne pridávanie konceptov na vyznačené miesto v grafe (obr.);
- odstraňovanie a úprava vlastností konceptov a relácií označených v grafe;
- vizualizácia konceptov prislúchajúcim označeným objektom výučby (obr. B-10);
- vizualizácia najbližších susedov konceptov (obr. B-11);
- zväčšenie / zmenšenie grafu;
- rekonštrukcia rozloženia konceptov;



Obrázok B-9: Pridanie konceptu na vyznačené miesto



Obrázok B-10: Vizualizácia konceptu konceptov prislúchajúcich objektu výučby Funkcie CxR



Obrázok B-11: Vizualizácia konceptu listp a jeho susedov

Príloha C: Príspevok prijatý na konferenciu IIT.SRC 2008

V tejto časti sa nachádza príspevok prijatý na študentskú konferenciu IIT.SRC 2008. Príspevok získal cenu „Best paper award“ v kategórii inžinierskych projektov.

Predkladaný príspevok poslúži ako podklad pre vypracovanie príspevku, ktorý bude mať ambíciu byť publikovaný na niektorej z medzinárodných konferencií.

Príloha D: Obsah dátového nosiča

Prílohou tejto práce ja dátový nosič. Štruktúra jeho obsahu je nasledovná:

```
\bibliography   ......... použitá literatúra v elektronickej podobe

\doc            ......... dokumentácia k diplomovej práci
                          v elektronickej podobe

\cd             ......... nainštalovaný spustiteľný nástroj
                          CourseDesigner

\cd-doc         ......... technická dokumentácia k nástroju
                          CourseDesigner

\cd-src         ......... zdrojový kód nástroja
   \project     ......... zdrojový kód nástroja integrovaný
                          v podobe projektu pre NetBeans IDE

\course-spec    ......... špecifikácia kurzu projektu PeWePro

\install        ......... inštalačné súbory
   \cd          ......... nástroj CourseDesigner
   \java        ......... prostredie Java pre spustenie nástroja
   \misc        ......... ostatné nástroje
   \netbeans    ......... vývojové prostredie pre uľahčenie
                          prezerania zdrojového kódu

\testing        ......... testovacie dáta
   \flp-course  ......... kurz funkcionálneho programovania
   \flp-ref     ......... referenčná štruktúra kurzu
   \results     ......... výsledky experimentov

\readme.txt     ......... tento súbor
```