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

PROJEKTOVÁ DOKUMENTÁCIA

JOBS
PORTÁL PRACOVNÝCH PRÍLEŽITOSTÍ

Tím č. 8: Hachiban
Vedúci tímu: tvarozek@gmx.net
Mailový alias: tp@atrip.sk
Predmet: Tvorba softvérového systému v tíme
Pedag. vedúci: Ing. Roman Filkorn
Ak. rok: 2005/2006

Členovia tímu:
Bc. Michal Barla
Bc. Peter Bartalos
Bc. Ján Porubský
Bc. Peter Sivák
Bc. Kristián Szobi
Bc. Michal Tvarožek

Zadanie – Portál pracovných príležitostí

Vedúci tímu: Ing. Roman Filkorn

Portál pracovných príležitostí je moderným nástrojom pre úspešnú výmenu informácií dvoch zúčastnených strán: zamestnávateľov a uchádzačov o zamestnanie. Portál na jednej strane poskytuje zamestnávateľom možnosť zverejniť ponuku pracovnej príležitosti širokej cieľovej komunite, na strane druhej umožňuje vyhľadávať medzi pracovnými ponukami a vyberať si na základe vlastného profilu a záujmov uchádzača o prácu. Portál pomáha spravované informácie kategorizovať, sledovať aktuálnosť, poskytovať previazanie s podobnými portálmi v iných (geografických) oblastiach, uľahčiť opakované činnosti najmä uchádzačom o prácu.

Portál bude integrovanou súčasťou väčšieho projektu. Bude vhodné, keď bude portálové riešenie vychádzať z existujúcich analýz, bude v čo najväčšej miere využívať existujúce modely domény a aj celkový zámer integrálneho projektu - znovupoužiteľnosť riešenia pre iné informačné domény. Takéto ciele kladú vyššie nároky na úroveň návrhu architektúry a celkovej implementácie riešenia - s ohľadom na konfigurovateľnosť a prispôsobiteľnosť vytvoreného produktu.

Vašou úlohou bude na základe zozbierania požiadaviek a primeraného naštudovania modelov a technológii integrálneho projektu analyzovať, navrhnúť, implementovať a overiť portál pracovných príležitostí tak, aby pokrýval nie len základné funkcionálne požiadavky, ale aby bolo riešenie v čo najväčšej miere znovupoužiteľné pri implementácii iných podobných portálových riešení.

Predslov

Predkladaný dokument obsahuje projektovú dokumentáciu k softvérovému systému vytvorenému v rámci predmetu Tvorba softvérového systému v tíme v akademickom roku 2005/2006. Zadaním bolo vypracovať Portál pracovných príležitostí ako súčasť väčšieho systému. Riešiteľom projektu bol tím č. 8 – Hachiban, pričom na tvorbe dokumentácie sa podieľali všetci členovia tímu.

Dokument pozostáva z dvoch hlavných častí. Časť I obsahuje dokumentáciu k inžinierskemu dielu – vytvorenému softvérovému systému. Dokumentácia súvisiaca s riadením projektu je obsiahnutá v časti II.

ČASŤ I
–
SOFTVÉROVÝ SYSTÉM

Obsah

0. Úvod

Predložená projektová dokumentácia, opisuje riešenie zadania Portál pracovných príležitostí. V prvej fáze riešenia sme vykonali analýzu problémovej oblasti a preštudovali sme existujúce materiály k štátnemu program NAZOU. Pokračovali sme analýzou nástrojov, technológií a spôsobov tvorby webových portálov. Následne sme vykonali vyhodnotenie fázy analýzy a plynule prešli k navrhovaniu riešenia, ktoré vyústilo do hrubého návrhu portálu pracovných príležitostí.

V druhej fáze riešenia sme rozpracovali návrh riešenia a vykonali jeho implementáciu a testovanie. Priebežne sme o vytváranom riešení napísali 2 články – na konferenciu IIT.SRC a na konferenciu ISD. Po odovzdaní prvej verzie produktu zákazníkovi sme vyhodnotili a vykonali pripomienky zákazníka v podobe dodatočných zmien a odovzdali hotový produkt.

Opis riešeného problému sa nachádza v kapitole 1. Táto obsahuje stručný prehľad problémovej oblasti a opisuje východiská pri tvorbe softvérového systému. Vysvetľuje jeho základné črty, hlavné ciele, spolu s najdôležitejšími vlastnosťami a najčastejšími prípadmi použitia.

Kapitola 2 sa sústredí na analýzu problémovej oblasti a identifikuje entity, ktoré ju tvoria. Venuje sa tiež prístupom k prispôsobovaniu sa potrebám používateľov v kontexte domény pracovných príležitostí.

Analýza nástrojov potrebných k riešeniu zadania sa nachádza v kapitole 3. Dôležitá je najmä analýza ontologických úložísk, ďalej analýza aplikačných a prezentačných rámcov a tiež analýza portálových riešení a nástrojov na návrh portálov.

Špecifikácia riešenia je opísaná v kapitole 4, ktorá obsahuje požiadavky na riešenie a existujúce ohraničenia. Opisuje tiež jednotlivé prípady použitia pomocou tabuliek a diagramov.

Hrubému návrhu riešenia je venovaná kapitola 5. Obsahuje návrh architektúry systému a návrh webu portálu. Kapitola ďalej obsahuje zhodnotenie analýzy nástrojov z kapitoly 4 a krátky opis niektorých zásad práce.

Proces tvorby prototypu spolu s uvedením problémov a ich riešení ako aj krátkeho zhodnotenia procesu prototypovania obsahuje kapitola 6.

Opis riešenia– zmeny návrhu a samotnú implementáciu obsahuje kapitola 7. Kapitola 8 obsahuje stručné zhrnutie práce vykonanej na projekte po dvoch semestroch. Na konci dokumentu sa nachádza zoznam použitej literatúry a prílohy. V prílohách uvádzame vybrané analýzy, technickú dokumentáciu k riešeniu, články na konferencie a používateľskú dokumentáciu.

0.1. Skratky

Skratka	Plný význam
API	Application Programming Interface
CRUD	Create Retrieve Update Delete
CSS	Cascading Style Sheets
EJB	Enterprise JavaBeans
FOSS	Free and Open Source Software
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	HyperText Markup Language
JDBC	Java DataBase Connectivity
JMS	Java Message Service
JSP	Java Server Pages
LDAP	Lightweight Directory Access Protocol
LGPL	Lesser General Public License
MVC	Model-View-Controller
NAZOU	NÁstroje pre Získavanie, Organizovanie a Udržovanie znalostí v prostredí heterogénnych informačných zdrojov
OPS	Orbeon Presentation Server
OWL	Web Ontology Language
SOAP	Simple Object Access Protocol (vo verzii 1.2 už len SOAP)
SQL	Structured Query Language
RDF	Resource Description Framework
RDFS	RDF Schema
RQL	RDF Query Language
RMI	Remote Method Invocation
RSS	Rich Site Summary, RDF Site Summary, Really Simple Syndication
SAX	Simple Api for XML
SeRQL	Sesame RQL
SMTP	Simple Mail Transfer Protocol
PFC	Page Flow Controller
W3C	World Wide Web Consortium
WebDAV	Web-based Distributed Authoring and Versioning
WML	Wireless Markup Language
WYSIWYG	What You See Is What You Get
XHTML	Extensible HTML
XML	EXtensible Markup Language
XPL	XML Pipeline Language
XSL	EXtensible Stylesheet Language
XSLT	XSL Transformations

1. Opis riešeného problému

V tejto časti stručne opíšeme povahu riešeného problému a všeobecný kontext vytváraného softvérového systému. Stručne charakterizujeme doménu pracovných príležitostí, v ktorej bude vytvorený systém pôsobiť, ďalej uvedieme hlavné ciele a vlastnosti navrhovaného riešenia. Kapitolu zakončíme prehľadom základnej funkcionality systému.

1.1. Prehľad problémovej oblasti

Trh práce možno charakterizovať ako miesto, kde sa stretáva dopyt po pracovných príležitostiach od rôznych firiem a organizácií s ponukou od ľudí hľadajúcich prácu. Problematika trhu práce a sprostredkovania pracovných príležitostí je v súčasnom svete veľmi aktuálna. Je to spôsobené najmä rastúcim dopytom po kvalifikovanej pracovnej sile ako aj narastajúcimi požiadavkami zamestnancov na svojich budúcich zamestnávateľov.

Do procesu sprostredkovania práce už dávnejšie vstúpili aj moderné technológie, ktoré vyústili do vzniku webových portálov pracovných príležitostí, umožňujúcich zamestnávateľom zverejňovať informácie o voľných pracovných miestach. Potenciálni zamestnanci majú možnosť v takto získaných pracovných ponukách vyhľadávať, alebo zverejniť svoj životopis, ktorý si môžu následne prezerať zamestnávatelia.

Celkový počet účastníkov trhu práce – firiem a potenciálnych zamestnancov je v súčasnosti už taký veľký, že hľadanie pracovných príležitostí a zamestnancov je časovo veľmi náročné. Rastúci počet profesií a požiadaviek na odborné znalosti zamestnancov túto situáciu len zhoršuje. Podobne možnosti cestovania po svete sa rozširujú a sťahovanie sa za prácou a práca v zahraničí sa stávajú bežnou praxou.

Medzi problémy, ktoré sa v súčasnosti prejavujú, patrí najmä skutočnosť, že zamestnávatelia musia informácie o voľných pracovných miestach zverejňovať ručne, často na viacerých miestach. Mnoho zamestnávateľov zverejňuje ponuky práce len na vlastných webových stránkach. Dôsledkom je vysoká časová náročnosť zverejňovania informácií o ponukách, spôsobená najmä nízkou mierou automatizácie pri zbieraní, zadávaní a vyhľadávaní ponúk. Navyše len malá časť voľných pracovných príležitostí je zverejnená na existujúcich portáloch.

Existencie mnohých portálov, z ktorých každý obsahuje len zlomok z celkovej ponuky voľných pracovných miest, kladie vysoké požiadavky na ľudí hľadajúcich zamestnanie. Uchádzači o zamestnanie musia hľadať prácu na viacerých portáloch s rozličnými rozhraniami, pričom aj samotné vyhľadávanie pracovných ponúk na jednom portáli je často obtiažne a málo intuitívne. Rozsah prispôsobovania sa portálov potrebám jednotlivých používateľov je len minimálny.

1.2. Ciele a vlastnosti produktu

Súčasné portálové riešenia v doméne pracovných príležitostí majú celý rad nedostatkov (pozri 1.1). Cieľom vytváraného portálu pracovných príležitostí je riešiť uvedené problémy v kontexte väčšieho systému, ktorý rieši automatizované získavanie a organizovanie znalostí o ponukách pracovných príležitostí. Účelom portálu je ponúknuť jednoduché a najmä efektívne rozhranie ku službám systému tak, aby používatelia mohli rýchlo a pohodlne vyhľadávať v údajoch o pracovných ponukách.

Zámerom štátneho programu je vytvoriť všeobecné nástroje na prácu so znalosťami, pričom ich charakter sa navyše môže (a bude) časom meniť, čo sa prejaví najmä v zmenách doménovej ontológie. V tomto duchu je cieľom portálu využiteľnosť aj v iných problémových oblastiach, ktorú chceme dosiahnuť modularitou a dostatočne všeobecnými a

modifikovateľnými komponentmi. Zároveň chceme navrhnúť prezentačnú časť tak, aby v maximálnej možnej miere využívala údaje z ontológie a vedela sa prispôsobiť aspoň menším zmenám aj bez priameho zásahu do zdrojových kódov.

Vzhľadom na rozsah portálu sú hlavné ciele projektu nasledovné:

- Sprostredkovanie rozhrania medzi používateľom a nástrojmi programu NAZOU
- Zníženie množstva manuálnej práce potrebnej na zadávanie ponúk
- Skrátenie času potrebného na nájdenie vhodnej ponuky, resp. vybavenie požiadavky
- Vyhľadanie optimálnych ponúk pre používateľa
- Zameranie sa na používateľa, prispôsobenie sa jeho potrebám a preferenciám
- Sprístupnenie väčšej časti voľných pracovných miest používateľom

Portál pracovných príležitostí bude s používateľmi komunikovať prostredníctvom webového prehliadača. Je určený pre uchádzačov o prácu a pre organizácie hľadajúce zamestnancov. Práve druhej skupine používateľov by sme chceli dať možnosť využiť aj alternatívne rozhranie portálu v podobe webových služieb.

Ďalšie vlastnosti softvérového systému:

- Návrh a implementácia portálu ako sady nástrojov
- Integrovateľnosť do systému nástrojov vytvorených v štátnom programe NAZOU
- Jednoduchý a efektívny prístup k funkciám, viaceré spôsoby prezentácie informácií
- Využitie prostriedkov webu so sémantikou, reprezentácia dát pomocou ontológie
- Spoľahlivosť, dostupnosť, bezpečnosť

1.3. Prehľad produktu

Očakávanú funkcionalitu výsledného portálu ilustrujeme na príklade niekoľkých scenárov použitia. Pretože cieľom projektu je nadviazať na existujúcu prácu vykonanú v štátnom programe NAZOU, neuvádzame na tomto mieste všetky možné scenáre použitia portálu. Detailný opis jednotlivých scenárov sa nachádza v [14].

Scenár #1: Vyhľadanie ponuky na pracovné miesto

Situácia: Používateľ hľadá vhodnú ponuku na voľné pracovné miesto.

Opis: Používateľ sa prihlási do systému. V časti pre vyhľadanie pracovnej ponuky zvolí spôsob vyhľadávania a zadá požadované kritériá na pracovnú ponuku. Následne si prezerá výsledky vyhľadávania v stručnom prehľade. Pre vybrané ponuky si prezrie podrobné údaje o ponuke. Po nájdení vhodnej ponuky si vytlačí podrobné údaje o ponuke.

Scenár #2: Vloženie ponuky na voľné pracovné miesto

Situácia: Používateľ vkladá ponuku na voľné pracovné miesto.

Opis: Používateľ sa prihlási do systému. V časti vloženie novej ponuky vyplní údaje o ponuke do formuláru. Po vložení potrebných údajov potvrdí vloženie ponuky do systému.

Scenár #3: Modifikácia existujúcej ponuky

Situácia: Používateľ modifikuje existujúcu ponuku, ktorú už v minulosti zadal.

Opis: Používateľ sa prihlási do systému. V časti vlastných ponúk si zo zoznamu svojich existujúcich ponúk vyberie tú, ktorú chce modifikovať. Následne vykoná príslušné zmeny v ponuke a potvrdí vloženie zmien do systému.

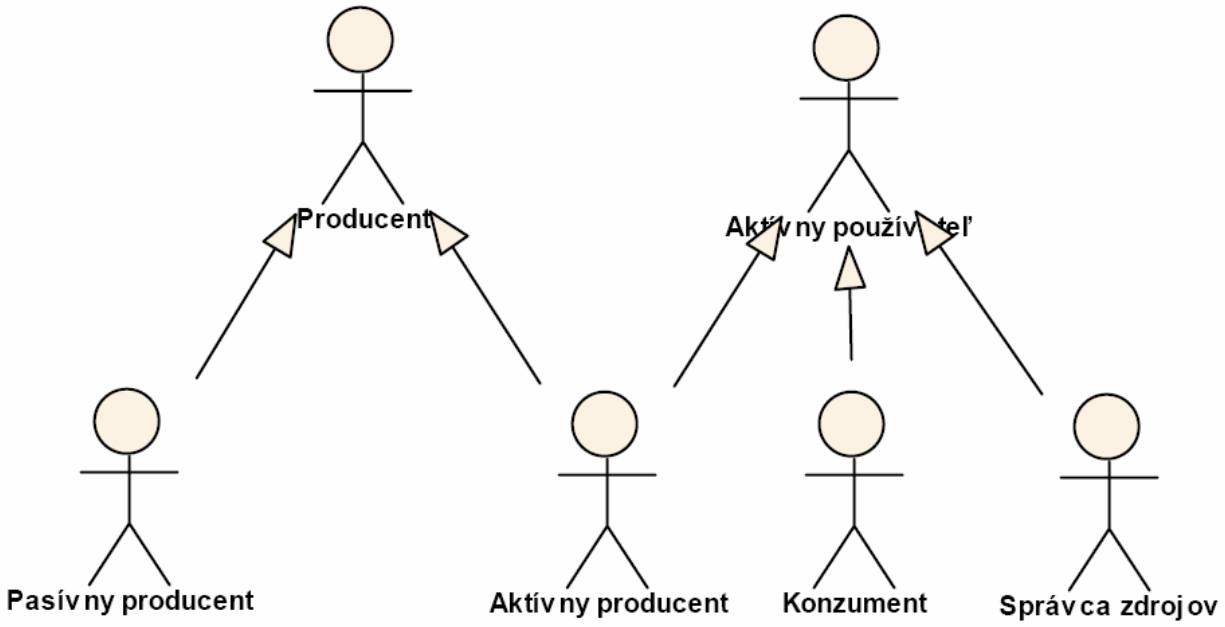
2. Analýza problémovej oblasti

V tejto kapitole uvádzame analýzu problémovej oblasti, z ktorej sme vychádzali pri špecifikácii a ďalšom riešení projektu. Kapitola obsahuje analýzu domény pracovných ponúk so zovšeobecnením na doménu všeobecných ponúk. Ďalej nasleduje analýza existujúcich portálových riešení a analýza prispôsobovania sa konkrétnemu používateľovi, resp. prispôsobovania sa rôznym skupinám používateľov.

2.1. Analýza domény pracovných ponúk

Pri analýze domény pracovných ponúk sme vychádzali z analýzy vykonanej v rámci projektu NAZOU [14]. Identifikovali sme dve role účastníkov procesu získavania a poskytovania ponúk – *producent* a *konzument*. Komunikačný prostriedok medzi producentom a konzumentom je *ponuka*.

Producent vytvára ponuky a poskytuje ich konzumentom. Naopak konzument vyhľadáva v ponukách, ktoré mu poskytli producenti. Účastník procesu môže byť zároveň aj producentom aj konzumentom. Uchádzač o prácu – konzument, môže vyhľadávať v ponukách zamestnávateľov – producentov alebo naopak zamestnávateľ – konzument, môže vyhľadávať v životopisoch uchádzačov o prácu, kde životopis predstavuje ponuku potenciálneho zamestnanca – producenta.



Obr. 1 Účastníci procesu získavania a poskytovania ponúk

Producentov a konzumentov možno ďalej špecializovať (pozri Obr. 1). Producent ponúk, ktorý aktívne zadáva ponuky do systému je *aktívny producent*. Opakom je *pasívny producent*, ktorý so systémom nepracuje (teoreticky nemusí vedieť ani o jeho existencii) a ponuky v informačnom priestore uverejňuje iným spôsobom, napr. na vlastnej webovej stránke. Na získanie údajov o ponukách pasívnych producentov predpokladáme využitie nástrojov vyvíjaných v rámci štátneho programu NAZOU.

Nepredpokladáme žiadne prípady použitia určené pre pasívneho producenta, lebo z jeho povahy vyplýva, že sa priamo nezúčastňuje na procese poskytovania ponúk. Rola konzumenta predpokladá aktívne využitie systému používateľom, ktorý v systéme má vytvorený

používateľský účet. Z tohto dôvodu nemá význam hovoriť o pasívnom či aktívnom konzumentovi.

Okrem dvoch základných rolí, sme v procese poskytovania ponúk identifikovali ešte rolu *správca zdrojov*, ktorá nepriamo vplýva na proces poskytovania ponúk. Úlohou správcu zdrojov je udržiavať databázu zdrojov ponúk.

Zadanie projektu predpokladá využitie portálu v doméne pracovných ponúk, problematiku je možné zovšeobecniť na doménu všeobecných ponúk a nielen ponúk práce. Doména všeobecných ponúk v sebe potom obsahuje napr. aj ponuky v cestovnom ruchu alebo ponuky nehnuteľností.

2.1.1. Ontológia pracovných ponúk

Údaje o pracovných ponukách je potrebné uložiť vo vhodnom tvare do databázy. Pretože je predpoklad, že sa údaje o pracovných ponukách budú meniť, je potrebné aby bol formát ich uloženia dostatočne flexibilný a odolný voči zmenám. Využitie systému aj pre iné domény v rámci domény všeobecných ponúk predpokladá navyše zmenu obsahu, štruktúry a sémantiky spracúvaných údajov. Z týchto dôvodov je vhodné uložiť údaje vo forme ontológie.

Pri analýze sme využili existujúcu ontológiu pracovných ponúk, ktorá bola vytvorená v rámci program NAZOU. V analýze vykonanej v rámci programu NAZOU boli identifikované jednotlivé koncepty pracovnej ponuky a vzťahy medzi nimi. Výsledná ontológia pracovných ponúk predstavuje explicitnú formálnu špecifikáciu konceptualizácie pracovnej ponuky. Práve využitie rovnakej definície pracovnej ponuky v našom projekte ako v štátnom programe NAZOU je predpokladom budúcej integrovateľnosti oboch riešení.

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

- Ontológia *region* – definujúca koncepty regiónov, krajín, jazykov a mien, ktoré sa používajú v daných regiónoch
- Ontológia *classification* – definujúca hierarchie pre priemyselné odvetvia, profesie, úrovne vzdelania, kvalifikácie a rôzne usporiadania
- Ontológia *offer* – definujúca všeobecnú ponuku

2.2. Analýza existujúcich portálov pracovných príležitostí

Analyzovali sme viaceré existujúce riešenia portálov pracovných príležitostí. Identifikovali sme ich hlavné vlastnosti a najmä nedostatky, ktoré chceme naším systémom riešiť.

2.2.1. Portál Profesia.sk

Portál je určený uchádzačom o prácu aj spoločnostiam hľadajúcim zamestnancov. Používatelia majú možnosť vložiť svoj životopis, resp. ponuku práce, v ktorých následne druhí môžu vyhľadávať. Je možná registrácia používateľov, avšak možnosti personalizácie sú obmedzené na použitie agenta pre ponuky, zadanie životopisu a históriu prezretých ponúk.

Možnosti vyhľadávania v ponukách predpokladajú vyhľadávanie štruktúrovaných, v ručne vložených údajoch o ponukách. Používateľ má možnosť vybrať si jednoduché alebo rozšírené vyhľadávanie v ponukách podľa viacerých kritérií ako sú napr. požadované

vzdelanie, lokalita, kategória, druh pracovného pomeru. V prípade rozšíreného vyhľadávania je nastavenie kritérií málo intuitívne a neprehľadné. Samotné vyhľadávanie je len filtrom nad ponukami v databáze, pričom výsledky vyhľadávania nie sú usporiadané podľa relevantnosti a nie je možné v nich ďalej vyhľadávať.

2.2.2. Portál CareerBuilder.com

Portál je opäť určený uchádzačom o prácu aj spoločnostiam hľadajúcim zamestnancov. Oproti portálu Profesia.sk má prepracovanejšie rozhranie a väčšiu funkcionalitu. Podporuje registráciu používateľov, ktorí následne majú možnosť vložiť svoj životopis, resp. ponuku práce. Možnosti personalizácie sú obmedzené na použitie agenta pre ponuky, zadanie životopisu, zadanie motivačných listov, históriu prezretých ponúk a históriu vyhľadávaní (zvolených vyhľadávacích kritérií).

Z hľadiska možností vyhľadávania poskytuje portál jednoduché aj rozšírené vyhľadávanie, podľa kľúčových slov a kategórií (profesia, odvetvie, lokalita, spoločnosť). Rozšírené vyhľadávanie umožňuje komplexné vyhľadávanie viacerých kategórií súčasne, ďalej definovanie lokalít práce a vzdialeností od nich, zadanie kľúčových slov, zadanie požadovaného vzdelania, typu pracovnej pozície a očakávaného platu.

Veľmi užitočnou je možnosť dodatočného prezerania výsledkov vyhľadávania, ktoré sú usporiadané podľa dátumu zadania ponuky. Portál využíva prehliadanie pomocou fazetového prehliadača, ktorý ďalej umožňuje ohraničiť výsledky podľa kategórií, platu, lokality a spoločnosti. Umožňuje tiež z výsledkov odstrániť ponuky obsahujúce vybrané kľúčové slová.

Pridanú hodnotu portálu tvoria tiaž informácie o vzdelaní, poradenstve a trhoch práce. Veľkým prínosom je tiež počet dostupných pracovných príležitostí, ktorý prevyšuje stotisíce pracovných miest. Pre porovnanie, Profesia.sk obsahuje približne päťtisíc pracovných príležitostí.

2.3. Analýza prispôsobovania

Portál pracovných príležitostí bude používateľom poskytovať prístup k rozsiahlemu informačnému priestoru. Veľké množstvo dostupných informácií a rozsiahlosť informačného priestoru často vedú k neschopnosti používateľov efektívne sa orientovať v danom priestore, čo často vedie k efektívnemu „strateniu sa“ používateľov.

Na portáli pracovných príležitostí sa bude stretávať veľké množstvo používateľov s rozličnými skúsenosťami, schopnosťami, cieľmi a preferenciami. Vytvoriť jedno rozhranie, ktoré bude vyhovovať všetkým nie je možné. Na druhú stranu uspokojiť sa s jedným rozhraním pre všetkých nie je dostatočne efektívne a v konečnom dôsledku pravdepodobne povedie k nespokojnosti používateľov.

Uvedené problémy možno aspoň čiastočne riešiť pomocou metód prispôsobovania sa portálu jednotlivým používateľom. Vo všeobecnosti možno prispôsobovať:

- Obsahu portálu
- Prezentáciu, resp. vzhľad portálu
- Navigáciu po portáli

Podľa spôsobov prispôsobovania sa systému rozlišujeme systémy:

- Adaptabilné, prispôsobujúce sa na základe informácií priamo zadaných používateľom

- Adaptívne, prispôsobujúce sa na základe informácií, zistených o používateľovi automaticky alebo nepriamo

V praxi sa najvhodnejšou javí byť kombinácia obidvoch prístupov, pretože niektoré charakteristiky používateľa sa nedajú zistiť automaticky. Ideálny stav je vždy taký, keď systém dokáže zistiť väčšinu (všetky) potrebných informácií o používateľovi samostatne, len s minimálnymi požiadavkami na používateľa.

Systém sa prispôsobuje používateľovi na základe informácií, reprezentovaných modelom používateľa. Existujú dva prístupy k realizácii modelov používateľa – pomocou stereotypov alebo prekryvným modelom.

Prekryvný model umožňuje reprezentáciu charakteristík každého používateľa samostatne, pričom predpokladá kópiu doménového modelu pre každého používateľa. Otvoreným problémom zostáva počiatočná inicializácia modelu, keďže na začiatku práce používateľa so systémom o ňom systém ešte nemá uložené žiadne informácie.

Stereotypy predstavujú jednoduchšiu metódu, ktorá neumožňuje personalizáciu – prispôsobovanie sa jednotlivým používateľom. Používatelia sú rozdelení do skupín a systém prispôsobuje obsah, jeho prezentáciu alebo navigáciu jednotlivým skupinám. Stereotypy sú výhodné najmä vtedy, ak nemáme dostatok informácií o jednotlivých používateľoch, teda na začiatku práce používateľa s adaptívnym systémom.

V praxi sa osvedčil kombinovaný prístup, v ktorom sa najprv inicializuje prekryvný model používateľa pomocou niektorého stereotypu a následne sa použije prekryvný model, ktorý zaznamenáva charakteristiky používateľa pri práci so systémom.

Charakteristiky používateľa sa delia na:

- Doménovo nezávislé – nesúvisia so zvolenou aplikačnou doménou a je možné ich využiť v ľubovoľnom adaptívnom systéme
- Doménovo závislé – úzko súvisia so zvolenou aplikačnou doménou, teoreticky využiteľné v inom systéme, ktorý pracuje nad rovnakou doménou
- Systémovo závislé – úzko súvisia s konkrétnym systémom, prakticky nevyužiteľné v inom systéme

Aby sa systém mohol prispôsobovať používateľovi, je potrebné identifikovať a získať relevantné informácie o používateľovi. Okrem doménovo nezávislých informácií sme identifikovali nasledovné zdroje informácii o používateľovi:

- Záznam aktivít používateľa na portáli
- Dolovanie v textoch (napr. životopis)
- Dotazníková metóda – odpovede na špeciálne pripravené testovacie otázky
- Informácie získané z adaptability portálu, priamo zadané používateľom

2.3.1. Ontológia používateľa

Z kapitoly 2.2 vyplýva potreba existencie modelu používateľa pre potreby prispôsobovania. Obsah a štruktúra modelu používateľa sa môže, resp. bude s časom meniť, čo vedie k potrebe jeho reprezentácie vo flexibilnej podobe. Vhodným formalizmom na zápis modelu používateľa je zápis pomocou ontológie. Z rozdelenia charakteristík používateľa podľa kapitoly 2.2 vyplýva rozdelenie ontológie používateľa na doménovo závislú a doménovo nezávislú časť.

Doménovo nezávislá časť ontológie modeluje používateľa definovaním konceptov, ktoré má zmysel skúmať v ľubovoľnej aplikačnej doméne. Sú to napríklad vek, pohlavie, vzdelanie a všeobecné preferencie používateľa.

Doménovo závislá časť ontológie modeluje preferencie používateľa vzhľadom na jednotlivé entity a atribúty doménovej ontológie pracovných ponúk. Tieto preferencie vždy obsahujú určitý stupeň neurčitosti (fuzzy prístup) a preto obsahujú vždy dve hodnoty, ktoré definujú intervaly úplného zamietnutia a úplného akceptovania atribútu.

Aby bolo možné portál pracovných príležitostí integrovať so systémom vytváraným v rámci štátneho programu NAZOU, rozhodli sme sa prevziať model používateľa z tohto systému.

2.3.2. Prispôsobovanie webu pre používateľov so špecifickými potrebami

Portál pracovných príležitostí je určený pre široké spektrum používateľov. Problematika sprístupnenia webu ľuďom so špecifickými potrebami sa v poslednom čase dostáva stále viac do popredia. Nezanedbateľnú časť potenciálnych používateľov portálu tvoria používatelia s nejakým postihnutím, ktoré znižuje ich schopnosť využívať funkcie portálu. Do tejto skupiny patria najmä ľudia s ťažkým postihnutím zraku, ktorí používajú špeciálne technológie na sprístupnenie obsahu webu. Najčastejšie sa používajú hlasové alebo hmatové systémy, pomocou ktorých sa text z webu prečíta alebo zobrazí v braillovom písme. Slabozrakí ľudia používajú tzv. softwarové lupy, ktoré im zväčšia časť obrazovky. Touto funkciou disponujú už aj niektoré webové prehliadače (Opera, Mozilla).

Pri tvorbe stránok pre zrakovo postihnutých ľudí je potrebné dodržiavať určité pravidlá, aby pomocné programy mohli transformovať obsah stránky do im zrozumiteľnej podoby. Pri vytváraní stránok pre zrakovo postihnutých používateľov je potrebné zohľadniť nasledovné skutočnosti :

- používatelia sú schopní získať iba informácie v textovej podobe
- nevidiaci dokáže vnímať informácie na stránke iba v lineárnej postupnosti, chýba mu celkový pohľad na stránku
- slabozraký pri zväčšovaní obrazovky môže vidieť v danej chvíli iba jej časť
- nevidiaci používatelia ovládajú programy primárne pomocou klávesnice a klávesových skratiek, alebo pomocou rozpoznávania reči

Dodržanie uvedených pravidiel zlepší prístupnosť stránok nielen pre používateľov so zrakovým postihnutím ale napríklad aj pre používateľov s postihnutím horných končatín, ktorí nemôžu počítač ovládať myšou resp. pre používateľov s poruchami sústredenia (pomalé reakcie na vizuálne podnety).

Z vykonanej analýzy prispôsobovania stránok pre ľudí so špecifickými potrebami vyplýva, že portál pracovných príležitostí by mal dodržiavať uvedené pravidlá pre konečnú prezentáciu informácií na webe, aby umožnil prácu s portálom aj používateľom s obmedzeniami.

3. Analýza riešenia

V tejto časti sa dokument venuje analýze dostupných technológií a nástrojov, určených na tvorbu webových portálov. Cieľom kapitoly je poskytnúť pohľad na dostupné technológie a nástroje, aby sme boli schopní z nich vybrať tie najvhodnejšie, pomocou ktorých následne budeme vytvárať portál pracovných príležitostí.

3.1. Analýza portálových riešení

Portál je webová aplikácia, ktorá poskytuje jednotné rozhranie k rôznorodým zdrojom na Internete. Prístup k zdrojom cez portál je transparentný, t.j. používateľ môže k portálu pristupovať z rôznych zariadení a z rôznych miest na svete. Základným stavebným kameňom portálov na platforme J2EE sú *portlety*. Portlety sú komponenty, ktoré zapúzdrujú časť funkcionality portálu (napr. manažment používateľov). Môžu byť flexibilne pridávané a odoberané z webového rozhrania portálu podľa požiadaviek prihláseného používateľa. Zobrazovaný výstup portálu je rozdelený na kontajnery a stránky. Každá stránka môže obsahovať niekoľko kontajnerov. Stránka môže obsahovať výstup viacerých portletov. Tieto výstupy sa zobrazia do rôznych kontajnerov. Každý portlet môže byť vizuálne minimalizovaný a maximalizovaný. Výstup portletov je vo forme fragmentov, nemôže teda obsahovať HTML tagy *body* a *html*. Portlety môžu medzi sebou komunikovať (*Inter-Portlet Communication*), čo umožňuje napr. zavolanie akcie "vyhľadávať" v jednom portlete a zobrazenie výsledkov v inom.

Existujú viaceré open source rámce na vývoj portálových riešení:

- Jetspeed
- eXo platform,
- Liferay Portal
- JBoss portal
- GridSphere

3.1.1. Jetspeed

Na základe obmedzení kladených na výsledný produkt (platforma J2EE, používanie FOSS) sme sa rozhodli pre analýzu portálových riešení, ktoré sú vyvíjané v projekte Apache Portals [1]. Jadrom tohto projektu je open source rámec na vývoj portálov Jetspeed-1, ktorý je dostupný v stabilnej verzii. Existuje už aj rámec Jetspeed-2, ktorý má lepšie navrhnutú architektúru a viac využíva programovanie pomocou komponentov. Je však nedostatočne zdokumentovaný a nie je dostupná jeho stabilná verzia.

Základné technológie použité v rámci Jetspeed-1 sú Java a XML. Jetspeed-1 je nezávislý na formáte prezentovaných dát a umožňuje integrovať obsah z rôznych zdrojov (napr. XML, RSS, SMTP). Obsah je možné pomocou transformácií XSL previesť do podoby, ktorá je najvhodnejšia pre zobrazenie na zariadení používateľa (napr. HTML, WML). Jetspeed-1 podporuje rámce na publikovanie obsahu Cocoon, WebMacro a Velocity. Ďalej podporuje vytváranie šablón a poskytuje niekoľko hotových portletov (napr. RSS portlet, XSLT portlet, JSP portlet). Podrobný návod na vyvíjanie aplikácií pomocou Jetspeed-1 je v [12].

Jetspeed-1 podporuje špecifikáciu *Java Portlet API JSR-168* [13] a je implementovaný ako servlet. Na nasadenie vyžaduje servlet kontajner kompatibilný so špecifikáciou *Servlet 2.3 API* (servlet kontajner Tomcat ju podporuje od verzie 4) a JDK 1.3 alebo vyšší spolu

s nainštalovaným *Java API for XML Processing (JAXP)*. Jetspeed-1 používa svoj vlastný bezpečnostný model na autorizovanie používateľov. Na ukladanie perzistentných dát implicitne používa databázu Hypersonic-SQL, ktorá sa dodáva spolu Jetspeed-1. Je však možné použiť aj inú databázu, ktorá podporuje JDBC 2.0. Pre mobilné zariadenia Jetspeed-1 podporuje špecifikáciu WML 1.1 a 1.2.

3.2. Ontologické úložiská

Ontológia je v kontexte informačných technológií „špecifikácia konceptualizácie“, alebo tiež formalizmus na zapísanie konceptov a vzťahov medzi nimi. Ontológie je možné použiť na modelovanie a ukladanie dát a znalostí o nich, pričom samotnú ontológiu je možné zapísať viacerými spôsobmi, napr. pomocou Resource Description Framework (RDF).

RDF je odporúčanie W3C, pôvodne určené pre modelovanie metadát, ale hodí sa aj všeobecne pre modelovanie dát. Je založené na trojiciach (subjekt, predikát, objekt), pričom objekt v jednej trojici môže byť subjektom v druhej trojici. Spájaním objektov pomocou predikátov vzniká graf. Existuje viacero jazykov určených na zápis modelu RDF. Na deklarovanie použitého slovníka je určená RDF Schema, pretože samotný RDF dátový model na to neposkytuje možnosti.

RDF Schema (RDFS) umožňuje definovať slovník pre RDF dáta. Opisuje napr. pravidlá, ktoré hovoria o tom, ktoré predikáty sa môžu použiť s danými objektmi. RDFS je validné RDF, pričom jediný rozdiel medzi RDF a RDFS je, že v RDFS je presne určená sémantika predikátov. Napr. predikát s názvom subClassOf špecifikuje hierarchickú organizáciu tried.

Identifikovali sme dve open source RDF databázy: Sesame [7] a Jena [11]. Sesame podporuje odvodzovanie a dopytovanie nad metadátami v RDF a RDF Schema; Jena okrem toho podporuje aj odvodzovanie a dopytovanie nad metadátami v OWL.

3.2.1. Sesame

Sesame na prácu potrebuje nejaké dátové úložisko. Môže ním byť relačná databáza, súborový systém alebo operačná pamäť. Prístup cez API je prostredníctvom HTTP, RMI alebo SOAP. Sesame podporuje viac dopytovacích jazykov. Na dopytovanie nepostačujú jazyky určené pre XML, lebo nepodporujú dopytovanie na sémantickej úrovni. Z tohto dôvodu sa používajú jazyky RQL a SeRQL. V súčasnosti sa ďalej vyvíja iba jazyk SeRQL [17].

Sesame sa môže používať buď ako knižnica jazyka Java alebo ako samostatný server. Pri použití Sesame ako knižnice je potrebné do premennej *classpath* pridať cestu k tejto knižnici. Pri serverovom použití je potrebné Sesame nainštalovať na niektorý servlet kontajner.

Sesame má administračný modul, ktorý umožňuje pridanie RDF/RDFS informácií a zmazanie celého úložiska. Ďalej obsahuje modul, umožňujúci export dát, schémy, alebo oboch častí súčasne.

3.3. Generovanie formulárov

Jednou z hlavných funkcionalít portálu je vloženie ponuky aktívnym producentom, prípadne jej editovanie, pričom výsledné zmeny sa musia prejaviť v ontologickom úložisku. Dôsledkom je potreba grafického používateľského rozhranie nad ontologickým úložiskom, ktoré by umožňovalo realizovať vzor CRUD. Keďže vytvorené riešenie má byť znovupoužiteľné aj v iných portálových riešeniach, nie je vhodné, aby formuláre boli „natvrdo“ naviazané na použitú ontológiu.

Z viacerých existujúcich spôsobov zápisu webových formulárov je momentálne najrozšírenejším využitie HTML forms, ktoré predstavuje možnosť zápisu formulára v jazyku

HTML. Takéto riešenie obmedzuje použitý značkovací jazyk na HTML. Novým štandardom v tejto oblasti sa stáva XForms, ktorého výhodou je oddelenie dát od prezentácie, definovanie dát jazykom XML a následná hardvérová nezávislosť formuláru.

Nie je nám známy žiadny nástroj, ktorý by dokázal vygenerovať grafické rozhranie na editovanie záznamov v ontologickom úložisku a preto našou úlohou bude vytvoriť takýto nástroj. Účelom nástroja je vygenerovanie zápisu formuláru v jazyku XForms, pomocou informácií uložených v ontologickom úložisku a prípadných metaúdajov o ontológii vo formáte XML.

3.4. Aplikačné rámce pre prezentáciu

Existuje niekoľko voľne dostupných implementácií prezentačných rámcov pre architektúru dátovodov a filtrov, ktoré sú určené na tvorbu webových systémov. Pri tvorbe portálu pracovných príležitostí sme uvažovali dva z nich – Cocoon a Orbeon.

3.4.1. Cocoon

Publikačný rámec Cocoon je voľne dostupná komponentovo založená implementáciá v jazyku Java, ktorá poskytuje prostriedky pre dynamické generovanie dokumentov prostredníctvom definovaných zreťazení. Jednotlivé zreťazenia sú definované centrálne, pričom dáta v zreťazení produkuje generátor. Následne môžu byť transformované sériou transformátorov a nakoniec poskytnuté na výstup v cieľovom formáte prostredníctvom serializátora. Každé zreťazenie obsahuje práve jeden generátor, nula a viac transformátorov a práve jeden serializátor.

Cocoon poskytuje niekoľko štandardných generátorov, transformátorov a serializátorov, nazývaných komponenty a tiež implementačné rozhranie pre tvorbu vlastných komponentov. Najčastejšie sú implementované vlastné akcie, ktorých úlohou je zmena stavu systému, mnohokrát využívané ako prostriedok pre zmenu toku riadenia. Okrem toho Cocoon poskytuje možnosti znovupoužitia definovaných zreťazení pomocou zdrojov, ako aj pohľady určené pre sledovanie dát prúdiacich medzi jednotlivými transformátormi, bez nutnosti zmeny definície zreťazenia.

3.4.2. Orbeon

Prezentačný server Orbeon je tiež komponentovo založená implementácia v jazyku Java, ktorá bola pôvodne vyvíjaná ako komerčný produkt a neskôr bola zverejnená pod licenciou LGPL. Tento prezentačný rámec sa v mnohých aspektoch podobá publikačnému rámcu Cocoon. Medzi najväčšie rozdiely patrí väčšia flexibilita pri definovaní zreťazení v prípade rámca Orbeon. Kým Cocoon neumožňuje výstup jedného komponentu poskytnúť ako vstup viacerým komponentom súčasne, rámec Orbeon to umožňuje.

Ďalším rozdielom je podpora technológií XPath 2.0, XQuery a XUpdate. Orbeon spomínané technológie interne podporuje, Cocoon zatiaľ nie. Rovnako syntaktická kontrola správnosti štruktúry dokumentu počas spracovania zreťazenia je možná iba v prezentačnom serveri Orbeon. Veľkým nedostatkom prezentačného servera Orbeon je nízky počet existujúcich komponentov a jeho slabá integrácia s produktmi tretích strán.

Rozdielny je tiež prístup k definovaniu toku riadenia. Kým v rámci Cocoon sa tok riadenia realizuje pomocou akcií, prípadne skriptovacieho jazyka FlowScript, Orbeon poskytuje vlastný jazyk založený na jazyku XML integrujúci prístup separácie dát, toku riadenia a spôsobu prezentácie.

3.4.3. Rozdiely rámca Cocoon a prezentačného serveru Orbeon

Identifikovali sme viaceré rozdiely medzi uvedenými rámcami. Príloha B obsahuje prehľad ich nízko-úrovňových rozdielov. Hlavné vysoko-úrovňové rozdiely sme zhrnuli do nasledovných bodov (zo strany serveru Orbeon):

- Väzba na XForms. OPS obsahuje implementáciu XForms na strane servera, ktorá je postavená na XForms štandarde.
- XML Pipeline Language (XPL). XPL je viac všeobecný a flexibilný ako Cocoon sitemap. Deklarácie stránok sú adresované pomocou OPS Page Flow Controlera, čo je separátny komponent. XPL je jednoduchý, deklaratívny jazyk, ktorý umožňuje spravovanie XML komponentov. Má zabudovanú agregáciu, podmienky, iterácie, validáciu s W3C XML Schema a Relax NG. OPS implementácia jazyka XPL je založená na SAX.
- Minimálna potreba písania Java kódu. Kombinácia XML technológií v OPS spôsobuje, že je pravdepodobné vytvorenie prezentačného kódu bez Java kódu.
- Silnejšia platformová konzistencia. Všetky technológie nachádzajúce sa v OPS ako Page Flow, XForms, XPL, XSLT a XML Schema sú navrhnuté tak, aby pracovali spolu na zachytávaní, prezentovaní dát a formátovaní XML dokumentov.
- Silnejšia väzba k XML štandardom. OPS podporuje a integruje Relax NG, XSLT 2.0, a XPath 2.0 špecifikácie, ktoré rámec Cocoon nepodporuje.
- Silnejšia separácia závislostí. OPS má silnú separáciu závislostí medzi stránkami (založených na separátnych modeloch a pohľadoch), formami, Page Flow a procesmi implementovanými pomocou XPL.
- Deklaratívny Page Flow Controller (PFC), podporujúci deklaratívny tok stránok (page flow) pre celú aplikáciu. Je navrhnutý na spoluprácu so serverovou XForms implementáciou OPS na sledovanie architektúry MVC.

3.5. Editory na návrh grafického rozhrania portálu

Pri tvorbe webového portálu je potrebné vytvoriť aj jeho grafický návrh. Existuje veľký počet editorov určených práve na tvorbu webu a webových aplikácií. Základné delenie editorov možno vykonať podľa toho, či podporujú prácu v grafickom režime alebo v textovom režime:

- WYSIWYG editory podporujú tvorbu stránok v grafickom režime, v ktorom autor stránky hneď pri jej tvorbe vidí, ako bude výsledná stránka vyzerať v prehliadači. Výhodou týchto editorov je jednoduchosť a intuitívnosť tvorby stránok.
- neWYSIWYG editory podporujú len editovanie zdrojového kódu stránky. Autor stránky pri práci nemá možnosť vidieť výsledný vzhľad stránky. Tento si môže pozrieť až po dokončení stránky v prehliadači. Napriek tejto skutočnosti sa tieto editory odlišujú od klasických textových editorov podporou pri editovaní štýlov, HTML tagov, alebo pri práci s viacerými zdrojovými súbormi.
- Hybridné editory, ktoré kombinujú oba prístupy a umožňujú editovanie stránok v grafickom režime a súčasne aj editovanie zdrojového kódu stránok [9].

Grafický návrh portálov ja v súčasnosti založený na využití kaskádových štýlov. Z tohto dôvodu je dôležitým aspektom pri výbere editora práve podpora práce so štýlmi. Medzi ďalšie dôležité parametre patrí používateľské prostredie, podpora jednotlivých HTML tagov a

jednoduché vytváranie tabuliek. Bližšie analyzujeme dva hybridné nástroje, ktoré najviac vyhovujú uvedeným požiadavkám – Macromedia Dreamweaver a NVU.

3.5.1. Macromedia Dreamweaver

Patrí medzi najznámejšie editory svojej triedy. Keďže už prešiel dlhším vývojom a viacerými verziami, v mnohých ohľadoch patrí medzi veľmi kvalitné a robustné nástroje s prepracovaným prostredím a funkciami. Jeho hlavné vlastnosti sú:

- možnosť editácie dokumentov v grafickom i zdrojovom režime
- podpora práce so štýlmi, podpora rozloženia prvkov na stránke pomocou CSS
- veľmi dobrá podpora pre prácu s tabuľkami
- podpora testovania kódu v rôznych prehliadačoch
- nástroj pre komfortný manažment webového sídla (hierarchická štruktúra lokálneho i vzdialeného webu)
- zabudovaný FTP klient
- podpora práce so šablónami
- ladenie JavaScriptov, integrovaný JavaScript debugger

3.5.2. NVU

Open source nástroj NVU je vo svojej verzii 1.0 pomerne nový a následne ešte neprešiel takým vývojom ako Dreamweaver. Odzrkadľuje sa to na niektorých jeho funkciách, ktoré ešte nie sú dostatočne prepracované. Najdôležitejšími vlastnosťami NVU sú:

- možnosť editácie v grafickom i zdrojovom režime
- zabudovaný FTP klient
- podpora tvorby formulárov a tabuliek
- podpora práce so šablónami
- nástroj na prácu so štýlmi
- podpora manažmentu webového sídla, práca s viacerými súbormi súčasne
- podpora platformy Windows aj Linux

3.6. Integrované vývojové prostredie

Z hľadiska efektívnej tvorby softvérových systémov je dôležité použitie integrovaných vývojových prostredí, ktoré poskytujú všetku potrebnú funkcionalitu prostredníctvom jedného rozhrania. Ideálne takéto prostredie tiež podporuje spoluprácu s rôznymi ďalšími nástrojmi na riadenie verzií súborov a generovanie dokumentácie.

3.6.1. Eclipse

Integrované vývojové prostredie Eclipse [5] podporuje efektívnu a pohodlnú tvorbu softvéru. Prostredie je primárne určené na vývoj aplikácií v jazyku Java. Základné prostredie neponúka veľké množstvo nástrojov, je však možné ho podľa potreby rozšíriť o množstvo zásuvných

modulov a následne pohodlne aktualizovať. Správne nakonfigurované prostredie Eclipse ponúka celý rad pohľadov, editorov, sprievodcov, a iných nástrojov.

3.7. Zdieľanie výsledkov práce členov tímu

Pri riešení projektu, na ktorom pracuje väčší počet ľudí, treba vyriešiť problém zdieľania spoločných súborov, či už ide o zdrojové kódy, dokumentáciu alebo iné súbory.

Základným problémom je spôsob zdieľania informácií používateľmi bez toho, aby si navzájom ničili prácu neustálym prepisovaním svojich zmien. V súčasnosti sú známe dva základné modely riešenia tohto problému: *lock-modify-unlock* a *copy-modify-merge* [3]. Prvý z nich je v podstate aplikáciou postupov známych z problémov synchronizácie systémového programovania. Pri zdieľaní je tento model nevhodný pre svoju prílišnú reštriktívnosť voči používateľom a možné administratívne problémy.

Copy-modify-merge model umožňuje každému používateľovi prácu nad jeho osobnou kópiou súborov z úložiska a následné spájanie týchto kópií do novej verzie v úložisku. Proces spájania môže byť do určitej miery podporovaný systémom na správu verzií, hlavná zodpovednosť však leží na používateľovi.

3.7.1. Subversion

Subversion je open source systém na kontrolu verzií ľubovoľnej kolekcie súborov. Umožňuje efektívne zdieľanie súborov, pretože má rozhranie do celosvetovej siete Internet a poskytuje vysokú bezpečnosť uloženia súborov tým, že si pamätá každú vykonanú zmenu.

Medzi hlavné prednosti nástroja Subversion patria:

- Posielanie diff (rozdielov medzi súbormi) v oboch smeroch. Iné nástroje spravidla posielajú diff iba smerom od serveru ku klientovi a opačne posielajú celé súbory.
- Viacero možností prístupu k úložisku – riadkový klient, http prístup cez WebDAV/Delta V, plugin do vývojového prostredia Eclipse, klienti tretích strán).
- Každá zmena zvyšuje číslo verzie, nielen zmena obsahu súborov.

3.8. Tvorba technickej dokumentácie

3.8.1. Javadoc

Štandardom v tvorbe dokumentácie pre Java aplikácie je nástroj Javadoc [10], slúžiaci na generovanie dokumentácie zo zdrojových kódov aplikácií. Potrebné informácie čerpá z deklarácií a zo špeciálnych komentárov v zdrojových kódoch. Následne z nich dokáže vytvoriť sadu HTML stránok opisujúcich public a protected triedy, rozhrania, konštruktory, metódy a premenné. Nástroj je možné spustiť na celé balíky alebo na samostatné zdrojové súbory. Výsledná dokumentácia je zobrazovaná v štruktúrovanej a dobre organizovanej forme, prispôsobenej rýchlemu vyhľadávaniu informácií.

Obsah a formát dokumentácie je možné prispôsobovať použitím tzv. doclets – programov, ktoré špecifikujú obsah a formát výstupu nástroja Javadoc. Nástroj štandardne používa svoj vlastný vstavaný doclet, ktorý je možné ďalej modifikovať. Alternatívne je možné si vytvoriť vlastný doclet a generovať tak vlastnú výstupnú dokumentáciu vo formáte HTML, XML, MIF alebo RTF. V každom behu programu sa vygeneruje celá dokumentácia. Nástroj nepracuje inkrementálne a nedokáže modifikovať alebo priamo začleniť výsledky predchádzajúcich behov do novo vytváranej dokumentácie.

Javadoc je závislý na Java kompilátore a pri svojom behu volá časti kompilátora na preklad deklarácií. Pre generovanie dokumentácie nie je potrebné, aby bol zdrojový kód kompletný a bezchybný. Nástroj však musí byť schopný nájsť všetky triedy, ktoré sú v zdrojovom kóde použité. Pravidlá písania komentárov môžeme nájsť v [4][8][10][16]. V Javadoc komentároch je možné použiť kľúčové slová – tagy, pomocou ktorých je možné doplniť dokumentáciu o rôzne rozširujúce informácie. Text komentárov môže byť navyše písaný v jazyku HTML, čo umožňuje ďalšiu úpravu výstupnej dokumentácie.

4. Špecifikácia riešenia

V tejto kapitole opíšeme špecifikáciu systému pomocou špecifikácie prípadov použitia, ktoré poskytujú prehľad funkcionality systému – definujú funkcionálne požiadavky. Uvádzame aj pohľady na prípady použitia vo forme diagramov prípadov použitia, ktoré sprehľadňujú vzťahy medzi jednotlivými prípadmi použitia a znázorňujú spoluprácu podsystémov. Vyjadríme sa tiež k nefunkcionálnym požiadavkám na riešenie, ktoré súvisia so skutočnosťou, že portál je vytváraný v kontexte väčšieho systému.

4.1. Prípady použitia

Pri špecifikácii prípadov použitia sme vychádzali z prípadov použitia identifikovaných v rámci štátneho programu NAZOU [14]. Z prípadov použitia sme vybrali tie hlavné a doplnili sme ďalšie na základe novo identifikovaných požiadaviek. Každý z prípadov použitia sme označili jedinečným identifikátorom v tvare UCXX. Každému prípadu použitia sme priradili prioritu v rozsahu 1-3. Význam pridelených priorít je uvedený v nasledovnej tabuľke:

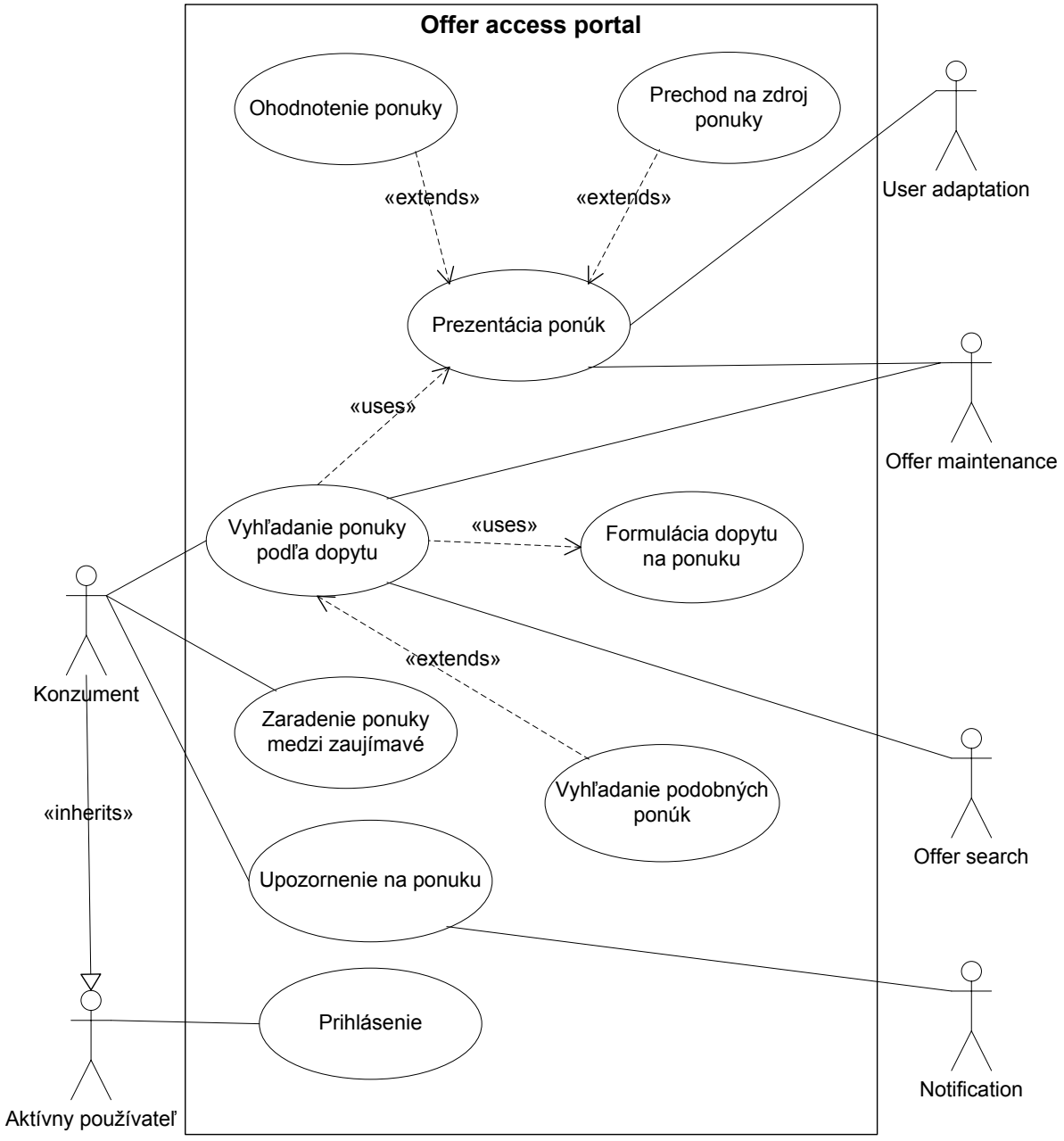
Priorita	Slovný opis
1	Vysoká
2	Stredná
3	Nízka

Podsystému Offer access portal poskytuje ponuky konzumentovi (pozri Obr. 2). Umožňuje sformulovanie dopytu, na základe ktorého sa vyhľadajú a prezentujú ponuky. Konzument má možnosť ohodnotiť ponuku, prejsť na zdroj ponuky, vyhľadať podobné ponuky alebo s vyhľadanou ponukou ďalej pracovať (napr. vytlačiť, pridať medzi obľúbené).

Identifikátor:	UC01	**Názov:**	Prihlásenie	**Priorita:**	1
Opis:		Používateľ sa prihlási do systému pomocou svojho prihlasovacieho mena a hesla. Od tejto chvíle je s používateľom asociovaný model používateľa.			
Scenár:		**Krok:**	**Činnosť:**		
Prihlásenie		1.	Používateľ zadá prihlasovacie meno a heslo.		
		2a.	Systém overí používateľove meno a heslo a prihlási ho do systému.		
		2b.	V prípade zadania nesprávnych prihlasovacích údajov systém odmietne používateľa prihlásiť a vráti ho na pôvodný prihlasovací formulár s oznámením o chybe.		

Identifikátor:	UC02	**Názov:**	Vyhľadanie ponuky podľa dopytu	**Priorita:**	1
Opis:		Na základe dopytu používateľa sa vytvorí usporiadaný zoznam ponúk, ktoré vyhovujú danému dopytu. Dopyt môže byť rozšírený o dodatočné, používateľom explicitne nevyjadrené kritériá. Zahŕňa formuláciu dopytu.			
Scenár:		**Krok:**	**Činnosť:**		
Vyhľadanie ponúk		1.	Konzument zadá dopyt (prípad použitia UC03).		
		2.	Dopyt je rozšírený o dodatočné kritériá.		
		3.	Podsystém *Offer search* poskytne portálu ponuky na základe dopytu.		

	4.	Podsystém *Offer maintenance* poskytne portálu dostupné informácie o nájdených ponukách.
	5.	Portál zobrazí ponuky konzumentovi (prípad použitia UC04).



Obr. 2 Diagram prípadov použitia podsystému Offer access portal

Identifikátor:	UC03	**Názov:**	Formulácia dopytu na ponuku	**Priorita:**	1
Opis:		Sformulovanie požiadaviek na hľadané ponuky. Funkcionalita môže byť špecializovaná pre konkrétnu realizáciu, ako je napríklad použitie dopytovacieho jazyka, formulára, alebo reštriktívneho prehľadávania. Konkrétna realizácia môže byť ďalej špecializovaná, napríklad použitím fuzzy alebo bool logiky.			
Scenár:		**Krok:**	**Činnosť:**		
Nové vyhľadávanie		1.	Konzument zvolí nové vyhľadávanie.		

	2.	Portál odstráni z aktuálneho dopytu všetky kritériá.
Zadanie kritéria	1.	Konzument zadá kritérium vyhľadávania.
	2.	Portál pridá kritérium do aktuálneho dopytu.
Odobratie kritéria	1.	Konzument vyberie kritérium vyhľadávania ktoré bolo zadané skôr.
	2.	Portál odstráni vybrané kritérium z aktuálneho dopytu.

Identifikátor:	UC04	**Názov:**	Prezentácia ponúk	**Priorita:**	1
Opis:		Ponuky sa prezentujú v tvare, ktorý si konzument zvolil, prípadne bol tento stav odsledovaný podsystémom *User adaptation*. Na prezentáciu ponuky je použitý prezentačný rámec využívajúci prezentačnú ontológiu Fresnel (nie je to len kópia stránky).			
Scenár:		**Krok:**	**Činnosť:**		
Zobrazenie ponúk		1.	Podsystém *Offer maintenance* poskytne informácie o vyhľadaných ponukách.		
		2.	Podsystém *User adaptation* poskytne spôsob zobrazenia pre aktívneho konzumenta.		
		2a.	Ak podsystém *User adaptation* neposkytne žiadne informácie o aktívnom konzumentovi, použije sa štandardný spôsob zobrazenia.		
		3.	Na základe aktuálneho spôsobu zobrazenia sa zobrazia vyhľadané ponuky.		
Zmena spôsobu zobrazenia ponúk		1.	Konzument zvolí iný spôsob zobrazenia		
		2.	Podsystém *User adaptation* upraví profil konzumenta		
		3.	Portál zobrazí ponuky zvoleným spôsobom		

Identifikátor:	UC05	**Názov:**	Ohodnotenie ponuky	**Priorita:**	3
Opis:		Konzument ohodnotí vhodnosť ponuky vzhľadom k svojim potrebám. Túto informáciu použije podsystém *User adaptation* pre úpravu profilu konzumenta a podsystém *Offer maintenance* pre úpravu metadát o ponuke, ktoré môže pomôcť ostatným konzumentom.			
Scenár:		**Krok:**	**Činnosť:**		
Ohodnotenie ponuky		1.	Portál prezentuje formulár pre zadanie štruktúrovaného hodnotenia ponuky.		
		2.	Konzument ohodnotí ponuku .		
		3.	Podsystém *User adaptation* upraví profil konzumenta. Podsystém *Offer maintenance* upraví metadáta príslušnej ponuky.		

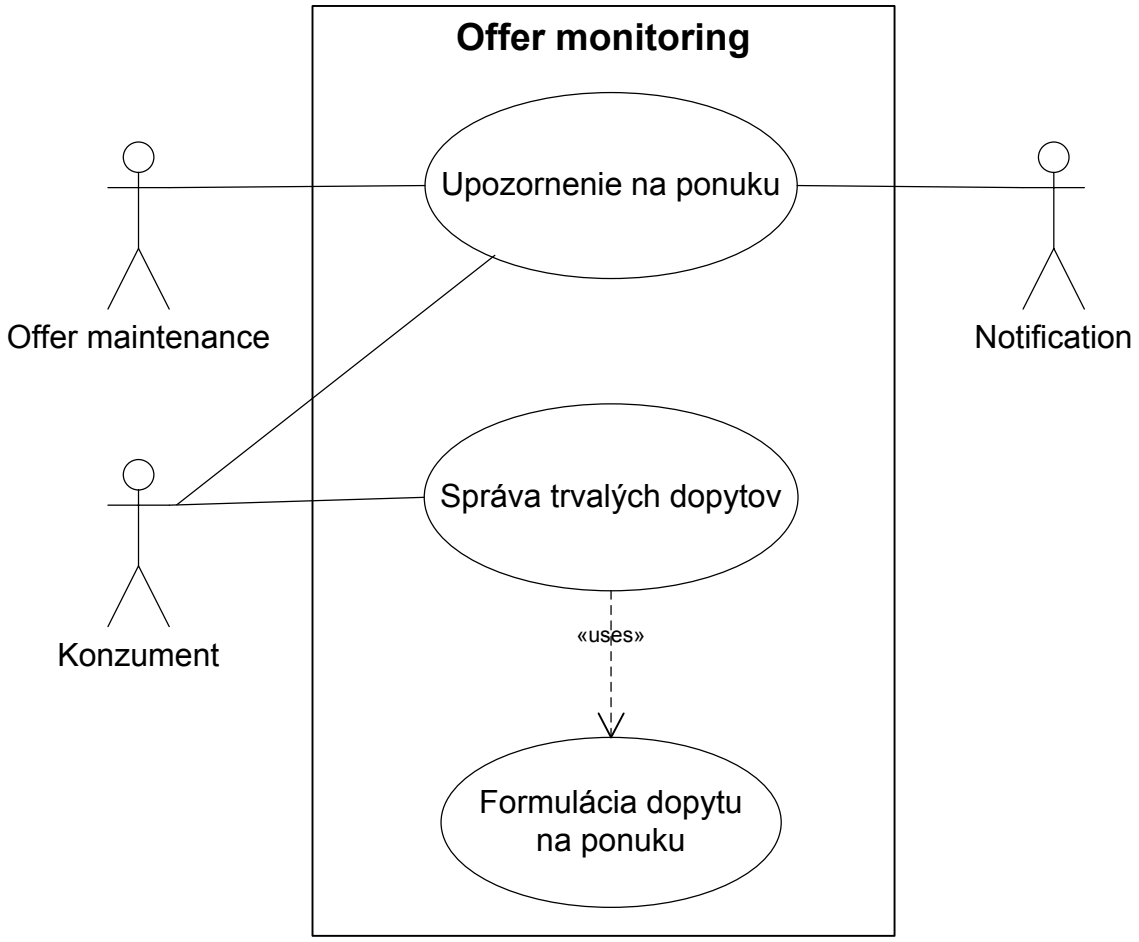
Identifikátor:	UC06	**Názov:**	Vyhľadanie podobných ponúk	**Priorita:**	3
Opis:		K skôr nájdeným ponukám sa vyhľadajú podobné na základe zvolených kritérií.			
Scenár:		**Krok:**	**Činnosť:**		
Zobrazenie podobných ponúk		1.	Konzument vyberie ponuku.		
		2.	Portál vytvorí dopyt na základe podobnosti s vybranou ponukou.		

Identifikátor:	UC07	**Názov:**	Zaradenie ponuky medzi zaujímavé	**Priorita:**	1
Opis:		Pri konkrétnej ponuke má konzument možnosť ďalšej práce s ponukou – možnosť zaradiť ponuku medzi svoje zaujímavé ponuky, prípadne si k zaujímavej ponuke zaznamenať poznámku. Konzument si môže ponuku vytlačiť – systém mu poskytne pdf súbor s obsahom ponuky.			
Scenár:		**Krok:**	**Činnosť:**		
Zaradenie medzi zaujímavé ponuky		1.	Konzument vyberie ponuku, ktorú chce zaradiť medzi zaujímavé ponuky.		
		2.	Portál zaradí ponuku do zoznamu zaujímavých ponúk pre daného konzumenta. (Zoznam sa mu zobrazí po prihlásení.)		
Zaznamenanie súkromnej poznámky k zaujímavej ponuke		1.	Portál prezentuje formulár pre zadanie súkromnej poznámky konzumenta k ponuke.		
		2.	Konzument zadá súkromnú poznámku, ktorá sa uloží k ponuke.		

Identifikátor:	UC08	**Názov:**	Upozornenie na ponuku	**Priorita:**	2
Opis:		Pri konkrétnej ponuke má konzument možnosť elektronickou poštou upozorniť na túto ponuku inú osobu.			
Scenár:		**Krok:**	**Činnosť:**		
Notifikácia o ponuke		1.	Konzument vyberie ponuku, na ktorú chce upozorniť iného konzumenta (ktorý nemusí byť Aktívnym konzumentom v našom systéme)		
		2.	Portál prezentuje formulár s návrhom správy, ktorá sa má odoslať		
		3.	Konzument zadá e–mailovú adresu adresáta		
		4.	Podsystém *Notification* odošle správu		

Identifikátor:	UC09	**Názov:**	Prechod na zdroj ponuky	**Priorita:**	1
Opis:		Pre konkrétnu ponuku je možné prejsť na jej pôvodný zdroj a tak zistiť pôvodné nespracované informácie.			
Scenár:		**Krok:**	**Činnosť:**		
Prechod na zdroj ponuky		1.	Konzument vyberie ponuku, ktorej zdroj chce vidieť.		
		2.	Portál zobrazí pôvodný zdroj vybranej ponuky.		

Na Obr. 3 je znázornené použitie podsystému Offer monitoring. Tento podsystém umožňuje konzumentovi zadať trvalý dopyt na ponuku. Pri výskyte ponuky, ktorá spĺňa kritériá dopytu je konzument upozornený podsystémom Notification.

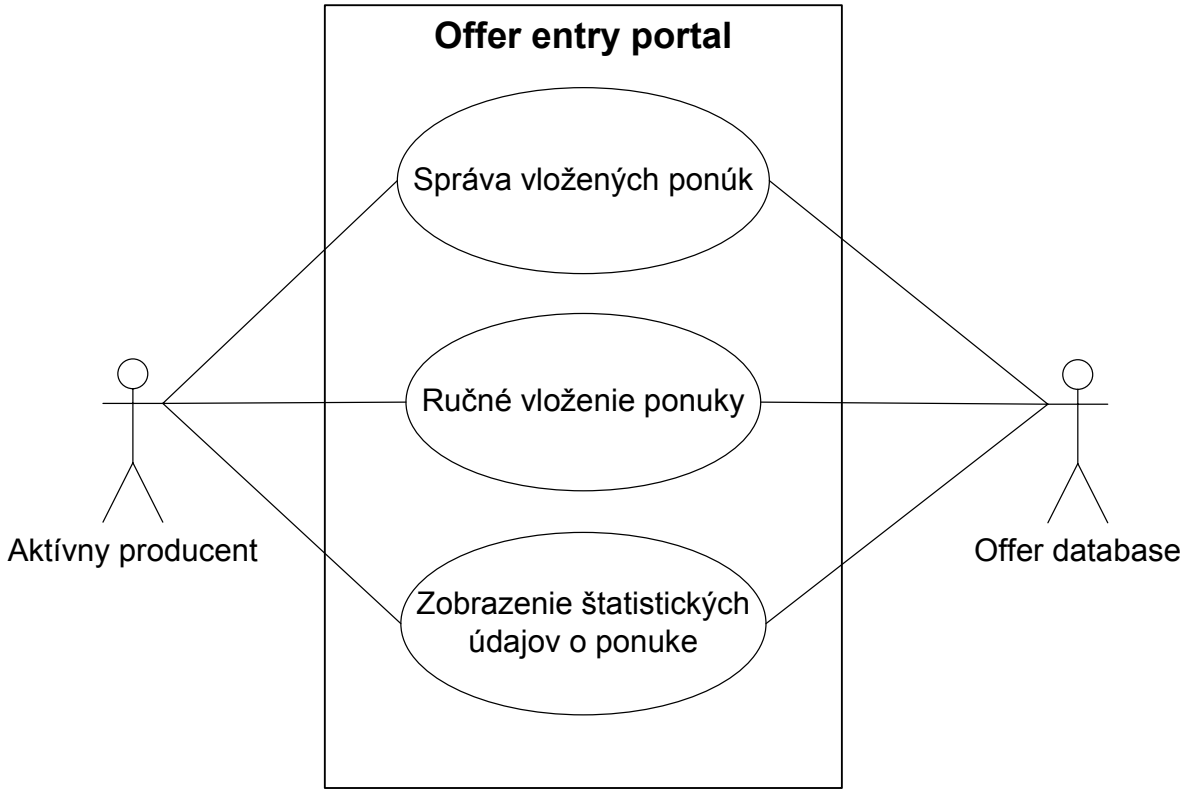


Obr. 3 Diagram prípadov použitia podsystému Offer monitoring

Identifikátor:	UC10	**Názov:**	Správa trvalých dopytov	**Priorita:**	2
Opis:		Tento prípad zahŕňa správu uložených dopytov, ktoré systém automaticky sleduje v mene konzumenta. Zahŕňa prípad použitia UC03.			
Scenár:		**Krok:**	**Činnosť:**		
Pridanie úlohy		1.	Používateľ špecifikuje dopyt, ktorý má byť trvale hľadaný.		
		2.	Podsystém *Offer search* vytvorí uložený dopyt.		
		3.	Podsystém *Offer monitoring* zaradí sledovanie do zoznamu.		
Odobratie úlohy		1.	Portál ponúkne zoznam trvalých dopytov pre daného konzumenta.		
		2.	Používateľ vyberie dopyt, ktoré chce zmazať.		
		3.	Podsystém *Offer search* zmaže uložený dopyt.		
		4.	Podsystém *Offer monitoring* vyradí sledovania zo zoznamu naplánovaných úloh.		

Identifikátor:	UC11	**Názov:**	Upozornenie na ponuku	**Priorita:**	2
Opis:		Upozornenie na ponuku slúži na aktívne upovedomenie konzumenta na výskyt ponuky, ktorá vyhovuje jeho uloženým trvalým hľadaniam. Obsahuje odoslanie správy konzumentovi rôznou formou.			

Scenár:	Krok:	Činnosť:
Upozornenie na ponuku	1.	Podsystém *Offer monitoring* zistí výskyt ponuky, ktorá vyhovuje uloženému trvalému dopytu a nebola ešte oznámená.
	2.	Podsystém *Offer maintenance* dodá informácie o ponuke konzumentovi (adresátovi).
	3.	Pre každý kontakt na konzumenta podsystém *Notification* sformátuje správu a odošle ju.
	4.	Konzument prijme správu.



Obr. 4 Diagram prípadov použitia podsystému Offer entry portal

Prípady použitia podsystému Offer entry portal sú zobrazené na Obr. 4. Tento podsystém umožňuje aktívnemu producentovi zadať nové ponuky, udržovať existujúce ponuky ako aj získať štatistické informácií o ponukách.

Identifikátor:	UC12	Názov:	Ručné vloženie ponuky	Priorita:	1
Opis:		Ručné vloženie ponuky aktívnym producentom.			
Scenár:		Krok:	Činnosť:		
Pridanie ponuky		1.	*Offer entry portal* ponúkne formulár pre zadanie ponuky.		
		2.	Aktívny producent zadá informácie o ponuke.		
		3.	Podsystém *Offer database* zabezpečí uloženie ponuky.		

Identifikátor:	UC13	Názov:	Správa ručne vložených ponúk	Priorita:	1
Opis:		Správa ponúk, ktoré aktívny producent ručne vložil do systému.			
Scenár:		Krok:	Činnosť:		

Modifikácia ponuky	1.	Portál ponúkne zoznam ponúk, ktoré zadal prihlásený producent.
	2.	Producent vyberie ponuku, ktorú chce modifikovať.
	3.	Portál prezentuje formulár s aktuálnymi informáciami o ponuke.
	4.	Producent urobí potrebné zmeny.
	5.	Podsystém *Offer database* aktualizuje záznam o ponuke.
Zmazanie ponuky	1.	Portál ponúkne zoznam ponúk, ktoré zadal prihlásený producent.
	2.	Producent vyberie ponuku, ktorú chce zmazať.
	3.	Podsystém *Offer database* odstráni záznam o ponuke z databázy.
Deaktivácia ponuky	1.	Portál ponúkne zoznam ponúk, ktoré zadal prihlásený producent.
	2.	Producent vyberie ponuku, ktorú chce zmazať.
	3.	Podsystém *Offer database* deaktivuje ponuku. Ponuka ostáva uložená v databáze, prístupná producentovi, neprístupná konzumentovi.

Identifikátor:	UC14	**Názov:**	Zobrazenie štatistických údajov o ponuke	**Priorita:**	2
Opis:		Producent si môže dať zobraziť štatistické údaje o ponuke, ktorú zadal do systému (počet zobrazení, celkový čas zobrazenia ponuky, priemerný čas zobrazenia ponuky, priemerné hodnotenie ponuky, typicky profil konzumenta,...)			
Scenár:		**Krok:**	**Činnosť:**		
Zobrazenie štatistických údajov o ponuke		1.	Portál ponúkne zoznam ponúk, ktoré zadal prihlásený producent.		
		2.	Producent vyberie ponuku, ktorej štatistiku si chce pozrieť.		
		3.	Portál prezentuje formulár s aktuálnymi štatistickými dátami o ponuke.		

4.2. Nefunkcionálne požiadavky

Skutočnosť, že portál pracovných príležitostí má nadviazať na existujúce výsledky zo štátneho programu NAZOU a má ambíciu stať sa jeho súčasťou vedie k niekoľkým dodatočným požiadavkám na proces jeho vytvárania ako aj na portál samotný.

Program NAZOU má predovšetkým výskumný charakter a jeho cieľom nie je implementácia komerčného riešenia. Väčší dôraz sa kladie na univerzálnosť, kompatibilitu s medzinárodnými štandardmi, viaceré spôsoby prezentácie informácií a využitie moderných prístupov webu so sémantikou a reprezentácie dát pomocou ontológií. Tieto skutočnosti vedú k vyšším požiadavkám na kvalitu návrhu, implementácie a dokumentácie riešenia.

Výsledný produkt by mal byť navrhnutý a implementovaný ako skupina spolupracujúcich nástrojov, pričom vývoj nástrojov ako aj nástroje samotné by mali dodržiavať štandardy a pravidlá vývoja schválené v rámci programu NAZOU. Portál by mal byť vyvinutý za použitia open source riešení, jednotlivé časti riešenia by mali byť prehľadne a konzistentne zdokumentované, aby bolo možné v práci na portáli pokračovať aj v budúcnosti.

Projekt tvorby portálu pracovných príležitostí je zasadený do školských podmienok, kde sa vyvíja v rámci predmetu Tvorba softvérového systému v tíme. Z tohto dôvodu pri vývoji portálu nie sú pre nás prvoradé vlastnosti ako bezpečnosť, spoľahlivosť a dostupnosť, aj keď si uvedomujeme, že v prípade komerčného vývoja portálu by na ne bol kladený veľký dôraz.

V súlade s etickými princípmi musí produkt jasne deklarovať smerom k používateľom fakt, že sú o nich zbierané dáta za účelom vytvorenia modelu používateľa a následného prispôsobovania sa portálu používateľom. Každý používateľ musí mať prístup k údajom o ňom uložených v modeli používateľa a musí mať možnosť ho zmeniť.

Portál musí poskytovať svojim používateľom primeranú úroveň zabezpečenia:

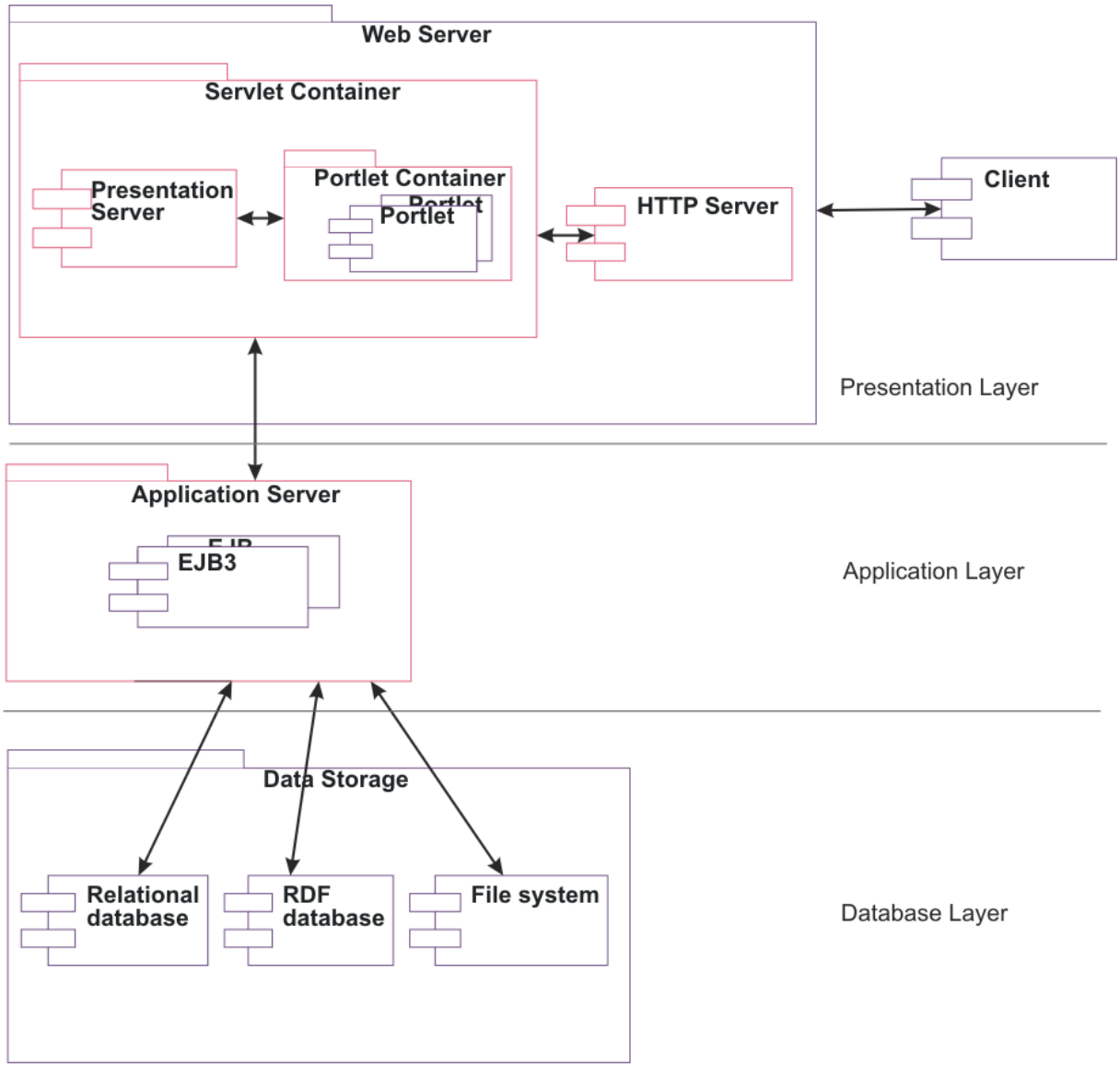
- Prístupové heslá musia byť prenášané šifrovane.
- Používateľ nesmie mať právo na zmenu ponuky, ktorá nebola ním zadaná.
- Používateľ smie pristupovať výhradné k svojmu modelu používateľa.
- Systém musí byť zabezpečený voči prípadným „spamovacím“ robotom, ktorý by sa mohli pokúšať vložiť do systému veľké množstvo nezmyselných ponúk.

5. Hrubý návrh riešenia

Kapitola sa venuje opisu hrubého návrhu portálu pracovných príležitostí. Uvádza globálny pohľad na vytváraný systém opisom jeho architektúry a venuje sa spôsobu notifikácie systému o zmenách v úložisku. V druhej časti kapitoly je opísaný grafický návrhu webu portálu a navrhnutý navigačný model portálu. Záver kapitoly tvorí zhodnotenie analýzy nástrojov z kapitoly 3 a voľba metodológie testovania.

5.1. Architektúra systému

Pri návrhu architektúry sme vychádzali z prostredia J2EE. Obmedzenia dané širším kontextom štátneho programu NAZOU nás viedli k použitiu architektonických vzorov, ktoré sú súčasťou J2EE.



Obr. 5 Architektúra portálu pracovných príležitostí

Napriek tomu, že je projekt primárne zameraný na doménu pracovných príležitostí, sme sa rozhodli architektúru poňať všeobecnejšie, aby ju bolo možné neskôr použiť len s malými zmenami pre všeobecnejšiu doménu získavania a poskytovania akýchkoľvek ponúk. Naším

cieľom je preniesť túto nezávislosť od konkrétneho druhu ponúk do výslednej implementácie a perspektívne tak vytvoriť rámec na vytváranie podobných portálov. Keďže je predpoklad, že sa výsledný produkt bude rozširovať aj o vlastnosti, ktoré nie sú súčasťou špecifikácie obsiahnutej v tomto dokumente, je architektúra navrhnutá komplexnejšie, ako by bolo potrebné na splnenie špecifikovaných požiadaviek.

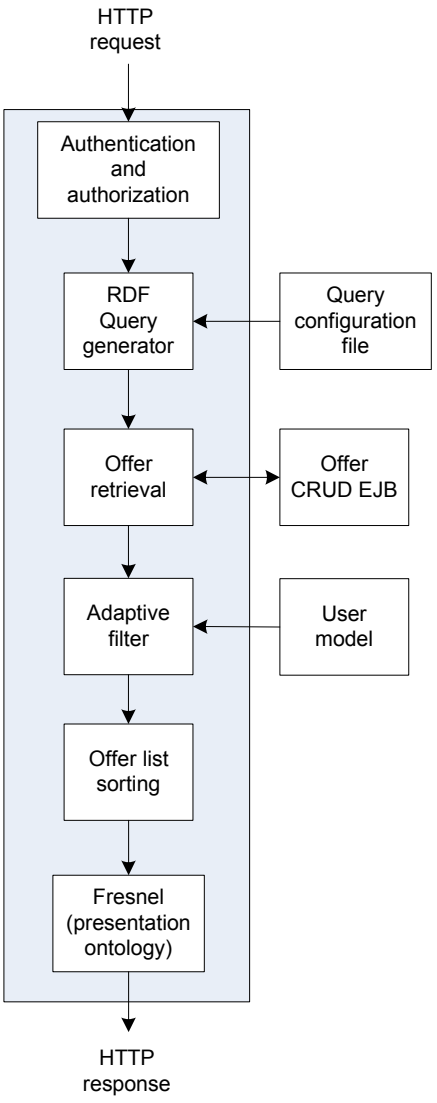
Architektúra portálu pozostáva z troch vrstiev, znázornených na Obr. 5. Červenou farbou sú vyznačené už existujúce komponenty tretích strán, ktoré len využijeme pri tvorbe portálu. Najvyššou je prezentačná vrstva, ktorá slúži na komunikáciu medzi používateľom a portálom. Aplikačná vrstva, zobrazená v strede, implementuje objektový model domény. Najnižšou vrstvou je vrstva dátových zdrojov, ktorá zabezpečuje komunikáciu s relačnou databázou, RDF databázou a súborovým systémom.

5.1.1. Prezentačná vrstva

Prezentačná vrstva využíva architektonický vzor dátovod na generovanie fragmentov webových stránok v jazyku XHTML verzie 1.0. Pre každú funkciu portálu definujeme samostatný dátovod, pozostávajúci z postupnosti troch druhov komponentov: generátorov, transformátorov a serializátorov. Na začiatku dátovodu sa nachádza generátor generujúci dáta, ktoré sa ďalej konvertujú do inej podoby pomocou postupnosti transformátorov a nakoniec sa serializátorom prevedú na reťazec znakov. Výber vhodného dátovodu na obslúženie požiadavky vykonáva komponent mapa na základe parametra s názvom *pageid* HTTP požiadavky, ktorý sa prenáša HTTP metódou GET. Systém využije existujúcu implementáciu mapy a dátovodu z rámca Cocoon. Navrhli sme použitie nasledovných dátovodov:

- generovanie formulára na zadanie ponuky,
- vytvorenie ponuky,
- zrušenie ponuky,
- generovanie formulára na vyhľadávanie,
- vyhľadávanie ponúk (pozri Obr. 6),
- ohodnotenie ponuky,
- generovanie formulára na zadanie trvalého dopytu,
- vytvorenie trvalého dopytu,
- zrušenie trvalého dopytu,
- zobrazenie zdroja ponuky.

Generovanie výsledných webových stránok sa vykonáva zostavením z stránky viacerých fragmentov. Okrem fragmentov generovaných pomocou dátovodov definujeme aj jednoduché fragmenty, ktoré sa generujú pomocou tried, alebo priamo pomocou značkovacieho jazyka. Je možné tiež použitie dodatočných fragmentov, ktoré sa často vyskytujú v portálových systémoch a sú už implementované v použitých rámcových systémoch.



Obr. 6 Dátovod na vyhľadávanie ponúk

Spôsob usporiadania jednotlivých fragmentov do stránok sa definuje pomocou konfiguračných súborov. V nich je možné vytvoriť alternatívy pre jednotlivých používateľov. Nasleduje zoznam možných fragmentov:

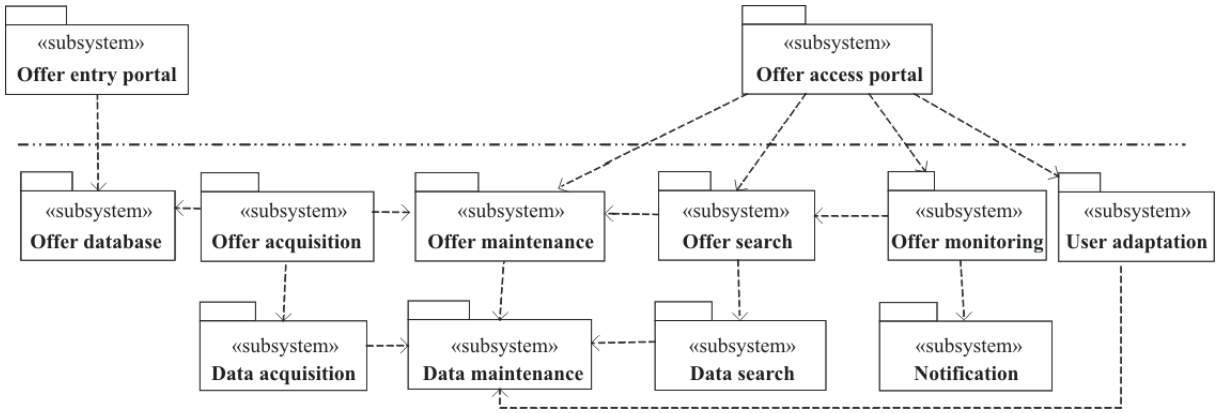
- prihlasovanie,
- menu,
- hlavička a päta stránky.

Portál chceme implementovať s využitím implementácie kontajnera portletov Jetspeed na zostavovanie výsledných stránok. Jetspeed aj Cocoon sú už implementované vo forme servletov, ktoré sa nasadia do servlet kontajnera Tomcat. Pre statický obsah (obrázky, statické webové stránky) chceme použiť HTTP server Apache namiesto servlet kontajnera.

5.1.2. Aplikačná vrstva

Navrhnutá aplikačná vrstva používa komponentový model a je založená na návrh, ktorý bol vykonaný v rámci štátneho programu NAZOU [14]. Navrhnuté komponenty sa podľa konvencií aplikačnej vrstvy J2EE delia na objekty aplikačnej logiky a aplikačné kontrolery. Identifikované triedy objektov aplikačnej logiky vychádzajú z analýzy domény získavania a poskytovania všeobecných ponúk.

Identifikovali sme tri aplikačné kontrolery, z ktorých prvý mapuje objektový model aplikačnej vrstvy na relačný model v relačnej databáze a druhý mapuje objektový model na model RDF v RDF databáze. Tretí kontroler sa stará o automatické rozposielanie ponúk v stanovených intervaloch pre trvalé dopyty používateľov. Na mapovanie objektového modelu na RDF model systém využíva vlastné komponenty, na rozdiel od mapovania medzi objektovým a relačným modelom, kde je možné využiť štandardné mapovacie nástroje. Jednotlivé komponenty portálu na aplikačnej vrstve chceme implementovať ako komponenty EJB (Enterprise Java Bean) a nasadiť ich na aplikačnom serveri JBoss.



Obr. 5

Na obrázku č. 5 je znázornený diagram podsystémov celého systému. Navrhnuté podsystémy majú logickú súdržnosť, preto sú znázornené vo forme balíkov. Spodná vrstva je nezávislá od konkrétnej aplikácie v poskytovaných službách a ich prezentácii. Vrchná vrstva modelu znázorňuje funkcionalitu špecifickú pre konkrétnu aplikáciu.

5.1.3. Prehľad podsystémov

Offer entry portal

Podsystém *Offer entry portal* poskytuje funkčnosť pre ručné zadávanie ponúk producentom. Cez tento podsystém sa používateľ – producent ponúk, stáva aktívnym používateľom celého systému. Tento podsystém prezentuje používateľovi prístupnú časť manažmentu ponúk v podsystéme *Offer database*, v ktorej producent zadáva nové ponuky, edituje („svoje“) existujúce ponuky, prípadne odstraňuje už neaktuálne ponuky zo systému.

Offer database

Podsystém *Offer database* realizuje uloženie vybraných údajov o zadaných ponukách z podsystému *Offer entry portal*. Tento podsystém umožňuje uchovávanie a spravovanie ponúk zadaných producentom ponúk priamo cez formuláre aplikačného portálu a je priamo využívaný podsystémom *Offer acquisition*, ktorý má potrebné znalosti o štruktúre dát. *Offer database* poskytuje rozhranie zdroja ponúk.

Offer acquisition

Podsystém *Offer acquisition* pristupuje k zoznamu zdrojov (webové stránky, elektronická pošta, diskusné príspevky, zdroje ponúk atď.), z ktorých získava dokumenty a z dokumentov následne extrahuje informácie o ponukách do vnútorných úložísk systému. Tento podsystém využíva na získavanie a identifikáciu ponúk v dokumentoch služby podsystému *Data acquisition*.

Data acquisition

Podsystém *Data acquisition* zabezpečuje získanie a ukladanie dokumentov, identifikáciu dokumentov potenciálne obsahujúcich ponuky na základe pravidiel vytvorených v doménovo špecifickej vrstve, a transformáciu dokumentov v procese anotácie.

Offer maintenance

Podsystém *Offer maintenance* zabezpečuje sprostredkovanie uchovaných ponúk pre ostatné podsystémy. Zahŕňa overovanie a udržovanie metadát o ponukách i samotných ponúk, vrátane kontroly aktuálnosti uloženej ponuky.

Data maintenance

Podsystém *Data maintenance* realizuje základnú správu dát v rôznych typoch dátových úložísk a tieto dáta poskytuje podsystému *Offer maintenance*. Podsystém ponúka rozhrania pre správu týchto úložísk v doménovo nezávislom tvare.

Offer search

Podsystém *Offer search* je určený na vyhľadávanie v lokálne uložených ponukách z rôznych zdrojov (web, mail a news). Umožňuje vyhľadávať ponuky na základe rôznych algoritmov a takto nájdené a pripravené ponuky sú potom použité ďalšími podsystémami pre získanie ponúk.

Data search

Podsystém *Data search* umožňuje vyhľadávanie s použitím rôznych metód na rôznej úrovni napr. slov, viet, zhlukov.

Offer monitoring

Podsystém *Offer monitoring* je určený pre pravidelné vyhľadávanie a monitorovanie novovzniknutých alebo zmenených ponúk. Podsystém sa používa na základe explicitného určenia používateľom portálu, ktorý definuje vyhľadávacie kritériá, podľa ktorých sú ponuky monitorované. O ponukách, ktoré spĺňajú špecifikované kritériá, je používateľ informovaný podsystémom *Notification*.

User adaptation

Podsystém *User adaptation* slúži na možnosť zmeny formy prezentácie a čiastočné ovplyvnenie logiky v závislosti od práve prihláseného používateľa. Podsystém využíva údaje o používateľovi z modelu používateľa tak, aby dokázal formovať prezentáciu do žiadaného tvaru podľa aktuálne prihláseného používateľa. Informácie o používateľovi z modelu používateľa môžu byť použité i pri formovaní.

Notification

Podsystém *Notification* realizuje oboznamovanie registrovaných používateľov s aktuálnymi zmenami v systéme (vznik relevantnej ponuky, zmena ponuky na relevantnú atď.). Samotný proces notifikácie môže prebiehať rôznym spôsobom (napr. formou správy elektronickej pošty alebo správy SMS).

Offer access portal

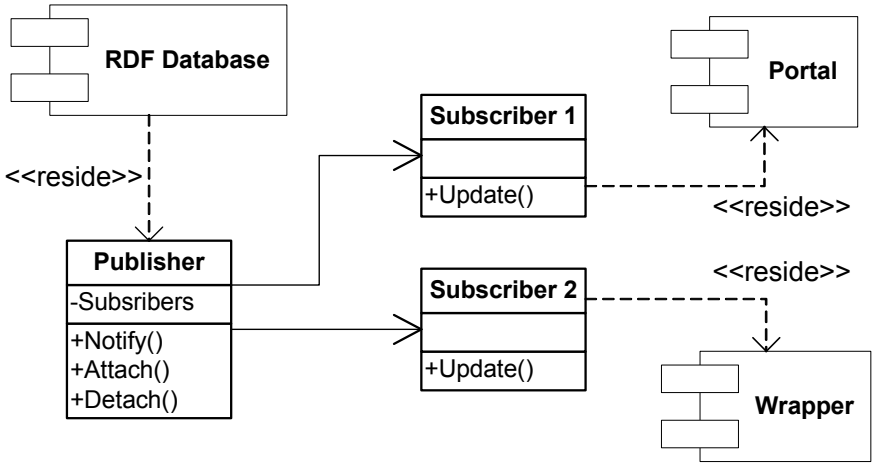
Podsystém *Offer access portal* je určený pre aktívny vstup používateľa, konzumenta, do celého systému. Ak neberieme do úvahy podsystém *Offer entry portal*, ktorý môže byť úplne samostatným systémom, ide o jediný podsystém realizujúci komunikáciu s používateľom. Podsystém *Offer access portal* využíva podsystém *User adaptation* pre ovplyvnenie, resp. prispôsobenie zobrazovanej prezentácie.

5.2. Notifikácia o zmenách v úložisku

Kapitola opisuje navrhnutý mechanizmus notifikácie zmien v ontologickom úložisku. Motiváciou pre zakomponovanie tohto mechanizmu je nutnosť sprístupniť ontologické úložisko aj pre systémy z iných projektov – ako napr. projekt WRAPPER, ktorý sa zaoberá získavaním pracovných ponúk z externých systémov a ich ukladaním do ontologického úložiska. Pri vykonaní zmeny v úložisku jedným systémom potrebujeme notifikovať aj ostatné systémy, ktoré pracujú s daným úložiskom.

V záujme využiť zdokumentované najlepšie postupy v tejto oblasti sme sa rozhodli prevziať do architektúry integračný návrhový vzor *Publish-Subscribe Channels* podľa [6]. Tento vzor je takou implementáciou vzoru *Observer*, ktorá umožňuje jeho jednoduchšie použitie v prostredí distribuovaných systémov.

Architektúra mechanizmu notifikácie (pozri Obr. 7) je založená na dvoch triedach: *Publisher* a *Subscriber*. Existuje iba jedna inštancia triedy Publisher a tá je umiestnená v blízkosti ontologiského úložiska. Vždy, keď sa vykoná zmena v úložisku sa zavolá metóda Publisher.Notify(). Publisher si udržuje zoznam inštancií triedy Subscriber, ktoré chcú byť notifikované o danom type zmeny v úložisku. Inštancie triedy Subscriber sa budú nachádzať v jednotlivých systémoch, ktoré pracujú s ontologickým úložiskom. Komunikácia medzi inštanciami tried Publisher a Subsriber bude prebiehať formou posielania správ cez vytvorený kanál, ktorý je zodpovedný za doručenie týchto správ, a ktorý bude implementovaný pomocou Java Message Service (JMS).



Obr. 7 Mechanizmus notifikácie o zmenách v úložisku

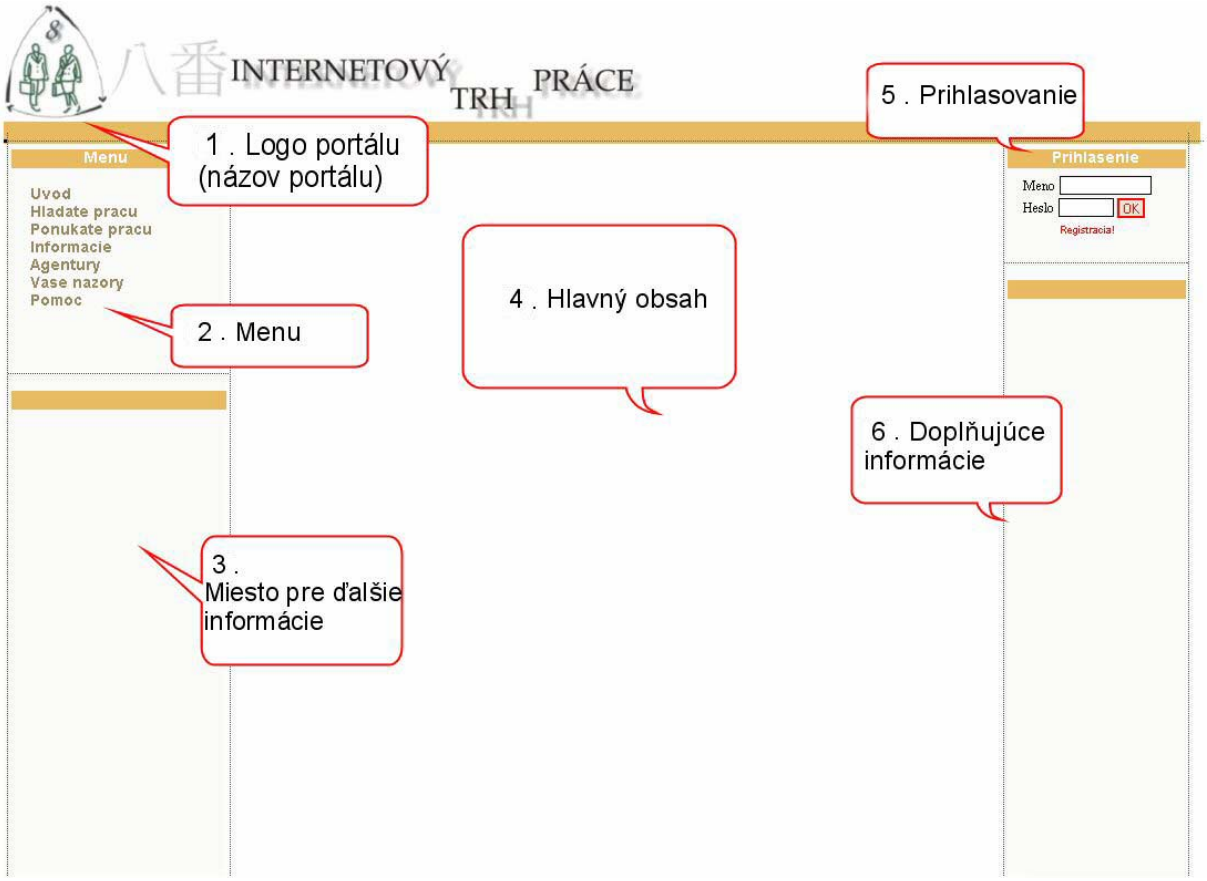
5.3. Návrh webu portálu

Dôležitou časťou riešenia portálu je grafické rozhranie, prostredníctvom ktorého používateľ pristupuje k jednotlivým funkciám portálu. Portál je zo svojej podstaty webová aplikácia a preto celá interakcia s používateľom prebieha práve cez webové rozhranie. Z tohto dôvodu je návrh webu portálu veľmi dôležitý z hľadiska jeho budúcej použiteľnosti. Na návrh webu portálu možno nazerať z dvoch uhlov pohľadu. Grafický návrh sa zaoberá tým, ako bude samotný portál vyzerať, ako budú rozložené jednotlivé prvky, aké farby a štýly sa použijú. Technologický návrh sa zaoberá skôr spôsobmi implementácie portálu a technickou realizáciou jednotlivých prvkov grafického rozhrania.

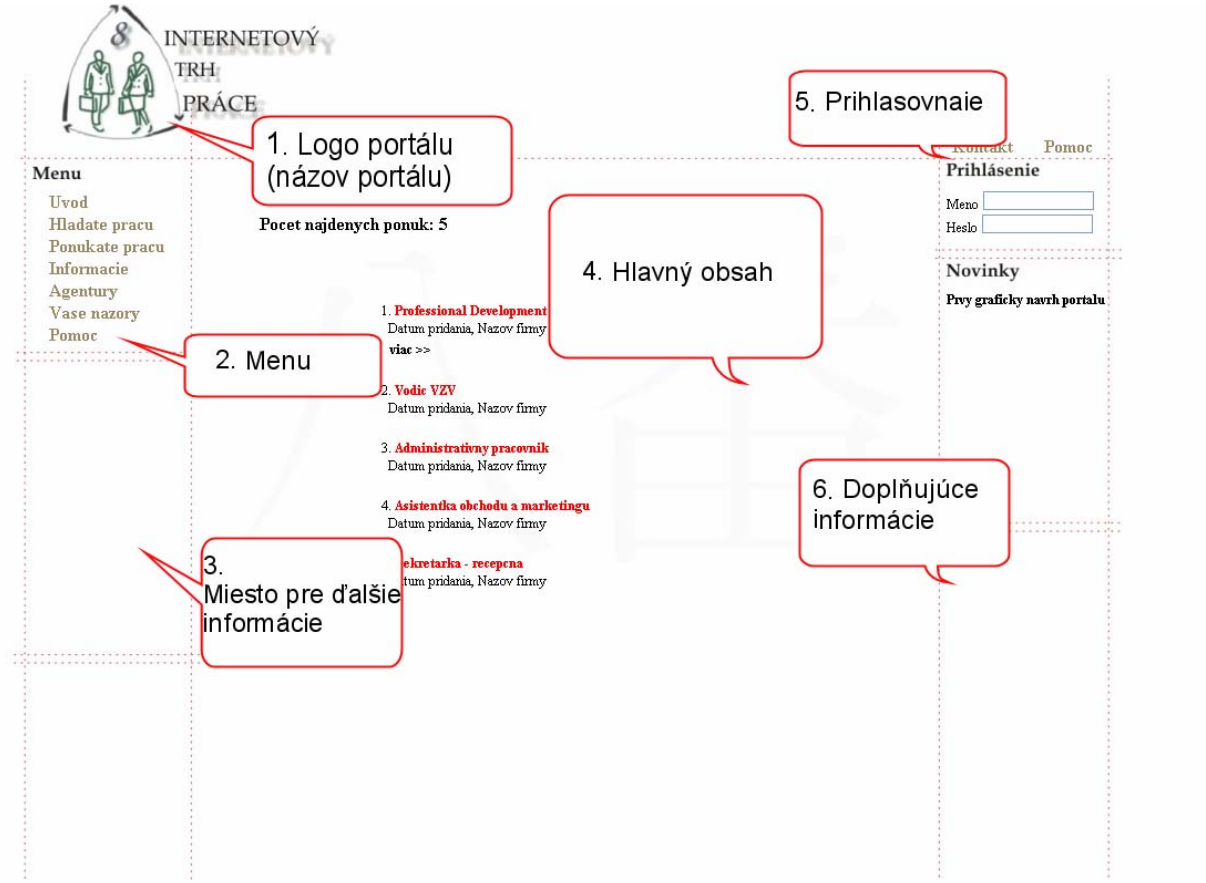
5.3.1. Grafický návrh webu portálu

Vytvorili sme dve verzie grafického návrhu webu portálu, ktoré sa líšia len v použitých grafických prvkoch. Prvý návrh je viac zameraný na grafiku (pozri Obr. 8), druhý sa vyznačuje skôr menším zastúpením grafických prvkov a farieb (pozri Obr. 9). Oba návrhy sú zatiaľ len predbežné a je predpoklad, že sa ešte bude meniť počet a rozloženie jednotlivých prvkov. Rozloženie jednotlivých prvkov v oboch návrhoch je totožné:

- V hornej časti portálu sa nachádza logo a názov portálu (1).
- Na ľavej strane sa nachádza hlavné menu (2), ktoré zabezpečuje hlavnú navigáciu používateľa po portáli a preto je potrebné na jeho zostavenie klásť veľký dôraz
- Pod hlavným menu je priestor pre dodatočné informácie (3). Typickým príkladom využitia je zobrazenie noviniek alebo dôležitých zmien na portáli. Pre podobné účely je určené aj miesto na pravej strane (6).
- V strede obrazovky sa nachádza hlavná časť stránky, slúžiaca na komunikáciu s používateľom a prezentáciu informácií (4).
- Oblasť pre prihlásenie sa pre registrovaných používateľov sa nachádza v pravom hornom rohu (5).



Obr. 8 Návrh s väčším zameraním na grafické prvky.



Obr. 9 Návrh s menším zastúpením grafických prvkov.

5.3.2. Technologický návrh

Z technického hľadiska zohľadňujeme pri návrhu webu portálu skutočnosť, že rozličný používatelia používajú pri prehliadané webu rôzne rozlíšenia. Aby sme prispôsobili návrh portálu tejto skutočnosti, vytvorili sme grafický dizajn stránky pomocou technológie kaskádových štýlov. Namiesto využitia statických hodnôt sme použili relatívne hodnoty, pričom používateľské rozhranie portálu sme prispôsobili rozlíšeniam 800x600 pixlov a vyšším.

Ďalším aspektom webu je prehľadnosť a jednoduchá navigácia medzi stránkami. Tieto vlastnosti chceme dosiahnuť vhodným použitím ďalších technológií ako je napr. Javascript, avšak s ohľadom na zásady tvorby stránok vhodných aj pre nevidiacich a slabozrakých (pozri 5.3.3).

Z technologického hľadiska je tiež dôležité využitie existujúcich štandardov a dobrých zásad pri tvorbe stránok. Jednotlivé stránky chceme vytvárať s využitím CSS tak, aby spĺňali štandard XHTML 1.0 a aby sa zobrazovali správne v najčastejšie používaných prehliadačoch (Internet Explorer, Firefox, Mozilla, Opera). Pre účely zobrazenia na mobilných zariadeniach tiež zvažujeme podporu WML.

5.3.3. Návrh sprístupňovania webu pre ľudí so špecifickými potrebami

Z analýzy prispôsobovania webu používateľom so špecifickými potrebami (pozri 2.3.2) vyplynula potreba venovať dodatočné úsilie na vhodný návrh webu portálu tak, aby bol prístupný aj tejto skupine používateľov.

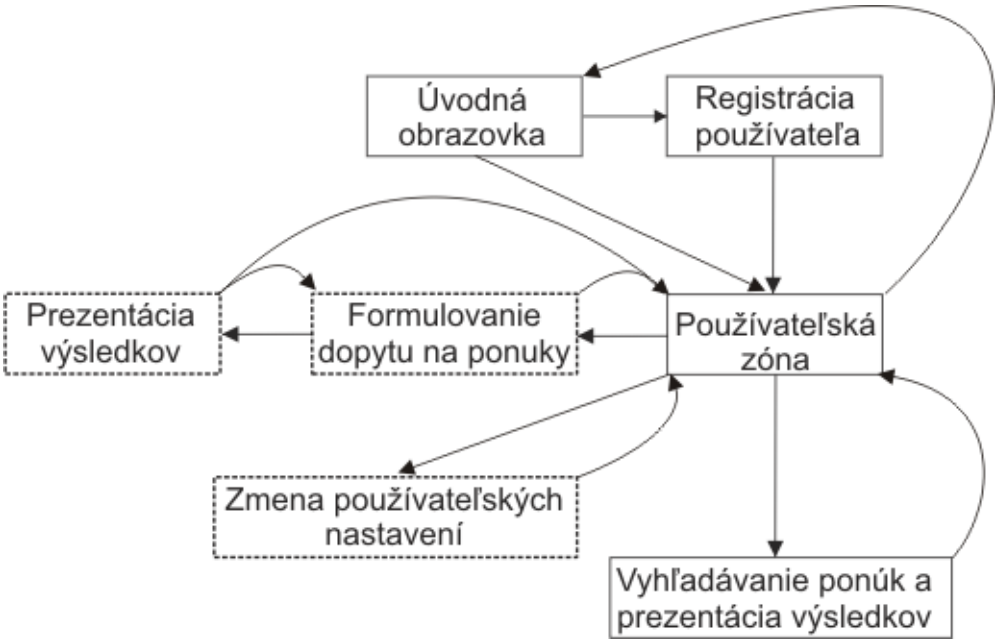
Na prvom mieste nášho záujmu boli nevidiaci používatelia, resp. používatelia s ťažkým zrakovým postihnutím, ktorí sú najviac obmedzení pri práci s portálom. Pridanou hodnotou dodržania pravidiel pre tvorbu portálu pre nevidiacich je aj zlepšenie prístupnosti portálu pre používateľov s inými obmedzeniami [15]. Príloha A obsahuje podrobný opis pravidiel pre tvorbu portálu pre nevidiacich, pričom hlavné z nich sú:

1. Pre všetky obrázky a grafické objekty musí byť definovaná hodnota alt atribútu.
2. Všetky časti stránky musia byť prístupné nezávisle na použitej technológii..
3. Vyhýbanie sa použitiu tabuliek a pravidlá ich použitia.
4. Každý odkaz musí opisovať svoj cieľ bez okolitého kontextu.
5. Stránka by mala byť prístupná aj pri rôznych farebných schémach.
6. Farby písma a podklad musia byť dostatočne kontrastné.
7. Hodnoty atribútov a štýlov musia byť zadané pomocou relatívnych jednotiek.
8. Nepoužívať efekty textu.
9. Hlavný obsah stránky je na jej začiatku.

5.4. Navigačný model portálu

Navigačný model portálu je postupnosť obrazoviek, zachytávajúca štruktúru navigácie po portáli. Pri návrhu navigačného modelu sme vychádza zo špecifikovaných prípadov použitia. Návrhy jednotlivých stránok sú predbežné a budú sa ešte meniť. V jednotlivých obrázkoch sú červeným rámikom označené dôležité časti konkrétnu stránku.

Globálny pohľad na navigačnú štruktúru portálu je na Obr. 10, pričom stránky zobrazené prerušovanou čiarou sa nachádzajú v štádiu rozpracovania a nemajú vytvorený grafický návrh.



Obr. 10 Navigačná štruktúra portálu

5.4.1. Registrácia nového používateľa

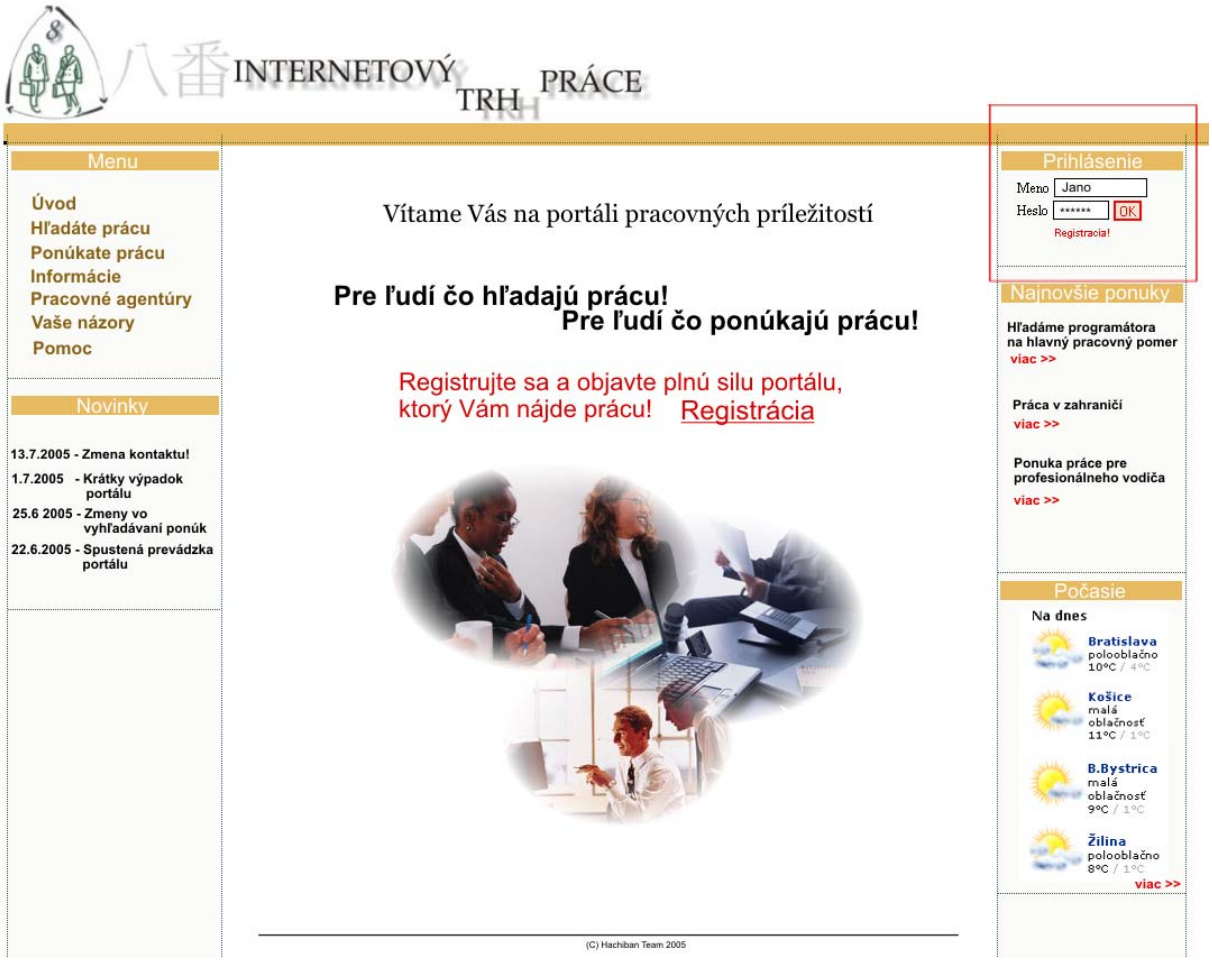
Registrácia používateľa je dôležitá z hľadiska prispôsobovania sa portálu potrebám používateľa. Registrácie je dostupná z hlavnej stránky portálu a zároveň návrh predpokladá aj stránku, na ktorej budú uvedené výhody registrácie. Aby sme však neodradili mnoho používateľov od používania portálu, nebude registrácia nevyhnutná.

Obr. 11 Registrácia nového používateľa.

Samotná registrácia bude vyžadovať iba minimum informácii od používateľa – používateľské meno a heslo (pozri Obr. 11). Všetky ostatné údaje (napr. email, adresa) budú predbežné nepovinné, aj keď ich vyplnenie je z hľadiska budúceho kontaktu s používateľom a prispôsobovania sa portálu žiaduce. Výsledkom registrácie bude buď obrazovka potvrdzujúca registráciu alebo obrazovka s oznámením o chybe.

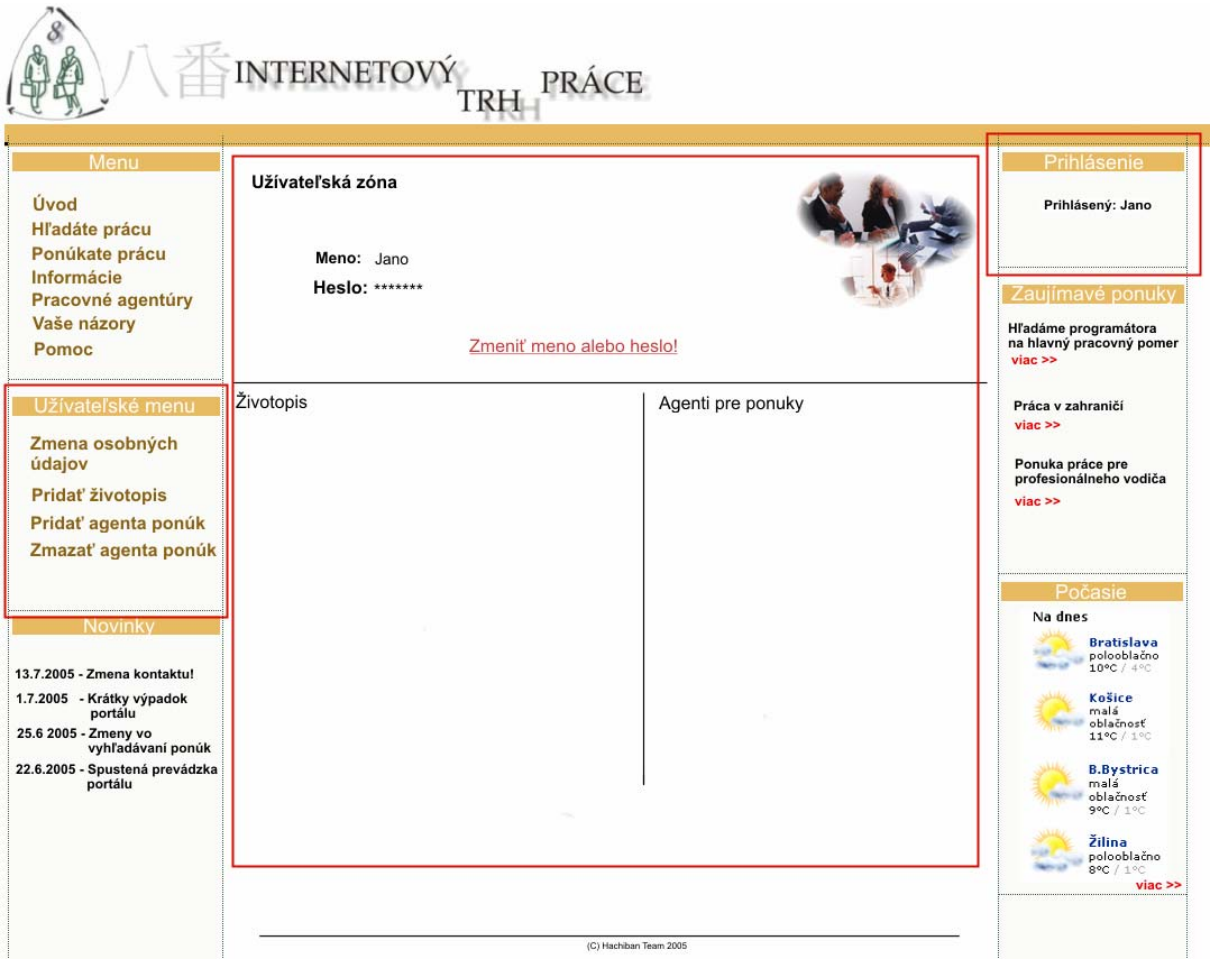
5.4.2. Úvodná obrazovka portálu

Prihlasovacie údaje používateľ vkladá v pravej hornej časti obrazovky (pozri Obr. 12). Zároveň sa tu nachádza aj odkaz na registráciu používateľa. Na viacerých miestach portálu budú umiestnená propagácia a vysvetlenie výhod registrácie, ktorá je nevyhnutná na prispôsobovanie sa portálu.



Obr. 12 Úvodná obrazovka portálu - verejná zóna.

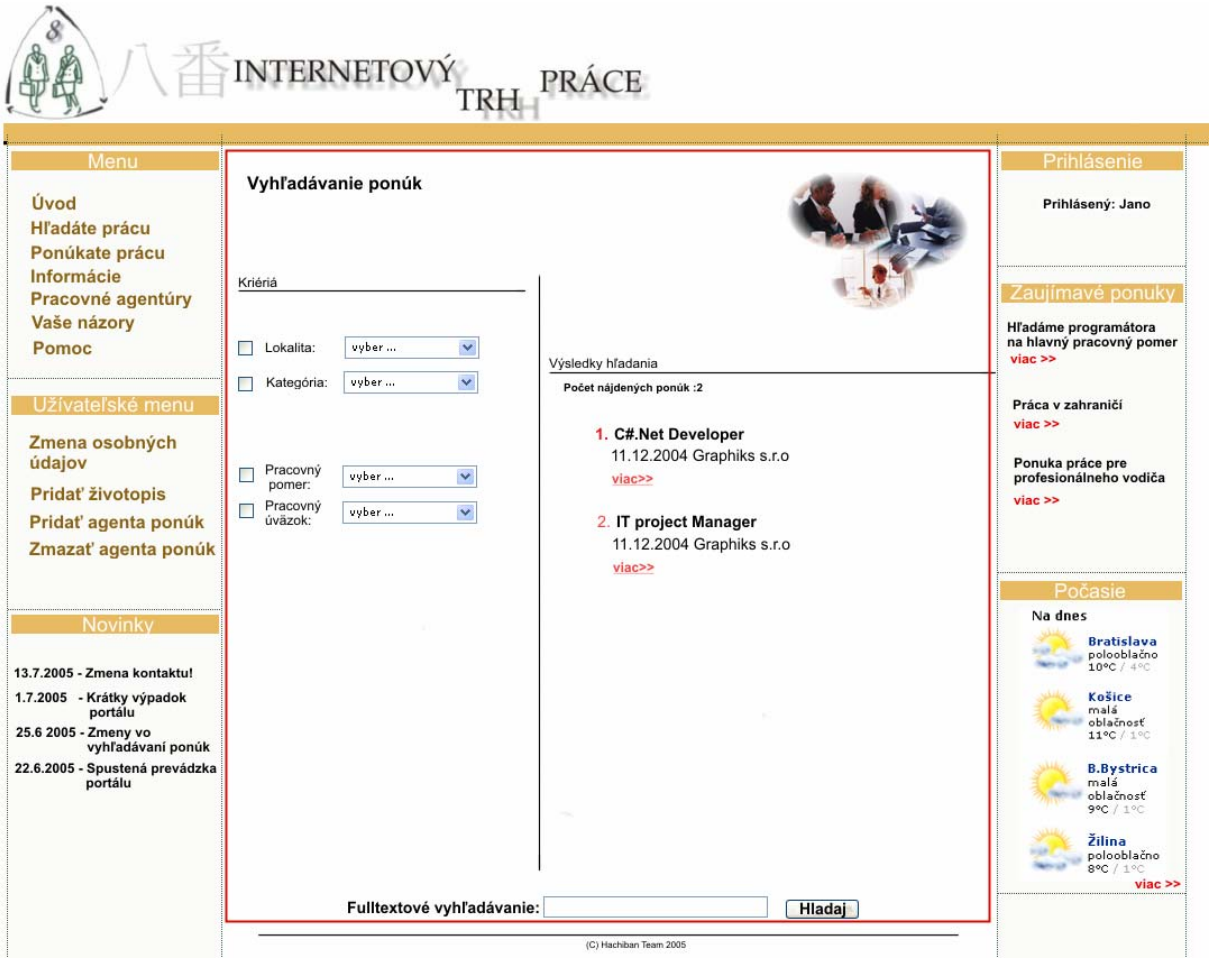
Po prihlásení sa, má používateľ prístup k vlastným nastaveniam a k svojmu životopisu (pozri Obr. 13). Súčasne s prihlásením sa portál začne prispôsobovať používateľovi na základe údajov o používateľovi, ktoré sú uložené v modeli používateľa.



Obr. 13 Úvodná obrazovka portálu - privátna zóna.

5.4.3. Formulovanie dopytu na ponuky

Dôležitou časťou portálu je formulovanie dopytu na ponuky používateľom a následná prezentácia nájdených ponúk. Predbežný návrh obrazovky pre rýchle vyhľadávanie ponúk je na Obr. 14.



Obr. 14 Zadávanie kritérií vyhľadávania a prezeranie nájdených ponúk.

5.5. Zhodnotenie analýzy nástrojov

Vo fáze analýzy sme sa venovali zisťovaniu vlastností viacerých alternatívnych nástrojov, ktorý je možné použiť pri tvorbe portálových riešení, aby sme si mohli vybrať to najvhodnejšie z nich. Vzhľadom na nástroje použité v štátnom programe NAZOU a zistené vlastnosti nástrojov a technológií z analýzy v kapitole 3 sme sa rozhodli použiť nasledovné nástroje, resp. technológie:

- Prostredie Linux
- Implementačný jazyk Java verzie 5
- Sesame – úložisko ontologických dát
- PostreSQL alebo MySQL – úložisko relačných dát
- Subversion – správa zdrojových súborov a dokumentácie
- Apache Cocoon – prezentačný rámec

- JBoss – aplikačný rámec
- Jetspeed-1 – rámec pre tvorbu portálov
- Dreamweaver alebo NVU – grafický návrh webu portálu
- Javadoc – generovanie technickej dokumentácie
- Eclipse – integrované vývojové prostredie

5.6. Testovanie

Na zabezpečenie požadovanej kvality vytváraného produktu sme sa rozhodli použiť testovaciu metodológiu unit testing a v rámci nej open source rámec JUnit. Podľa tejto metodológie testovanie nemajú na starosti testeri ani koncoví používatelia ale samotní vývojári. Aby bolo možné v praxi unit testing efektívne využiť je potrebné dodržiavať nasledovné pravidlá:

- implementácie testovacích tried sú oddelené od zvyšku kódu
- každá testovacia funkcia je atomická a automatická
- žiadne dve testovacie funkcie nie sú na sebe závislé
- testujú sa najmä hraničné prípady
- testovaciu funkciu treba implementovať ako samostatný program (inicializácia a všetky kroky potrebné pre splnenie zámeru)
- podľa potreby je nutné napísať viacero testovacích metód pre jednu funkciu a v každej z nich sa sústrediť na niečo iné (napr. inú výnimku)

6. Prototyp

Kapitola opisuje postup prototypovania rizikových častí budúceho portálu pracovných príležitostí. Zaoberá sa analýzou rizík, ktoré môžu významnou mierou ovplyvniť tvorbu portálu. Následne opisuje samotné prototypovanie jednotlivých rizikových častí. Zachytáva naše skúsenosti s jednotlivými technológiami, spolu s problémami s ktorými sme sa stretli a ich riešeniami. Viaceré problémy sme vyriešili zmenou pôvodného návrhu, nepoužitím pôvodne zmýšľaných technológií a ich nahradením inými.

6.1. Analýza rizík

Za cieľ práce na prototype sme určili *overenie architektúry* navrhnutého systému a *overenie použiteľnosti a integrovateľnosti jednotlivých čiastkových technológií*. Overenie technológií pokladáme za mimoriadne dôležité najmä preto, že sa z veľkej časti jedná o FOSS riešenia, ktoré nie sú primerane zdokumentované. Navyše s väčšinou z uvedených technológií máme len málo skúseností, resp. sú pre nás úplne nové.

Na základe uvedených skutočností sme si stanovili za cieľ naštudovať a sfunkčniť tie technológie, ktoré pokladáme za rizikové:

- Sesame – prístup k ontologickému úložisku.
- Aplikačný rámec JBoss pre prácu s EJB.
- Rámec pre tvorbu portálov Jetspeed.
- Rámec pre tvorbu webových aplikácií Cocoon.
- XForms – práca s formulármi.

Veľmi dôležitým aspektom prototypu je nielen overenie funkčnosti jednotlivých častí portálu ale aj ich *integrovateľnosť do jedného spolupracujúceho celku*. Z tohto dôvodu sme v prototype kládli veľký dôraz aj na integrovateľnosť a schopnosť vzájomnej spolupráce jednotlivých komponentov navrhnutej architektúry. Práve finálna integrácia komponentov totiž môže zabrať veľké množstvo času a v konečnom dôsledku by nemusela byť ani úspešná, čo by v prípade neexistencie „záložného plánu“ mohlo viesť až k neúspechu celého projektu.

6.2. Enterprise Java Beans

Pôvodný návrh počítal s využitím technológie EJB (Enterprise Java Bean) pri implementácii aplikačnej logiky portálu. Súčasťou prototypovania bolo aj overenie náročnosti práce s touto technológiou. Na základe analýzy sme na jej implementáciu vybrali aplikačný server JBoss.

Inštalácia prebehla bez závažnejších problémov, avšak pri pokuse o spustenie ukážkového EJB podľa návodu na jeho vytvorenie vzniklo veľa problémov. Server JBoss totiž vracal chybu pri pokuse o zavedenie skúšobného EJB. Podľa jej opisu nevedel nájsť skupinu knižníc, ktoré boli jeho súčasťou.

Pretože využitie tejto technológie v projekte nie je nevyhnutné, rozhodli sme sa jej nevenovať príliš veľa času a predbežne ju nepoužiť. Ponechávame si však otvorenú možnosť sa k nej vrátiť v prípade, že všetky dôležitejšie úlohy budú vyriešené.

6.3. Prístup k ontologickému úložisku Sesame

Primárnymi dátami, s ktorými bude pracovať portál pracovných príležitostí sú dáta pracovnej ponuky. Z tohto dôvodu sme prototypovali prístup k ontologickému úložisku Sesame na

príklade niekoľkých tried a ich atribútov. Na uchovávanie dát v pamäti používame objekt Java Bean (trieda v Jave, s atribútmi, v ktorých sa uchovávajú želané dáta a dvojicami get a set metód, ktoré umožňujú nastaviť a získať hodnoty atribútov). Použitie Java Bean prináša nasledovné výhody:

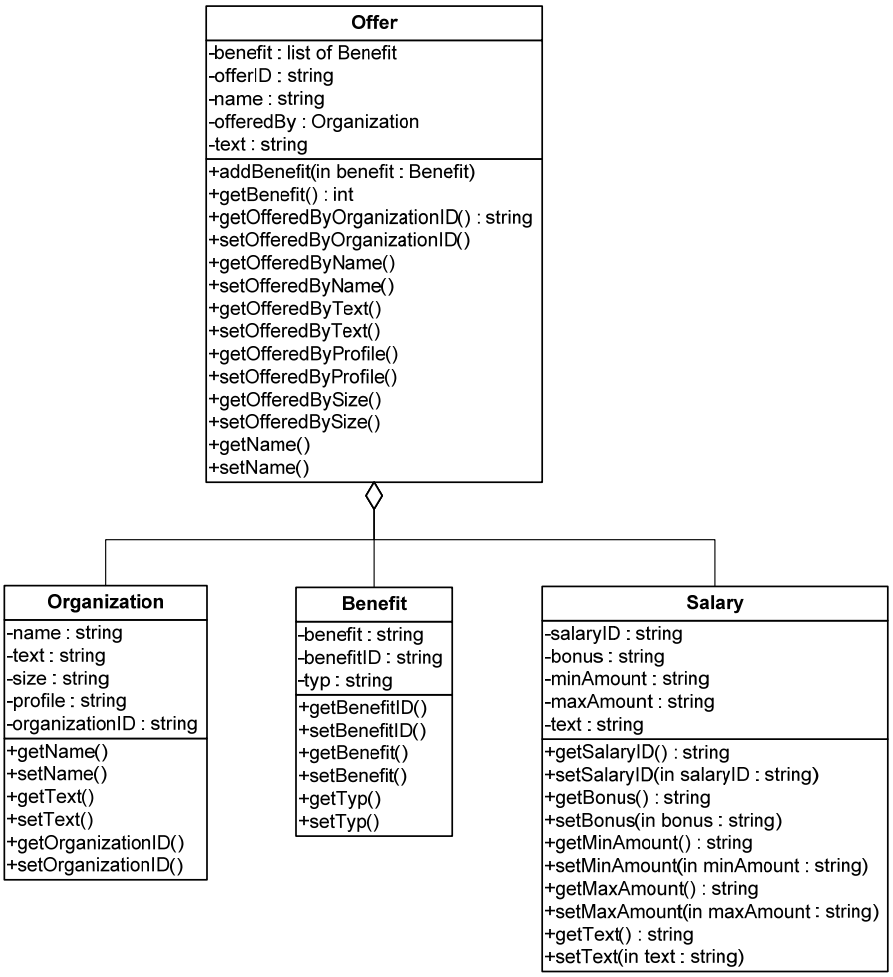
- Oddelenie dát od funkcionality.
- Integráciu s rámcom Cocoon.

V prototype používame ručne vytvorenú triedu Java Bean. Neskôr však predpokladáme jej dynamické generovanie, čo bude mať za následok flexibilnejšie prispôsobovanie sa zmenám v ontológii, prípadne zmene doménovej oblasti.

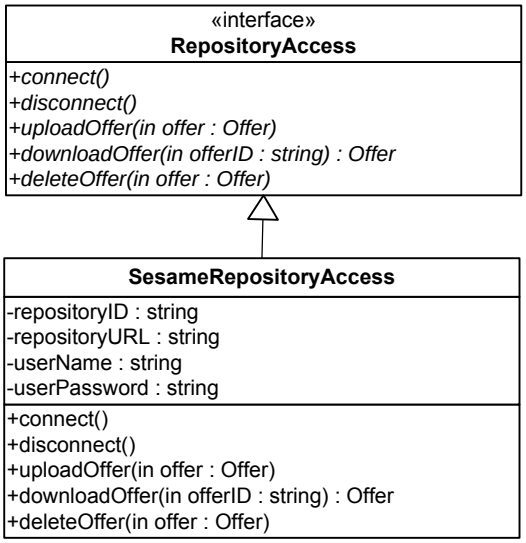
Prototyp obsahuje Java Bean triedu *Offer* v balíku *nazou.support.jobofferportal.**model***. Táto trieda agreguje niekoľko ďalších tried, ktoré súhrne obsahujú informácie pre pracovnú ponuku. Každá trieda obsahuje práve jednu logicky ucelenú časť informácií (pozri Obr. 15). Trieda *Offer* implementuje get a set metódy pre atribúty niektorých agregovaných tried. Týmto spôsobom sa stáva použitie viacerých tried transparentným. V kontexte portálu pracovných príležitostí vykonávame nad úložiskom nasledovné operácie:

- Pridanie ponuky
- Zmena ponuky
- Vymazanie ponuky

Pre tieto potreby sme vytvorili balík Java tried *nazou.support.jobofferportal.**repository***. Obsahuje triedy poskytujúce API pre realizovanie operácií nad úložiskom a rozhranie *RepositoryAccess*, ktoré implementujú všetky triedy pracujúce s konkrétnym úložiskom. Rozhranie obsahuje funkcie na pripojenie k úložisku, pridanie, získavanie a odobratie dát. Pretože v súčasnosti predpokladáme len prístup k úložisku Sesame, existuje zatiaľ len jedna trieda implementujúca toto rozhranie – trieda *SesameRepositoryAccess* (pozri Obr. 16). V budúcnosti predpokladáme dynamické generovanie objektu Java Bean ponuky, čo bude mať vplyv aj na ukladanie a získavanie dát ponuky do/z úložiska.



Obr. 15 Diagram tried balíka *nazou.support.jobofferportal.model*.



Obr. 16 Diagram tried balíka *nazou.support.jobofferportal.repository*.

6.3.1. Rámec pre tvorbu portálov Jetspeed

Podľa návrhu portálu bude úlohu prezentovať informácie zabezpečovať rámec Jetspeed[1]. Cieľom prototypu bolo veriť jeho funkcionalitu a náročnosť jeho využitia v podmienkach portálu pracovných príležitostí.

Základnou úlohou bolo naučiť sa konfigurovať Jetspeed a zoznámiť sa s ním na dostatočne podrobnej úrovni. Ďalšou úlohou bolo vytvorenie základu budúceho portálu v Jetspeede, pričom v rámci prototypu sme vytvorili len prvý návrh vzhľadu portálu.

Samotný Jetspeed bol pre nás novou technológiou, v dôsledku čoho sme pri jeho sfunkčnení narazili na množstvo problémov a komplikácií. Úvodná práca s Jetspeedom vyžadovala najmä konfiguráciu jednotlivých jeho funkcií a vlastností.

Inštalácia Jetspeedu

Pri vytváraní nového projektu a práci s Jetspeedom je potrebný program Maven, ktorý má uľahčovať prácu a vytváranie nového portálu. Prvým problémom, s ktorým sme sa pri rámci Jetspeed stretli bola jeho inštalácia. Najnovšia verzia Jetspeedu je verzia 2.0, pre potreby projektu sme sa však rozhodli použiť verziu 1.6, pretože podľa autorov Jetspeedu je verzia 2.0 ešte nestabilná. Navyše k nej neexistuje takmer žiadna dokumentácia.

Na skompilovanie portálového riešenia Jetspeed je potrebné nainštalovať a nakonfigurovať nástroj, ktorý riadi proces zostavovania programu zo zdrojových súborov – Apache Maven. Aktuálna verzia Apache Maven je 2.0. Po úspešnej inštalácii a nakonfigurovaní sme zistili, že Maven vo verzii 2.0 odmieta spolupracovať s projektmi Jetspeed-u verzie 1.6. Následne sme prešli k verzii 1.0.2, s ktorou sme doteraz nezaznamenali žiadne problémy. Najväčšie problémy pri inštalácii a práci s Jetspeedom vôbec nám spôsoboval najmä nedostatok dokumentácie tak k Jetspeedu ako aj k Mavenu.

Tvorba nového portálu

Po nainštalovaní Jetspeedu bolo prvoradou úlohou vytvorenie nového portálu. Aj keď na prvý pohľad to vyzerá ako triviálna úloha, v novom prostredí s veľmi slabou dokumentáciou zabrala dosť veľa času.

Asi najväčší problém pri práci s Jestpeedom je skutočnosť, že okrem domovskej stránky existuje o ňom iba veľmi málo zdrojov dokumentácie a informácií. (Aj keď je všeobecne opisovaný ako silný nástroj na tvorbu portálov.) Na domovskej stránke projektu Jetspeed je k dispozícii návod k verzii 1. V niektorých častiach je však neúplný a pôsobí nedokončeným dojmom. Niektoré časti chýbajú úplne. Situáciu ešte sťažuje fakt, že na stránke sa nachádzajú aj nefunkčné odkazy – napríklad verzia 1.6 Jetspeedu sa nedala z tejto stránky ani stiahnuť.

Jetspeed obsahuje veľké množstvo konfiguračných súborov, pomocou ktorých je možné ho prispôsobovať vlastným potrebám. Položky jednotlivých súborov v novom projekte sa menia tak, že sa vytvorí nový súbor s menom ako mal pôvodný s koncovkou merge. Do tohto súboru sa potom dajú všetky vlastnosti, ktoré je potrebné zmeniť. Pri vytváraní nového portálu sa tieto vlastnosti zmenia a všetky ostatné sa nechajú na pôvodných hodnotách.

Grafická časť portálu

Prostredie Jetspeedu rieši grafickú časť portálu pomocou šablón, pričom je možné si vybrať medzi dvoma druhmi šablón, z ktorých sme pre náš prototyp vybrali šablóny Velocity. V projekte je možné definovať základné šablóny pre hornú, ľavú a spodnú časť portálu.

[1] Jetspeed, http://portals.apache.org

Samotné vytvorenie šablón bola dosť jednoduchá úloha, ktorá spočívala iba v skopírovaní už existujúceho kódu z pôvodného hrubého návrhu do Jetspeedu.

Správa používateľov

Jetspeed obsahuje už vytvorený spôsob autorizácie používateľov. Má zabudovaného používateľa – admin, ktorý slúži na správu portálu. Ďalej existuje anonymný užívateľ, za ktorého sa považuje každý neprihlásený návštevník portálu.

Admin môže upravovať jednotlivé účty používateľov a takisto prihlásený používateľ si môže podľa svojej potreby upravovať portál. Pre potreby prototypu bol vytvorený jeden testovací používateľ, na ktorom je prezentovaná funkčnosť riešenia.

Portlety

Najdôležitejším prvkom Jetspeedu sú portlety, ktoré robia z Jetspeedu silný nástroj a umožňujú vytvárať portály, ktoré si môže každý užívateľ prispôsobiť podľa vlastných požiadaviek.

V Jetspeede existuje veľké množstvo portletov, ktoré sa dajú po jednoduchej konfigurácii hneď použiť. Samozrejme je možné vytvárať aj vlastné portlety. Do protoypu sme zahrnuli iba portlety, ktoré boli v Jetspeede už vytvorené a nevytvárali sme vlastné portlety. V časti prototypu sme teda pracovali s portletmi iba na úrovni konfigurácie a ich rozmiestňovania po portáli.

Počas prototypovania nastali pri konfigurovaní portletov komplikácie. Každý portlet v Jetspeede je reprezentovaný časťou portálu, ktorá môže byť ohraničená napríklad rámčekom. Pre každý portlet je možné sprístupniť používateľovi funkcie ako napr. minimalizácia portletu, maximalizácia. Pre každý portlet je možné nastaviť, komu sa daný portlet bude zobrazovať a komu nie. Problém spočíval v tom, že anonymnému používateľovi sa nezobrazovali tlačidlá na editáciu portletov (maximalizácia, minimalizácia a pod.). Tento problém sa nakoniec podarilo vyriešiť prostredníctvom konfiguračného súboru, avšak zabralo to značné množstvo času kvôli nedostatku príslušnej dokumentácie.

6.3.2. Spolupráca rámcov Cocoon a Jetspeed

Po sfunkčnení rámca Jetspeed sme overili možnosť jeho spolupráce s prezentačným serverom Cocoon. Základná myšlienka spočívala v zobrazení výstupu z rámca Cocoon v portletoch rámca Jetspeed, pričom obe tieto riešenia sú vyvíjané pod záštitou Apache Software Foundation[1].

Inštalácia prezentačného servera Cocoon

Cocoon v najnovšej verzii 2.1.8 je podľa jeho tvorcov stabilné a spoľahlivé riešenie, aj keď niektoré jeho časti sú pod stálym vývojom. Pri jeho inštalácii sme narazili na jeden drobný problém, spôsobený medzerou v ceste k adresáru v ktorom bol kompilovaný. V zostavovacích skriptoch sa totiž vyskytuje chyba, znemožňujúca zostavovanie v adresári, ak sa v ceste vyskytuje medzera.

Cocoon a Jetspeed

Zabalenie prezentačného servera Cocoon do portálového riešenia Jetspeed si vyžiadalo veľa experimentovania. Znova sme sa stretli s nedostatkom relevantnej dokumentácia. Zistili sme, že existuje špeciálny portlet na komunikáciu s Cocoonom. Po hlbšom skúmaní sme však došli

[1] Apache Software Foundation, http://www.apache.org

k poznaniu, že tento portlet nie je podporovaný portálovým riešením Jetspeed od verzie 1.3. Verzia podporujúca tento portlet je až Jetspeed 2.0, ktorá nie je stabilná.

Možným riešením bolo využitie portletu WebPagePortlet, ktorý je súčasťou Jetspeedu a ktorý dokáže v sebe zobraziť obsah HTML stránky zadanej pomocou URL. Tento však nepodporuje navigáciu v rámci portletu a každý odkaz otvorí v novom okne. Iným riešením bolo využiť prístup autorov z University of Indiana, ktorí rozšírili HTML Rewriter portletu WebPagePortlet o podporu navigácie. Tento prístup sa nám však nepodarilo aplikovať kvôli rozdielu vo verziách Jetspeedu (tento prístup bol testovaný na verzii 1.3).

Riešenie sa našlo vo využití portletu IFramePortlet, ktorý do vnútra portletu vloží frame a umožní aj vnútroportletovú navigáciu, pričom funguje ako malý prehliadač HTML stránok. Následne celá komunikácia a volanie Cocoonu z portletov v Jetspeede prebieha v prototype pomocou volania URL adries.

6.4. Reprezentácia formulárov

Jedným z cieľov prototypovania bolo aj nájdenie vhodnej vnútornej reprezentácie formulára ponuky. Pôvodný návrh portálu predpokladal použitie existujúceho štandardu XForms konzorcia World Wide Web, neskôr sme však rozhodli namiesto tejto technológie použiť Cocoon Forms. XForms je štandard založený na XML, ktorý oddeľuje model formulára od jeho konkrétnej vizuálnej reprezentácie. Využitie technológie XForms sme prototypovali preto, lebo žiadny člen tímu doteraz nemal s touto technológiou žiadne skúsenosti a znamenala značné riziko aj kvôli svojej novosti.

6.4.1. XForms

Pri práci s XForms sme na niekoľko problémov. Na stránke konzorcia W3C sa uvádza, že štandard XForms už má dostatok funkčných implementácií, náš prieskum však ukázal, že tieto implementácie ešte nie sú využiteľné. Ani jeden z troch najrozšírenejších webových prehliadačov (Opera, Mozilla Firefox, Internet Explorer) nebol schopný korektne zobraziť formulár zapísaný pomocou XForms.

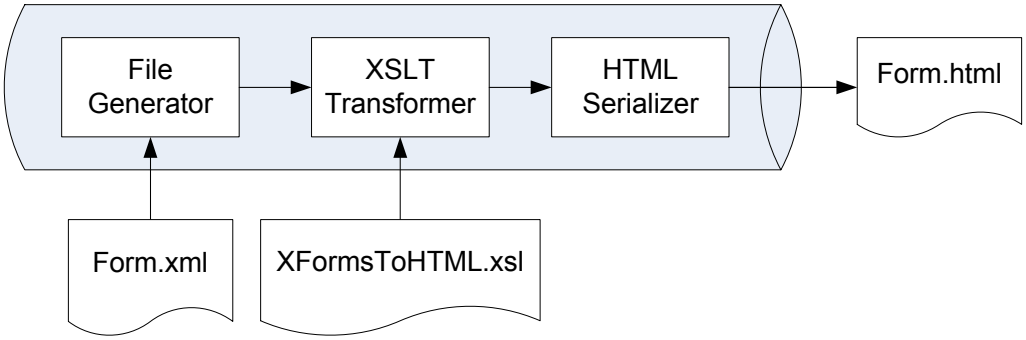
Spoločnosť Novell vyvinula zásuvný modul pre Internet Explorer (aktuálne v beta verzii), s ktorým tento prehliadač vedel korektne pracovať s XForms formulármi. Ak by sme formuláre portálu budovali pomocou XForms, zobrazovali by sa korektne iba používateľom používajúcim Internet Explorer s daným rozšírením. Toto používateľské obmedzenie je pre nás neakceptovateľné, pretože portál by sa mal korektne zobrazovať vo všetkých troch najrozšírenejších prehliadačoch bez potreby ich ďalších úprav.

Pokúsili sme sa tiež o transformáciu formulára z XForms reprezentácie do HTML reprezentácie, ktorá je korektne zobrazovaná vo všetkých prehliadačoch. Týmto spôsobom by bolo možné spracovanie formulára na strane servera pre klientov, ktorí nepodporujú XForms štandard a zároveň zasielanie XForms reprezentácie formulára klientom, ktorí túto technológiu implementujú.

Na spracovanie formulára sme potrebovali aj jeho reprezentáciu, ktorá používa pamäťové objekty, mohli sme využiť napríklad DOM (Document Object Model). Aplikačný rámec Cocoon obsahuje priamu podporu XML dokumentov. Medzi svojimi komponentmi ich prenáša pomocou volania udalostí SAX (Simple API for XML).

Overili sme, že dokážeme vnútornú reprezentáciu formulára ponuky pretransformovať do formy vhodnej na zobrazenie na koncovom zariadení – konkrétne do formulárov HTML. Na túto transformáciu existuje niekoľko nástrojov. Niektoré z voľne prístupných sme vyskúšali. Zistili sme, že ich jadrom spravidla bolo niekoľko štýlov XSL, ktoré riadili prácu štandardného XSLT transformátora. Keďže Cocoon umožňuje zaradiť do vytvoreného

dátovodu aj komponent XSL transformátora, skúsili sme na jeho riadenie použiť XSL štýly z týchto nástrojov. Vytvorili sme jednoduchý dátovod v Cocoone a skúšali pretransformovať jednoduchý dokument (pozrie Obr. 17).



Obr. 17 Jednoduchý dátovod v rámci Cocoon.

Obsahuje tri komponenty:

- generátor, ktorý číta zo súboru formulár v XForms,
- transformátor XSL a
- HTML serializátor, ktorý serializuje udalosti SAX do výstupného prúdu HTML.

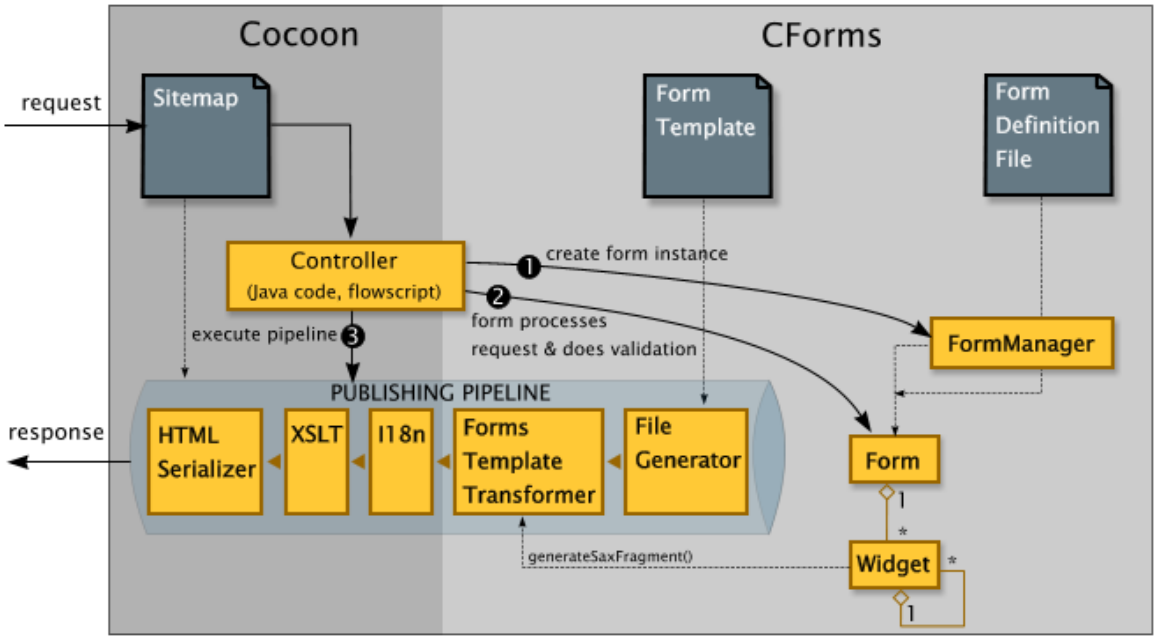
Pri niektorých vstupných dokumentoch sme zistili, že transformácia neprebehla správne. Bolo to pravdepodobne tým, že dokument v týchto prípadoch nebol úplne platný XForms dokument, prípadne preto, že použitý XSL štýl nepodporoval celú špecifikáciu XForms. Týmto postupom sme si overili použitie XSLT transformátora v Cocoone a našli taký XSL štýl, ktorý opisuje túto transformáciu a funguje v najväčšom počte prípadov.

6.4.2. CForms (Cocoon Forms)

CForms je oficiálny rámec pre prácu s formulármi pre Cocoon. Takisto ako XForms oddeľuje model dát formulára od jeho zobrazenia. Navyše je priamo integrovaný v Cocoone a poskytuje integrovanú validáciu formulárov, konverziu dát. Výhodou je aj to, že poskytuje API priamo do flowscriptu Cocoonu a obsahuje ďalší rámec, ktorý zabezpečuje previazanie údajov z formulára na objekt Java Beans alebo XML dokument.

Problém, ktorý sme identifikovali pri XForms technológií, s nesprávnym zobrazovaním formulárov v prehliadačoch je pri CForms vyriešený. Cocoon obsahuje transformácie, ktoré dokážu z formulára v CForms zápise vygenerovať klasický HTML formulár, bežne zobraziteľný v ľubovoľnom prehliadači. Pri potrebe iného výstupu ako HTML sa jednoducho zmení použitá transformácia.

Typický scenár prepojenia



Obr. 18 Prepojenie CForms s rámcom Cocoon, prevzatý z [2].

Obr. 18 zobrazuje prepojenie Cocoonu s rámcom CForms:

- V prvom kroku FormManager vytvorí na požiadavku kontroléra inštanciu formulára, ktorá je založená na modeli formulára (form definition). Tieto modely sú uchovávané vo vyrovnávacej pamäti, takže vytváranie inštancií je rýchle.
- Kontrolér potom môže predvyplniť formulár dátami z XML súboru alebo objektu Java Bean pomocou prepájacieho rámca. Pre prepojenie formulára s týmito zdrojmi dát existuje prepájací rámec.
- Volaním dátovodu sa formulár zobrazí.
- Keď je formulár vyplnený a odoslaný, kontrolér prenechá inštancii formulára spracovanie požiadavky, takže všetky prvky formulára sa dokážu nastaviť na hodnoty z požiadavky. Následne sa spustí validácia jednotlivých polí podľa modelu formulára. Ak validácia nebola úspešná, formulár sa opäť zobrazí spolu s oznamom o chybe, inak kontrolér vykoná akciu podľa aplikačnej logiky (naviazanie dát späť na objekt Java Beans alebo XML, prípadne spustenie ďalších procesov nad dátami).

6.4.3. Prototyp formulára

Vytvorili sme prototyp, ktorým sme overili nasledovné aspekty riešenia:

- Zápis definície formulára vo formáte požadovanom CForms.
- Definíciu zobrazenia formulára vo formáte požadovanom CForms.
- Implementáciu logiky aplikácie – flowscript, ktorá zobrazí formulár, uloží získané dáta do Java Bean objektu (Binding framework) a zabezpečí perzistenciu týchto dát v ontologickom úložisku.
- Konfigurácia *sitemap.xmap,* kde sú zadefinované potrebné dátovody na zobrazenie formulára, pokračovanie vykonávania flowscriptu (continuation), zobrazenie stránky podľa šablóny (JXTemplate).

6.5. Zobrazenie formulárov v rámci Cocoon

Značná časť práce s portálom pozostáva z vypĺňania formulárov. Na ich zobrazenie používame rámec Cocoon, ktorý podporuje dynamické generovanie formulárov. Hlavným dôvodom potreby dynamického generovania je skutočnosť, že formulár musí odrážať aktuálne údaje uložené v ontológii.

Pretože generovanie formulára z ontológie v RDF úložisku do vnútornej reprezentácie v portáli je časovo náročný proces (trvá minimálne niekoľko sekúnd), bude sa vykonávať iba pri vytvorení a zmene ontológie. Samotná zmena ontológie sa vykonáva mimo portálu pomocou špecializovaných nástrojov a preto je potrebné zabezpečiť notifikáciu portálu o zmene.

V prvej verzii portálu tento problém riešime pomocou administračnej časti, ktorá umožní administrátorovi v prípade potreby pregenerovať vnútornú reprezentáciu formulárov. Tá sa uloží do skupiny XML súborov. V neskoršej predpokladáme pridanie komponentu, ktorý bude sledovať RDF úložisko, a v prípade zmeny, o tom bude notifikovať portál, ktorý následne automaticky vykoná pregenerovanie formulárov.

Bude potrebné sledovať aj typ vykonanej zmeny, keďže zmena samotných ponúk by nemala vyvolať generovanie formulára. Treba tiež zabezpečiť, aby sa počas generovania nevyskytla chvíľa, v ktorej už nie je dostupná predchádzajúca podoba formulára a ani jeho nová reprezentácia.

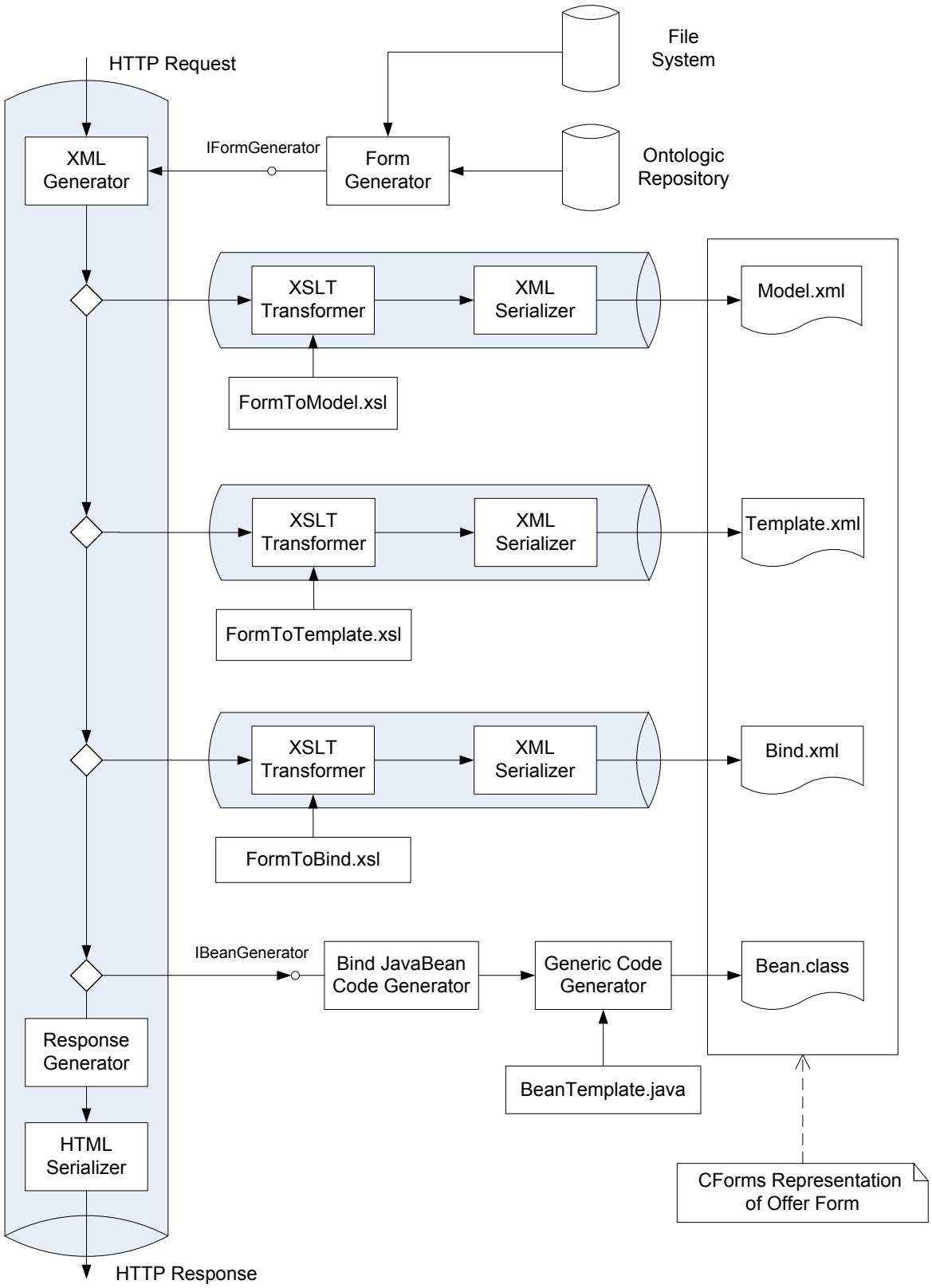
Pre generovanie vnútornej reprezentácie ponuky vytvoríme formulár s jedným tlačidlom. Pre obslúženie kliknutia na toto tlačidlo definujeme v aplikačnom rámci Cocoon skript, ktorý postupne vykoná prvé štyri z nasledujúcich dátovodov (pozri Obr. 19):

- dátovod generujúci model formulára,
- dátovod generujúci šablónu formulára,
- dátovod generujúci naviazanie formulára na triedy Java Bean,
- dátovod generujúci triedy Java Bean,
- dátovod zobrazujúci samotný formulár.

Logika prvých troch dátovodov generujúcich súbory XML v zápise Cocoon Forms je rovnaká. Štvrtý dátovod generuje Java triedy (Java Beans) potrebné na naviazanie hodnôt z formulára.

Na začiatku každého dátovodu je generátor implementovaný ako trieda v jazyku Java, ktorá generuje XML súbor pre vstup do transformátora. Obr. 19 znázorňuje generátor produkujúci výstup na základe ontológie, ktorá je uložená v ontologickom úložisku Sesame. Medzi ontologickým úložiskom a generátorom sa nachádza komponent, ktorý má dve funkcie:

- načítanie dát z ontologického úložiska
- poskytnutie týchto dát generátoru cez spoločné rozhranie využívajúce rozhranie List v jazyku Java



Obr. 19 Dátovody na generovanie formulára v CForms reprezentácii.

Výhodou uvedeného prístupu je nezávislosť dátovodu na type úložiska. Základ tvorí hlavný dátovod zobrazený vľavo. Jeho úlohou je získať formulár z komponentu *Form Generator* cez rozhranie *IFormGenerator*, pretransformovať ho do XML a postupne spúšťať ďalšie dátovody, prípadne iné komponenty, ktoré transformujú túto reprezentáciu do troch súborov

a jednej skupiny súborov. Tieto súbory spolu tvoria Cocoon formulár. Podľa úspešnosti jednotlivých krokov generovania sa nakoniec vytvorí výstupný text pre administrátora, ktorý ho informuje o výsledku celého procesu.

Výstupom XSLT transformátora v prvom dátovode je opis modelu formy, v druhom je to šablóna formy a v treťom opis naviazania formuláru na Java Beany. Aj keď sú výstupy transformátorov v tvare XML, posúvajú sa medzi komponentmi len ako volania SAX udalostí. Preto je potrebný posledný komponent dátovodu - XML serializátor, ktorého výstupom je konečný XML súbor.

Dátovod generujúci triedy Java Bean pracuje mierne odlišným spôsobom. Tu sa nachádzajú dva transformátory. Výsledok transformátora *Bind JavaBean Code Generator* sú šablóny pre požadované Java Bean triedy. Tieto slúžia ako vstup ďalšiemu transformátoru *Generic Code Generator*, ktorý na základe vstupu vygeneruje triedy v jazyku Java a následne ich skompiluje a umiestni medzi ostatné súbory opisujúce vnútornú reprezentáciu formulára ponuky.

Výsledky prvých štyroch dátovodov slúžia ako vstup skriptu v Cocoone, ktorý na ich základe zobrazí výsledný formulár a umožní aj použitie akcií nad týmto formulárom. Vygenerovaná šablóna formuláru slúži ako vstup pre generátor. Generátor nahradí premenné skutočnými hodnotami. Následne sa výsledok generátoru spracuje transformátor, ktorý na základe modelu formuláru vygeneruje cieľový formulár. Pred tým sa však ešte v skripte formulár naviaže na triedy Java Bean. Po vyplnení ponuky a potvrdení pridania ponuky tlačidlom sa zavolá trieda, ktorá ponuku uloží do ontológie.

Posledný dátovod zobrazuje samotný formulár a je riadený skriptom. Vygenerovaná šablóna formuláru slúži ako vstup pre generátor. Generátor nahradí premenné skutočnými hodnotami. Potom sa výsledok generátoru spracuje transformátor, ktorý na základe modelu formuláru vygeneruje cieľový formulár. Pred tým sa však ešte v skripte formulár naviaže na triedy Java Bean. Po vyplnení ponuky a potvrdení pridania tlačidlom je zavolaná trieda, ktorá ponuku uloží do ontológie.

6.5.1. Form Generator

Na získavanie dát potrebných pre vytvorenie formulára z ontologického úložiska slúži komponent *Form Generator*. Okrem toho komponent analyzuje štruktúru ontológie, pre určenie grafických komponentov formulára.

V procese získavania dát o formulári pre jeho zobrazenie je potrebné získať aj údaje potrebné na jeho spätné uloženie do ontologického úložiska. Pre všetky položky z ontológie je potrebné získať vlastnosti *rdfs:label* a *URI*. Vlastnosť *rdfs:label* je text, ktorý sa zobrazí používateľovi, *URI* je jedinečný identifikátor danej položky. Ak sa jedná o triedu, zaujíma nás zoznam jej podtried, a rovnako zoznam existujúcich inštancií. Pre možnosť spätného uloženia informácií z formulára do ontológie je potrebné si pamätať aj *URI* hrán (predikátov), ktorými sú spojené vlastnosti triedy s danou triedou, pretože všetky dáta v ontológii sú uložené ako tvrdenia subjekt – predikát – objekt.

Proces analýzy štruktúry ontológie a určení grafických komponentov spočíva v prehľadávaní ontológie a pravidlách, pre rozoznávanie viacerých grafických komponentov. Pravidlá priraďujú vzory v ontológii ku grafickým komponentom, pričom existuje problémov pri určovaní komponentov. Jedným je určenie vlastnosti alebo triedy, ktorá bude nakoniec znázornená ako grafický komponent alebo bude len hodnotou v nejakom komponente. Zatiaľ sme identifikovali nasledovné vzory v ontológii:

- Trieda bude mať vo formulári vyjadrenie v ovládacom prvku typu *droplist*, ak má podtriedy, ktoré už nemajú ďalšie podtriedy. Prvkami *droplistu* budú jednotlivé podtriedy (ich rdfs:label). Týmto spôsobom si používateľ vyberie typ inštancie,

ktorú chce vytvoriť (ak nepredpokladáme, že nadtrieda môže mať inštancie – je definovaná ako zjednotenie podtried). Okrem toho bude potrebné zobraziť grafické komponenty pre naplnenie vlastností inštancie triedy.

- V prípade, že trieda má podtriedy a tieto majú takisto podtriedy (vetvenie do ľubovoľnej hĺbky), bude reprezentovaná *listboxom*, v ktorom bude zoznam podtried prvej úrovne. Po vybratí určitej podtriedy sa v tom istom *listboxe* zobrazia podtriedy prvej úrovne tejto konkrétnej podtriedy. Label *listboxu* sa upraví tak, aby vyjadroval zanorenie v úrovni tried. Pod *listboxom* bude tlačidlo pre návrat vyššie. Okrem tohto *listboxu* bude potrebné zobraziť aj grafické komponenty pre naplnenie vlastností inštancie triedy.
- Trieda bude vystupovať ako *droplist*, ak existujú jej inštancie a kardinalita relácie triedy je 1. Prvkami *droplistu* budú jednotlivé inštancie (ID). Tým je zabezpečený výber existujúcich inštancií. Ak bude umožnené pridávať nové inštancie, budú potrebné grafické komponenty pre naplnenie inštancie.
- Trieda bude vystupovať ako *listbox*, ak existujú jej inštancie a kardinalita relácie triedy je viac ako 1. Tým je zabezpečený výber existujúcich inštancií. Ak bude umožnené pridávať nové inštancie, budú potrebné grafické komponenty pre naplnenie inštancie.

Rozhranie medzi týmto komponentom a generátorom je zabezpečené cez rozhranie IFormGenerator. Toto rozhranie poskytuje prístup k zoznamu grafických komponentov a im zodpovedajúcim hodnotám. Ak je typ prislúchajúcej hodnoty trieda, bude táto hodnota vyjadrená ďalším podobným zoznamom. Každá položka v zozname, či už trieda alebo hodnota, bude mať už spomínané atribúty *rdfs:label*, *URI* a zoznam URI hrán. V prototype sme overili, že vieme z ontológie zisťovať potrebné vlastnosti.

6.6. Zhodnotenie prototypu

Za cieľ prototypovania sme si určili naštudovanie a overenie potrebných technológií a ich spoločné využitie na príklade ukážkového rozhrania portálu. Myslíme si, že sa nám tento cieľ z väčšej časti podarilo naplniť. Naštudovali sme si potrebné technológie a oboznámili sa s problémami, ktoré nás čakajú pri ich použití. Práve počas riešenia problémov s technológiami sme zistili, že niektoré nie je možné v rozumnom čase riešiť ináč ako nepoužitím danej technológie, resp. jej nahradením inou. Týmto spôsobom sme zamietli využitie EJB a zmenili technológiu XForms za CForms.

Dôležitým míľnikom pri tvorbe prototypu bolo práve objavenie CForms v Cocoone a ich využitie k tvorbe formulárov pre ponuky pracovných príležitostí. Využitie CForms sprevádzalo vynaloženie veľkého množstva úsilia, ktoré bolo nevyhnutné na odskúšanie a sfunkčnenie formulárov pomocou existujúcich ukážok kódu pre Cocoon.

Z ďalších technológií sme úspešne overili prístup k ontologickému úložisku Sesame a využitie rámcov Jetspeed a Cocoon, kde sme porozumeli základným princípom zápisu toku riadenia cez flow. Mierne problematickou sa ukázala spolupráca rámcov Cocoon a Jetspeed (pozri 6.6.1).

Za hlavný problém využitia spomenutých technológií nepokladáme ani tak ich zložitosť ako skôr nestabilitu niektorých z nich a predovšetkým veľký nedostatok primeranej dokumentácie. Práve tu pokladáme za najväčší prínos prototypovania oboznámenie sa s uvedenými technológiami a následné zníženie neistoty a tým aj rizík spojených s ich využitím.

Vytvorený prototyp pokladáme za dobrý základ pre pokračovanie v riešení portálu.

6.6.1. Skúsenosti s Jetspeedom a Cocoonom

Jetspeed sa javí byť silným nástrojom na tvorbu portálov. Ako veľmi problematická sa ukázala jeho úvodná konfigurácia a pochopenie ako sa vlastne portál v Jetspeede vytvára. Tieto problémy vychádzajú do veľkej miery z toho, že k nemu neexistuje dostatočná dokumentácia a všetko treba ručne hľadať a skúšať. Po rozbehnutí a nakonfigurovaní je však k dispozícii portál s veľkým rozsahom funkcií a možností, s ľahkou úpravou a využívaním týchto funkcií.

Prototypovanie odhalilo problematickú komunikáciu Jetspeedu s Cocoonom, ktorá je realizovaná iba na úrovni URL adries. Jetspeed v tomto prípade vyzerá ako umelo prilepená časť v celom riešení portálu. Prototyp teda nastolil otázku, či je v našom projekte vhodné použiť Jetspeed alebo sa radšej prikloniť k portálovému riešeniu, ktoré ponúka samotný Cocoon a ktoré následne poskytuje lepšie možnosti vzájomnej komunikácie.

7. Produkt

V tejto kapitole opisujeme vytvorený produkt – portál pracovných príležitostí – webovú aplikáciu, ktorá umožňuje zadávateľom pracovných ponúk zadávať a spravovať ponuky pracovných pozícií za ich spoločnosti. Zároveň aplikácia umožňuje potenciálnym záujemcom o prácu jednoduché prezeranie zadaných ponúk a vyhľadávanie na základe kritérií.

Vytvorený portál na rozdiel od väčšiny podobných aplikácií využíva na opis dát doménovú ontológiu a dáta samotné ukladá do ontologického úložiska. To umožňuje konzumentom ponúk vyhľadávať vhodné ponuky na základe jednoduchých sémantických dopytov vo forme fazetového sémantického prehliadača.

Na prvý pohľad skrytou ale vo svoje podstate kľúčovou črtou aplikácie je jej flexibilnosť, ktorú chápeme vo viacerých smeroch:

- Aplikácia je flexibilná voči použitej doménovej ontológii. Portál bol navrhnutý a implementovaný tak, aby sa dal jednoducho prispôsobiť pre iné domény ako je doména pracovných ponúk.
- Aplikácia má flexibilnú architektúru, ktorá umožňuje jednoduché pridávanie ďalších aplikácií a funkcionality. Portál má ambíciu stať sa miestom integrácie viacerých nástrojov, ktoré pracujú so znalosťami vo forme ontológií a prispievajú akýmkoľvek spôsobom k prezentácii týchto znalostí používateľovi.
- Aplikácia je flexibilná voči používateľovi. Portál obsahuje aj prvky adaptability a je navrhnutý tak, aby ho nástroje pracujúce s modelom používateľa dokázali ľahko prispôsobovať konkrétnemu používateľovi.

V tejto kapitole opisujeme koncepciu a základné vlastnosti jednotlivých častí výsledného produktu. Podrobný opis riešenia z implementačného hľadiska sa nachádza v technickej dokumentácii uvedenej v prílohe D.

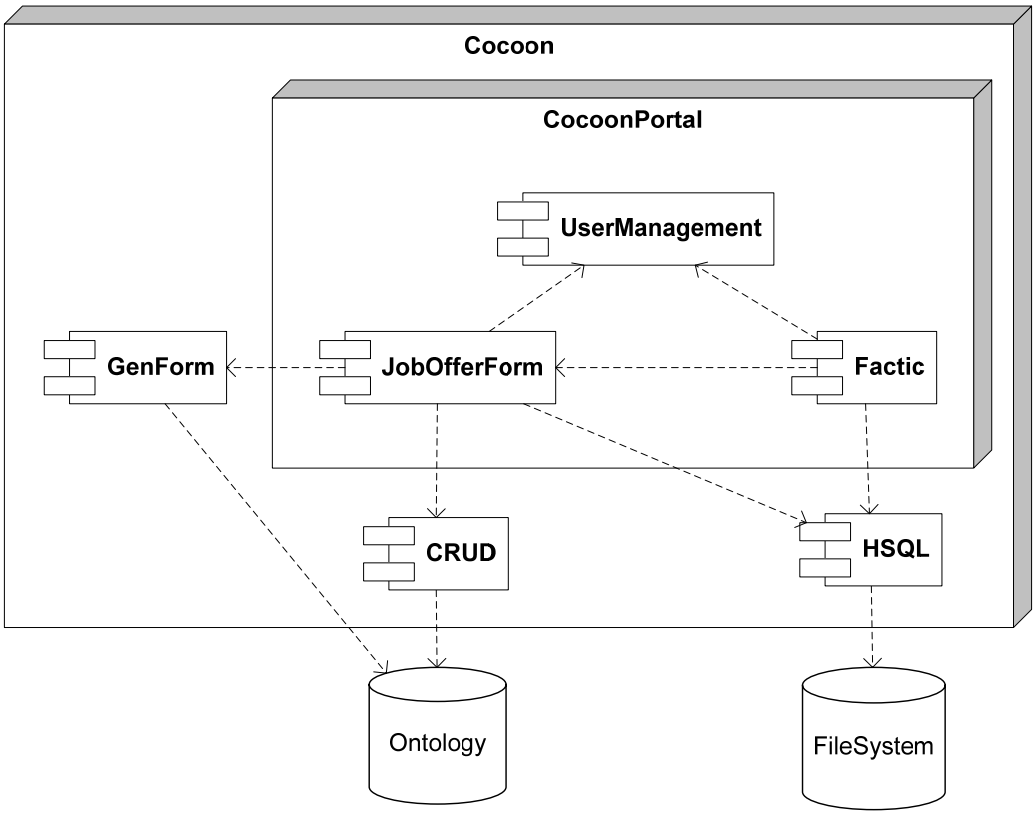
7.1. Architektúra

Architektúra riešenia sa oproti pôvodnému návrhu značne zjednodušila. Na základe výsledkov prototypovania sme z návrhu vylúčili portálové riešenie JetSpeed, keďže neposkytovalo vhodné možnosti komunikácie s aplikačnou vrstvou v Cocoone. Práve JetSpeed z veľkej časti nahrádza použitý rámec Cocoon a jeho portálové riešenie *Portal Block.*

Prehľad architektúry portálu je na Obr. 20. Do Cocoonu sme nasadili fazetový prehliadač (komponent *Factic*) a aplikáciu pre zobrazenie formulárov (komponent *JobOfferForm*). Keďže Cocoon neobsahoval implementáciu správy používateľov ale len zodpovedajúce rozhrania, implementovali sme kompletnú správu používateľov pre portál (komponent *UserManagement*).

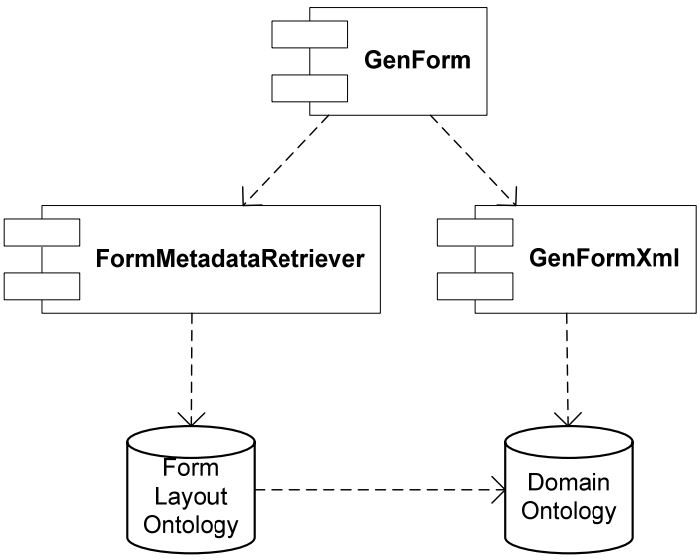
Vzor Create-Retrieve-Update-Delete nad ontologickou databázou realizuje komponent *CRUD*, ktorý poskytuje mapovanie medzi ontologickou a objektovou reprezentáciou ponuky. Na uchovávanie neontologických údajov o ponuke, ako je napríklad čas poslednej zmeny alebo prihlasovacie meno autora ponuky, slúži komponent *HSQL*.

Pôvodný návrh generovania formulárov sme zmenili v dôsledku problémov so spracovaním reťazcov v XSL, ktoré sa vyskytli pri jeho realizácii. Pôvodný návrh počítal s generovaním vnútornej XML reprezentácie formulára, z ktorej sa následne pomocou XSL transformácií mal generovať opis formulára do XML súborov potrebných pre rámec CForms. Objekty Java bean sa mali generovať pomocou Java reflection.



Obr. 20. Prehľad architektúry portálu.

Naše aktuálne riešenie (Obr. 21) je celé založené na Jave, pomocou ktorej generuje opis formulára z vnútornej reprezentácie v podobe XML. Komponent *GenForm* z architektúry (pôvodne *Form Generator*) sme dekomponovali na tri časti – *FormMetadataRetriever*, ktorý sprístupňuje dáta o výzore formulára a *GenFormXML*, ktorý generuje opis polí formulárov z doménovej ontológie. Následne *GenForm* využíva služby oboch komponentov pre generovanie opisu formulárov a Java Bean objektov pre CForms.



Obr. 21. Dekompozícia generovania formulárov.

7.2. Cocoon Portal

Pre realizáciu portálu sme na základe výsledkov prototypovania použili rámec Apache Cocoon a jeho blok *Portal*[3]. Základný portál poskytuje službu prihlasovania sa používateľov a s tým súvisiaceho základného rozdelenia portálu na verejnú a privátnu časť. Zároveň portál obsahuje implementáciu copletov (ekvivalent portletov) so širokými možnosťami adaptability. Používateľ si môže zvoliť coplety, ktoré chce mať zobrazené na stránke a zvoliť ich umiestnenie. Ďalším významným prvkom adaptability je podpora prispôsobenia celkového vzhľadu portálu (*skins*).

Portal blok v aktuálnej verzii má zadefinované rozhrania pre manažment používateľov (pridávanie/odoberanie používateľov, zmena profilu a role používateľov), avšak neobsahuje implementáciu týchto rozhraní. V základnom portáli sa teda nedá pracovať s profilmi používateľov, ktoré bolo potrebné doimplementovať.

7.2.1. Bezpečnosť a riadenie prístupu

Autentifikácia

Po úspešnom zalogovaní sa je používateľ autentifikovaný – systém vie, kto používateľ je. Prístupové meno a heslo sú uložené v autentifikačnom poskytovateľovi, ktorého zvyčajne tvorí nejaký typ databázy (napr. LDAP, RelDB, XML).

Autorizácia

Úspešná autentifikácia je základným predpokladom pre prístup používateľa k chráneným častiam aplikácie. Autorizácia určuje, ku ktorým špecifickým častiam má autentifikovaný používateľ prístup a následne akú úroveň prístupu má povolenú.

V našom portáli sa to, či je používateľ autorizovaný a môže pristúpiť k danému zdroju určuje na základe rolí. Nadefinovali sme nasledovné role:

- *Administrátor* – má prístup ku všetkým častiam aplikácie.
- *Administrátor organizácie* – má prístup k častiam aplikácie, ktoré sa týkajú organizácie používateľa. Administrátor organizácie môže vykonávať správu používateľov danej organizácie a manažment všetkých ponúk danej organizácie.
- *Normálny používateľ* – môže meniť iba svoj profil a spravovať vlastné (používateľom vytvorené) ponuky.

Bezpečnosť v prezentačnom serveri Cocoon

Autentifikácia je v Cocoone zabezpečená autentifikačným rámcom, ktorý je súčasťou Cocoonu. Rámec dokáže chrániť rovnakým spôsobom niekoľko dokumentov (pričom pod pojmom dokument môžeme chápať požiadavku na dátovod, súbor, atď.) tak, aby mal používateľ po autorizácii prístup ku všetkým. Rovnako podporuje aj viacero spôsobov autorizácie – každý dokument alebo skupina dokumentov môže byť chránený zvlášť.

Autentizačný rámec obsahuje komponenty, ktoré sa dajú využiť v sitemape. Sú to predovšetkým akcie, ktoré riadia samotné spracovanie HTTP požiadavky v dátovode. Autorizačné skupiny sú vyjadrené pomocou *handlerov*, ktoré sú priradené k jednotlivým dokumentom. Spravovanie konfigurácií jednotlivých handlerov má na starosti Autentizačný manažér.

[3] Cocoon poskytuje dve portálové riešenia: stabilný *portal framework* a novší *portal engine,* ktorý sme použili.

O jednotlivých používateľoch je v *session* vytvorený objekt *authentification*. Tento obsahuje okrem základných údajov(id, rola) aj položku *data*, do ktorej je možné ukladať doplňujúce údaje. Rámec neumožňuje priraďovanie viacerých rolí jednému používateľovi.

Ak používateľ nemá mať prístup k požadovanému dokumentu, je možné zavolať náhradný dátovod (ktorý typicky zobrazí obrazovku s chybovým hlásením, prípadne vyzve používateľa aby sa autentifikoval).

Pre kompletnú správu riadenia prístupu by *Portal* mal zabezpečovať:

- Použitie viacerých autentifikačných manažérov
- Manažment používateľov
- Manažment rolí
- Mapovanie používateľov na role
- Manažment práv pre konkrétne role

Cocoon samotný v súčasnosti umožňuje iba použitie viacerých autentifikačných manažérov (prvý bod). Implementácia manažmentu používateľov, rolí a prístupových práv sa očakáva v ďalších vydaniach rámca Cocoon. Napriek tomu už existujú v Cocoon API triedy, ktoré implementujú niektoré metódy potenciálne využiteľné pri riešení problémov s bezpečnosťou.

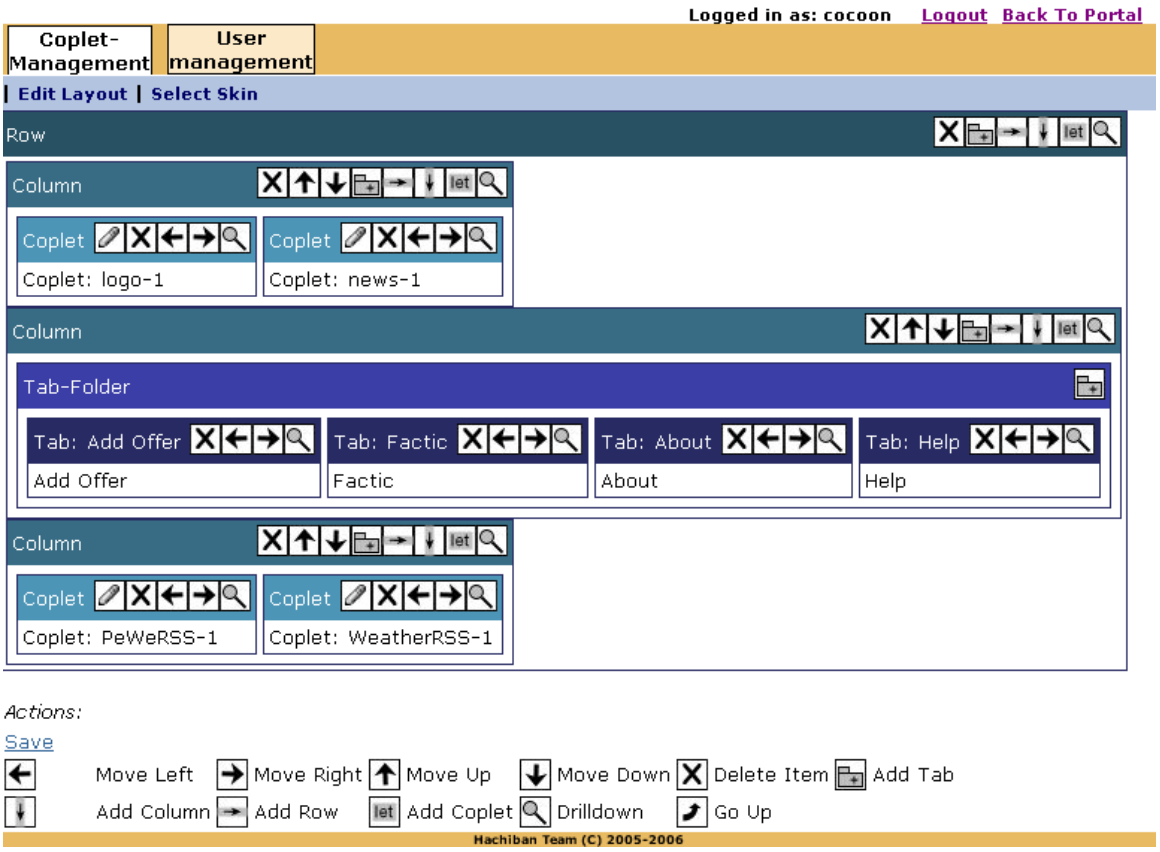
Dopracovanie implementácia bezpečnosti v Cocoone

Naše riešenie spočíva v napísaní vlastných Cocoon flowscript funkcií a Java tried, ktoré nadväzujú na koncepciu autentizačného frameworku a dopĺňajú chýbajúcu funkcionalitu. Uvedený postup umožňuje budúci bezproblémový prechod na verziu Cocoonu, v ktorej bude kompletná implementácia manažmentu používateľov.

Delenie prístupu používateľov k zdrojom na základe rolí sme riešili na úrovni sitemap pomocou komponentu *selector*. Toto riešenie nie je ideálne, ale keďže táto časť rámca Cocoon bola pôvodne úplne nefunkčná boli sme nútení použiť náhradné riešenie. Rôzny obsah a správanie formulárov slúžiacich pre správu používateľov sme riešili dynamicky na úrovni Cocoon flowscript funkcie, ktorá formulár zobrazuje.

7.2.2. Adaptabilita portálu

Portal blok má v sebe implementované prvky adaptability. Každý prihlásený používateľ si môže upraviť rozmiestnenie copletov podľa svojich preferencií (ukážka rozhrania je na obrázku 22). Okrem rozmiestnenia copletov má používateľ možnosť pridávať dostupné coplety a odoberať tie coplety, ktoré to majú povolené. V prípade, že používateľ využije túto možnosť, portál vytvorí profil rozmiestnenia stránky pre konkrétneho používateľa a ďalej ho používa namiesto prednastaveného.



Obr. 22. Ukážka manažmentu copletov v časti *Tools* portálu

Okrem uvedenej možnosti úpravy rozmiestnenia copletov, ktorá sa ukladá do profilu má používateľ možnosť pri práci minimalizovať, maximalizovať a zatvárať coplety, pri ktorých je to povolené.

Ďalšou možnosťou adaptability, ktorú implementuje blok *Portal* je podpora zmeny celkového vzhľadu portálu pomocou tzv. skinov. Správca portálu môže nadefinovať ľubovoľné množstvo skinov z ktorých si následne používateľ môže vybrať. Každý skin tvoria obrázky, šablóny, kaskádové štýly a prípadné JavaScript súbory.

7.2.3. Internacionalizácia

Významnou vlastnosťou Cocoonu a teda aj bloku *Portal* je podpora internacionalizácie pomocou vstavaného i18n transformátora. Tento transformátor nahrádza špeciálne značky v šablónach rozhraní hodnotami z príslušného slovníka podľa zadanej lokality.

Naše riešenie sme pripravili na budúcu internacionalizáciu používaním i18n značiek v definíciách používateľských rozhraní namiesto napevno zadefinovaných textových hodnôt. Vytvorili sme anglickú mutáciu portálu vytvorením anglických slovníkov, ktoré využíva i18n transformácia.

Internacionalizáciu formulárov, ktoré sa generujú z ontológie sme zabezpečili parametrizovateľnosťou metód, ktoré z ontologického úložiska získavajú štruktúru ontológie. Pri metódach je možnosť zadať ako parameter jazyk opisov položiek („label“), ktoré sa načítavajú z ontológie.

7.3. Navigačná štruktúra portálu

Pri návrhu a implementácii portálu sme rozpracovali počiatočnú navigačnú štruktúru portálu. Navrhli sme verejnú a privátnu zónu portálu, pričom privátnu zónu sme rozdelili podľa rôznych rolí používateľov v systéme (administrátor systému, zamestnanci organizácií). Výsledná navigačná štruktúra verejnú, privátnu a v rámci nej systémovú časť (pozri prílohu D).

Kvôli neočakávaným komplikáciám s manažmentom používateľov, riadením prístupu používateľov ku zdrojom, nemožnosti priradiť jednému používateľovi viacero rolí a následnému nedostatku zdrojov sme implementovali len časť navrhnutej navigačnej štruktúry portálu. Vytvorený portál však napriek tomu poskytuje potrebnú základnú funkcionalitu vrátane správy používateľov a spracovania prístupových práv.

7.4. Fazetový prehliadač – Factic

Na prehliadanie informácií o ponukách a vyhľadávanie konkrétnych ponúk sme využili nástroj *Factic* – fazetový prehliadač zo štátneho programu NAZOU[4]. Samotný nástroj sme integrovali do portálu a prispôsobili jeho potrebám. Prepojenie fazetového prehliadača s portálom je bližšie opísané v technickej dokumentácii v prílohe D. Vzhľad používateľského rozhrania *Facticu* sme upravili pomocou XSL šablón podľa štýlu zobrazovania portálu.

Do rozhrania *Facticu* sme pridali aktívne prvky – tlačidlá, ktoré používateľovi umožňujú v portáli vykonať rôzne operácie s inštanciami ponúk – ich editovanie, zmazanie, zverejnenie alebo skrytie. Prispôsobili sme tiež prezentáciu inštancií pomocou farby pozadia, ktorá indikuje skrytie/zverejnenie ponuky.

Pretože portál umožňuje editovanie ponúk za rôzne organizácie rôznymi používateľmi, rozšírili sme *Factic* o podporu spracovania prístupových privilégií používateľov. Na zistenie údajov o používateľovi využívame prostriedky Cocoonu, pričom samotné spracovanie vykonávame pomocou XSL šablón, ktoré upravia výstup z jadra fazetového prehliadača.

7.5. Spracovanie formulárov CForms

7.5.1. Formuláre v portáli

Pre spracovanie formulárov sme použili rámec CForms (pozri kap. 6.4.2), ktorý poskytuje širokú paletu ovládacích prvkov, prepojenie formulára na objekt Java bean, automatickú kontrolu správneho vyplnenia formulára (správne dátové typy, vyplnené povinné polia).

Formuláre založené na CForms sme použili na troch miestach:

- Formulár ponuky
- Formuláre registrácie
- Formuláre správy používateľov

Formulár ponuky

Pracovná ponuka je v systéme reprezentovaná ako jeden formulár. Znamená to, že má jeden model, ktorý je zobrazený pomocou jednej šablóny a jedným konfiguračným súborom je prepojený na objekty Java bean.

[4] NAZOU, http://nazou.fiit.stuba.sk

Pri tomto formulári sme využili možnosť CForms štruktúrovať formulár do tabiek. Takisto sme využili ovládací prvok *TreeWidget* pre zobrazenie hierarchických štruktúr, ich prezeranie a výber položiek z nich. V prípade, že bola niektorá vlastnosť v doménovej ontológii viacnásobná, vložili sme príslušné ovládacie prvky do prvku *repeater.*

Formulár je štruktúrovaný do nasledujúcich častí

- *General* – obsahuje polia pre vyplnenie základných informácií o ponuke
- *Duty Location* – umožňuje výber regiónu, v ktorom sa ponúka práca
- *Apply Information* – umožňuje zadať inštrukcie pre záujemcov o ponuku
- *Benefits* – slúži na zadanie výhod súvisiacich s ponúkanou prácou
- *Contract* – umožňuje zadať zmluvné a platové podmienky ponuky
- *Requirements* – slúži na zadávanie požiadaviek na uchádzača

Uvedená štruktúra je predvolená, avšak nič nebráni správcovi systému v zadefinovaní novej (pozri kap. 7.6.2).

Formuláre registrácie

Pri registrovaní používateľ vypĺňa dva formuláre. V prvom vyplní údaje o organizácii, ktorá sa touto registráciou zavedie do systému a v druhom údaje o sebe, ako správcovi za danú organizáciu. Ten sa pridá do systému ako neaktívny používateľ, ktorého musí schváliť správca systému.

Formuláre správy používateľov

Tieto formuláre umožňujú prezeranie, editovanie a pridávanie používateľov. Pri týchto formulároch sa dynamicky mení ich model v závislosti od role používateľa (napríklad správca organizácie logicky nemôže pridať používateľa, ktorý by bol celkovým správcom systému). Dynamická zmena modelu formulára cez objektový model umožňuje použitie jedného základu pre všetky typy používateľov.

7.5.2. Životný cyklus formulára

Každý CForms formulár v Cocoone je definovaný minimálne dvomi konfiguračnými súbormi:

- *Model* – definuje *widgety* formulára a dátové typy, ktoré týmto *widgetom* zodpovedajú. V modeli sa definuje aj validácia jednotlivých *widgetov,* kontexová nápoveda a prípadne obsluhy udalostí, ktoré môžu nastať vo *widgete* (napr. *on-change*).
- *Šablóna* – definuje rozmiestnenie *widgetov* vo formulári a akciu, ktorá sa s formulárom vykoná. CForms umožňuje v šablóne definovať a používať makrá a JavaScript, ktoré umožňujú použitie rôznych pokročilých widgetov a spôsobov ich rozmiestnenia (napr. *TreeWidget,* tabky).

Tretím, nepovinným konfiguračným súborom je *binding* obsahujúci pravidlá mapovania prvkov formulára do elementov XML alebo polí java bean objektov. V našom riešení sme použili *binding* pre transformáciu medzi formulárom a java bean objektami.

Najdôležitejší formulár nášho riešenia je dynamicky generovaný a slúži na realizáciu vzoru CRUD (Create-Retrieve-Update-Delete) nad inštanciami zvolenej triedy (v našom prípade sa jedná o pracovnú ponuku).

Životný cyklus dynamicky generovaného formulára [19] a pracovnej ponuky, ktorá je v ňom uložená, zobrazená a editovaná je nasledovný:

1. Na základe štruktúry ontológie a ďalších informácií o výzore formulára sa vygeneruje:
    1.1 model formulára
    1.2 šablóna formulára
    1.3 java bean objekty zodpovedajúce modelu formulára
    1.4 pravidlá pre mapovanie formulára do java bean objektov
    1.5 pravidlá pre mapovanie java bean objektov do ontológie
2. Z ontologického úložiska sa pomocou ontologicko-objektového mapovača získa objektová reprezentácia formulára (*Retrieve*)
3. Obsah polí java bean tried sa namapuje na formulár
4. Formulár sa zobrazí, používateľ edituje ponuku
5. Zmenený a validovaný obsah formulára sa namapuje späť do polí java bean tried
6. Obsah Java bean tried sa pomocou objektovo ontologického mapovača uloží späť do ontologického úložiska. (*Update*)

7.5.3. Realizácia CRUD

Vzor CRUD je často používaným základným vzorom pre entity uložené v relačnej databáze [20]. Existujú objektovo-relačné mapovače (napr. Hibernate[5]), ktoré zjednodušujú realizáciu vzoru CRUD na napísanie mapovacích pravidiel (napr. vo forme java annotations priamo do kódu).

Keďže náš portál používa na uchovávanie dát ontologické úložisko Sesame[6], rozhodli sme sa vytvoriť pre realizáciu vzoru CRUD objektovo-ontologický mapovač.

Objekty predstavujú inštancie tried Java Bean, vytvorené pre potreby vnútornej reprezentácie entít v systéme. V rámci portálu ide o triedy, umožňujúce uchovávanie informácií o pracovnej ponuke, ktoré sa zobrazia (a môžu byť menené) vo formulári. Tieto objekty sú mapované na RDF graf, predstavujúci ontologickú reprezentáciu ponuky a uchovávané v ontologickom úložisku.

Riešenie v podobe vzoru CRUD umožňuje transformáciu objektov Java Bean do reprezentácie vo forme RDF grafu, ktorý je nasledovne uložený do ontologickej databázy (Obr. 23). Podobne je realizovaná aj opačná postupnosť – transformáciu z RDF reprezentácie získanej z úložiska do objektovej reprezentácie.

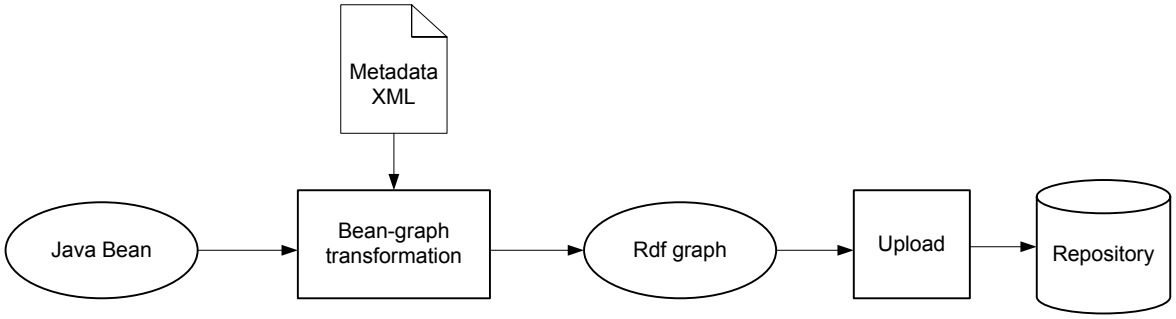
Problematike objektovo-ontologického mapovania nebola doposiaľ venovaná veľká pozornosť. Základným problémom, ktorý s týmto mapovaním súvisí je sémantická rozdielnosť medzi deskriptívnou logikou (*description logic*) a objektovo orientovaným prístupom. Ontológia z princípu ponúka omnoho širšie možnosti na opis entít ako objektový prístup. Vytvoriť súbor tried (v zmysle objektovo-orientovanej paradigmy programovania), ktoré sú schopné niesť informácie o nejakej entite, opísanej pomocou ontológie, je netriviálny problém [21].

Ontologickú entitu (napr. inštanciu triedy *jo:JobOffer*) reprezentujeme pomocou hierarchie tried Java Bean, ktorá čiastočne korešponduje s RDF grafom určitej entity. Každá

[5] Hibernate, object/relational persistence service, http://www.hibernate.org/
[6] Sesame, open source RDF database, http://www.openrdf.com/

trieda obsahuje *get* a *set* metódu, pre každú vlastnosť príslušnej ontologickej triedy. Navyše obsahuje ID, ktoré obsahuje URI inštancie danej ontologickej triedy.

Základná schéma mapovania objektov Java Bean do ontologickej reprezentácie v úložisku je znázornená na obrázku 23. Konverziu medzi objektmi a RDF grafmi, zabezpečujú Bean-graf transformácie, ktoré používajú dodatočné mapovacie pravidlá.



Obr. 23. Prehľad postupu objektovo-ontologického mapovania.

Mapovacie pravidlá

Mapovacie pravidlá obsahujú nevyhnutné informácie o previazaní objektov Java Bean tried a inštancií ontologických tried. Každé pravidlo obsahuje informácie o mapovaní premennej objektu do RDF trojice. Jedno pravidlo môže obsahovať nasledujúce informácie:

1. *BeanFieldName* – názov premennej v Java Bean triede (k premenným objektov agregovaných tried sa dostávame pomocou bodkovej notácie).
2. *OntologyPropertyURI* – predikát z ontológie pre príslušnú vlastnosť.
3. *Type* – typ vlastnosti (či ide o dátový alebo objektový typ, pričom v prípade dátového typu je uvedený konkrétny typ).
4. *Multiple* – multiplicita, ktorá hovorí o tom, či môže byť daná vlastnosť viacnásobná.
5. *Recursive* – rekurzívnosť, ktorá hovorí, či majú mapovacie algoritmy postupovať rekurzívne na vlastnosti asociovanej objektovej vlastnosti.
6. *RecursivelyDelete* – vymazávanie, ktoré určuje, či je potrebný pri vymazávaní rekurzívny postup.
7. *DoubleRange* – multiplicita cieľovej triedy, ktorá určuje, či daný predikát môže smerovať do viacerých rôznych tried, alebo len do jednej.
8. *BeanName* – názov Java Bean triedy, ktorá reprezentuje príslušnú ontologickú triedu.

Transformácia objektov do RDF grafu pre zápis do ontológie

Mapovanie objektov do ontologického úložiska je rekurzívny proces. Rekurzia je logickým riešením, keďže mapovaná entita je reprezentovaná súborom tried v hierarchickej štruktúre. Transformácia sa spúšťa na objekt, ktorý je na vrchole tejto hierarchie. V jednom kroku rekurzie sú transformované všetky atribúty objektu s priradenou hodnotou. Pre každý z nich je vytvorená RDF trojica (subjekt – predikát – *objekt*[7]), z ktorých sa poskladá výsledný RDF graf. Tie premenné objektu, ktoré neboli nastavované, nie sú transformované. Nasledovne je

[7] pre rozlíšenie objektu v zmysle RDF od objektu ako inštancie triedy v OO paradigme budeme RDF objekt značiť kurzívou

proces rekurzívne spustený na objektové premenné objektu, ktoré predstavujú prepojenie na inštancie ďalších Java Bean tried.

RDF trojica sa vytvára nasledovne:

- *subjekt* je vytvorený z ID objektu.
- *predikát* sa získa z mapovacích pravidiel.
- *objekt* je vytvorený ako literál v prípade, že premenná predstavuje jednoduchý dátový typ, alebo URI v prípade, že ide o objekt (iná inštancia Java Bean tried):
    - literál je vytvorený z hodnoty danej premennej a jeho typ sa určí z mapovacích pravidiel
    - URI je určené ako ID objektu, ktorý je objektovou premennou v práve transformovanom objekte

Transformácia RDF grafu z ontológie do objektov

Mapovanie RDF grafu do objektov Java Bean tried je rovnako rekurzívny proces ako v prípade mapovania objektov do RDF grafu. Proces mapovania je realizovaný nad hierarchickou grafovou štruktúrou, ktorá je zložená z jednotlivých podgrafov objektových vlastností inštancií ontologických tried. Transformácia začína s podgrafom na najvyššej úrovni tejto hierarchie.

V prvom kroku sú transformované trojice na prvej úrovni grafu (čiže tie, ktoré majú ako subjekt ID inštancie pracovnej ponuky) do objektu triedy, ktorá je na vrchole hierarchie Java Bean tried. Nasledovne môže byť proces rekurzívne spúšťaný na tie uzly, ktoré predstavujú URI objektovej vlastnosti, ktorú je potrebné transformovať do podobjektu. To, či sa bude proces rekurzívne spúšťať je závislé na informáciách v mapovacích pravidlách.

Každej premennej v objekte prislúcha trojica v RDF grafe. Pri postupnej transformácii sa pre každú trojicu v objekte nájde premenná, ktorá jej zodpovedá. Ak *objekt* v trojici je literál, tak sa naplní hodnota premennej podľa hodnoty literálu. Ak *objekt* predstavuje URI, tak sa vytvorí inštancia zodpovedajúcej Java Bean triedy. Vytvorenej inštancii sa nastaví ID podľa príslušného URI *objektu*. Táto inštancia sa potom priradí príslušnej premennej v objekte. Pri rekurzívnom postupe sa následne naplnia premenné vytvorenej inštancie.

7.6. Generovanie formulárov

Významnou časťou riešenia je generovanie formulárov pre rámec CForms. Pod týmto názvom chápeme generovanie všetkých častí potrebných pre realizáciu vzoru CRUD, ktoré sú závislé od zvolenej triedy, nad ktorou chceme vzor CRUD vykonávať. Menovite sa jedná o generovanie modelu a šablóny formulára, objektov Java Bean pre uchovávanie obsahu formulára, pravidiel pre mapovanie medzi formulárom a objektovou reprezentáciou a pravidiel pre mapovanie medzi objektovou a ontologickou reprezentáciou.

Generovanie pozostáva z dvoch relatívne nezávislých krokov:

1. Generovanie vnútornej reprezentácie formulára na základe štruktúry ontológie.
2. Generovanie jednotlivých súborov potrebných pre prácu s formulárom v prostredí Cocoon s využitím ontológie o výzore a špecifických vlastnostiach formulára.

Uvedené dvojkrokové riešenie okrem možnosti paralelnej práce poskytuje aj istú úroveň nezávislosti na použitom prezentačnom rámci. V prípade, že sa zmení spôsob zápisu formulárov v Cocoone alebo by sa malo generovanie formulárov implementovať v inom

prostredí, je potrebné upravovať iba druhý krok procesu – vnútorná reprezentácia formulára ostáva nezmenená.

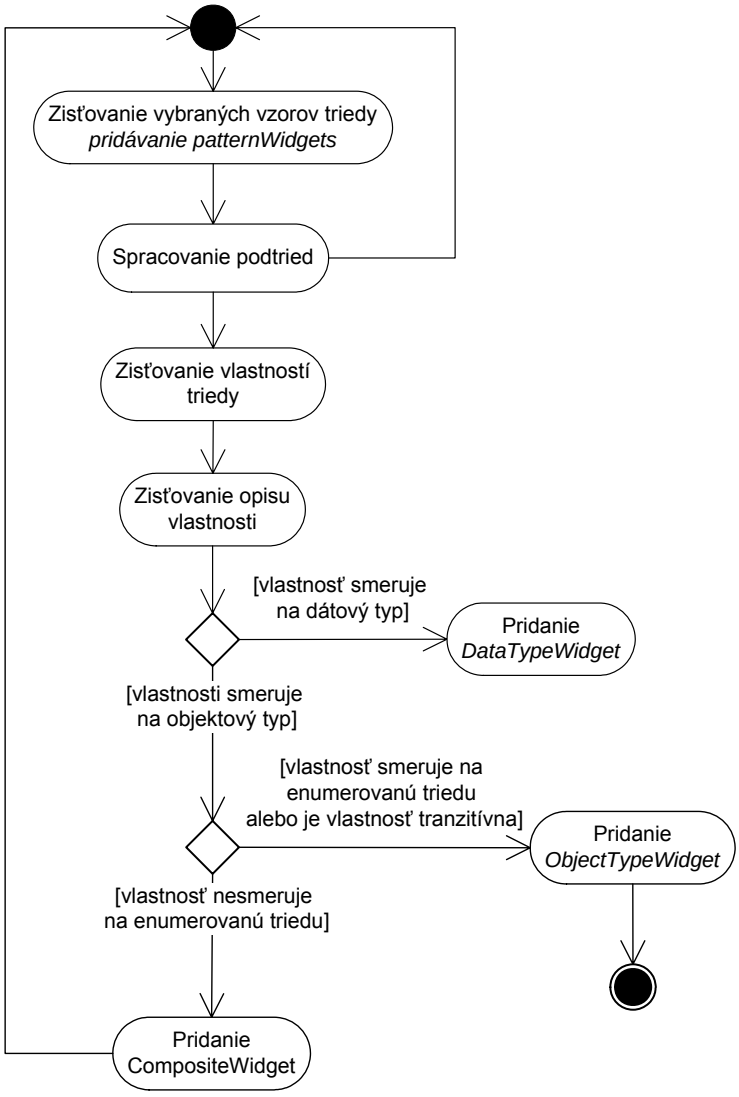
7.6.1. Generovanie vnútornej reprezentácie formulára

Generovanie vnútornej reprezentácie formulára je prvým krokom procesu generovania formulárov. Jeho vstupom je URI triedy, pre ktorú generujeme formulár a jazyk opisov vlastností formulára. Výstupom je XML súbor, v ktorom je zapísaná hierarchia *widgetov* potrebných pre editovanie inštancií zvolenej triedy (to znamená všetkých jej dátových a objektových vlastností).

Ontologické vzory

Generovanie vnútornej reprezentácie prebieha na základe vzorov, ktoré sa vyhľadávajú v ontológii. Vzory prepájajú štruktúru ontológie s formulárovými prvkami, ktoré slúžia na jej napĺňanie.

Vzory môžu byť jednoduché (napr. literálu *xsd:Date* zodpovedá *widget* s kalendárom) alebo zložité (ak je trieda enumerovaná, zodpovedá jej *widget* so zoznamom inštancií, ak má vlastnosť viacnásobnú kardinalitu, budú *widgety* triedy/literálu, na ktorú vlastnosť ukazuje zabalené v *repeatri*). Ďalšie príklady sú uvedené v [18].



Obr. 24. Algoritmus generovania vnútornej reprezentácie formulára.

Typy widgetov

Vnútorná reprezentácia rozlišuje tri typy *widgetov*:

- *DatatypeWidget* – predstavuje dátové vlastnosti.
- *ObjectTypeWidget* – predstavuje *widget* s asociovaným (hierarchickým) zoznamom inštancií pri enumerovaných triedach, resp. podtried pri hierarchiách.
- *CompositeWidget* – predstavuje *widget*, ktorý obsahuje ďalšie *widgety*.

Proces

Algoritmus generovania vnútornej reprezentácie formulára postupne prechádza cez všetky vlastnosti triedy s daným URI a rekurzívne sa spúšťa na všetky podtriedy danej triedy (Obr. 24).

Pri prechádzaní vlastnosťami sa skúma kontext každej vlastnosti. Pre každú dátovú

vlastnosť sa vytvorí zodpovedajúci *DataTypeWidget,* pre vlastnosť smerujúcu na enumerovanú triedu sa vytvorí a naplní *ObjectTypeWidget.* Pre štandardnú objektovú vlastnosť sa vytvára *CompositeWidget* a algoritmus sa rekurzívne spúšťa nad triedou, do ktorej objektová vlastnosť smeruje. Vytvorený objektový model, ktorý pozostáva z prepojených *Widgetov* sa na konci serializuje do XML.

7.6.2. Generovanie jednotlivých súborov

Výzor a špecifické vlastnosti formulára

Pri generovaní formulárov na základe štruktúry ontológie a vzorov, ktoré sa v nej nachádzajú sa dá vygenerovať kompletný model formulára, avšak jedným z cieľov nášho produktu je poskytnúť používateľsky príjemné a konfigurovateľné formuláre. Správcovi systému portál umožňuje nadefinovať nasledovné vlastnosti prvkov vo formulári:

- Štruktúrovanie prvkov formulára do väčších častí – v prípade portálu do záložiek.
- Poradie – myslí sa jednak poradie záložiek a jednak poradie prvkov v jednotlivých záložkách.
- Viditeľnosť – správca môže úplne skryť niektoré prvky formulára.
- Predvolená hodnota – ovládací prvok môže mať predvolenú hodnotu, typickú pre určitú skupinu používateľov.
- Výber existujúcej hodnoty – správca môže používateľom dovoliť okrem vytvorenia novej inštancie triedy znovu použiť už existujúce inštancie.

Tieto požiadavky nedokážeme pokryť iba na základe údajov zo samotnej doménovej ontológie, preto sme vytvorili ďalší model, ktorý by dokázal zachytiť rôzne konfigurácie vzhľadu formulárov. Tento model (príloha E) sme zadefinovali ako ďalšiu ontológiu, ktorá rozširuje doménovú ontológiu o špecifické informácie týkajúce sa formulárov.

Správca systému môže v tejto ontológii nadefinovať ľubovoľné množstvo šablón formulárov pre rôzne skupiny používateľov. Ontologické riešenie uľahčuje aj prispôsobovanie systému, kde každý používateľ môže mať vo svojom modeli odkaz na svoj výzor formulára, ktorý následne tvorí súčasť modelu používateľa. Táto súčasť môže byť menená rovnakými prostriedkami ako zvyšné časti modelu používateľa podľa toho, ako používateľ používa systém.

Z hľadiska procesu generovania výzoru formulára sme vytvorili v systéme modul (Obr. 21), ktorý sprístupňuje dáta o výzore formulára pre zadanú šablónu – *FormMetadataRetriever*. Tento sa využíva pri výslednom generovaní formulára z vnútornej XML reprezentácie. V budúcnosti je možné sprístupňovanie tejto ontológie prepracovať tak, aby využívalo objektovo-ontologický mapovač opísaný v kap. 7.5.3.

Spracovanie vnútornej reprezentácie formulára

V druhom kroku generovania formulárov sa z vnútornej reprezentácie formulára vygenerujú XML súbory potrebné pre rámec CForms.

V týchto XML súboroch sme identifikovali ucelené logické celky, pre ktoré sme vytvorili zodpovedajúce triedy v jazyku Java. Generovanie potrebných súborov spočíva vo vytvorení navzájom poprepájaných inštancií týchto tried a následnej serializácii tohto objektového modelu do potrebnej XML reprezentácie.

Proces generovania začína získaním zoznamu špecifických vlastností formulára o všetkých záložkách. Vytvorí sa koreňová inštancia hierarchie, ktorá zodpovedá šablóne. Do nej sa pridá špecifická časť pre každú záložku. Ďalej sa získajú informácie o rozmiestnení

ovládacích prvkov v samotnej záložke, pričom sa postupne vytvárajú inštancie tried, ktoré reprezentujú model, šablónu a väzobné informácie.

8. Záver

Na začiatku riešenia projektu sme vykonali analýzu problémovej oblasti a existujúcej dokumentácie k štátnemu programu NAZOU. Priebežne sme sa zaoberali analýzou nástrojov a technológií vhodných pre riešenie projektu v rámci ohraničení štátneho programu. Pokračovali sme prácou na špecifikácii požiadaviek na portál pracovných príležitostí a tvorbou hrubého návrhu systému.

Po prvom kontrolnom bode sme sa venovali analýze rizík pri tvorbe portálu, ktorej cieľom bolo identifikovať kritické časti portálu, ktoré by mohli zásadným spôsobom ohroziť úspešnosť celého projektu. Ako hlavné riziko sme identifikovali použitie a integráciu veľkého počtu nových a neodskúšaných technológií. Práve túto časť sme sa následne rozhodli prototypovať v závere zimného semestra.

Tvorbou prototypu portálu sme overili našu schopnosť naštudovať vybrané technológie a nástroje, pomocou ktorých sme následne overili celý cyklus práce s formulárom ponuky od jeho tvorby a načítanie dát z úložiska, cez jeho prezentáciu až po spätné uloženie dát do úložiska.

V druhej fáze riešenia počas letného semestra sme rozpracovali a upravili návrh podľa nových skutočností – absencia správy používateľov v Cocoone a obmedzenia správy prístupových práv, ktoré sme identifikovali až počas implementácie portálu. Aj napriek tomu sme implementovali portál pracovných príležitostí s podporou hlavných funkcií – vytvárania, editovania a prehliadania ponúk pracovných príležitostí.

Zamerali sme sa na flexibilné využitie formulárov pomocou ontologicko-objektového mapovača a (polo)automatické generovanie formulárov z ontológie. Pri generovaní sme upravili pôvodný návrh procesu z prvého semestra riešenia, ktorý bol založený na XSL transformáciách kvôli nepraktickej práci s reťazcami v XSL. O tejto problematike sme napísali dva odborné články. Prvý, na konferenciu IIT.SRC, bol prijatý a úspešne prezentovaný v rámci konferencie. Druhý sme zaslali na konferenciu ISD, pričom sme zatiaľ nedostali informáciu o výsledku posudzovania.

V kontexte projektu NAZOU sme portál prepojili s fazetovým prehliadačom Factic vytvoreným v rámci tohto projektu. Fazetový prehliadač sme prispôsobili a rozšírili pre potreby portálu. Keďže vývoj portálu bude v budúcnosti pokračovať v rámci štátneho programu a samotný portál má ambíciu stať sa významným integračným nástrojom, pokladáme za dôležitú vlastnosť výsledného riešenia možnosť jednoduchej integrácie nových nástrojov do portálu a adaptabilitu používateľského rozhrania podľa nastavení jednotlivých používateľov.

Zoznam použitej literatúry

[1] Apache Portals Project
http://portals.apache.org/

[2] Cocoon Forms Introduction
http://cocoon.apache.org/2.1/userdocs/basics/index.html (december 2005)

[3] Collins-Sussman B., Fitzpatrick B., Pilato M.: Version control with Subversion: For Subversion 1.1 (book compiled from Revision 1337)
http://svnbook.red-bean.com/en/1.1/svn-book.pdf

[4] Documentation Comments.
http://java.sun.com/docs/books/jls/first_edition/html/18.doc.html

[5] Eclipse.
http://www.eclipse.org

[6] G. Hohpe, B. Woolf: Enterprise integration patterns. Addison-Wesley. 2004.
ISBN: 0321200683

[7] Home of Sesame
http://openRDF.org

[8] How to Write Doc Comments for the Javadoc Tool.
http://java.sun.com/j2se/javadoc/writingdoccomments/

[9] Interval.cz – Nástroje pre webdesign
http://interval.cz/idcategory=17

[10] Javadoc - The Java API Documentation Generator.
http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/javadoc.html

[11] Jena – A Semantic Web Framework for Java
http://jena.sourceforge.net

[12] Jetspeed Portal Tutorial
http://portals.apache.org/jetspeed-1/tutorial/

[13] JSR-000168 Portlet Specification
http://www.jcp.org/aboutJava/communityprocess/review/jsr168/

[14] Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov: Výstup SV01/2

[15] Prístupnosť webových stránok pre používateľov s ťažkým zrakovým postihnutím
www.blindfriendly.sk/index.php?room=2

[16] Requirements for Writing Java API Specifications.
http://java.sun.com/j2se/javadoc/writingapispecs/index.html

[17] The SeRQL query language
http://www.openrdf.org/doc/sesame/users/ch06.html

[18] Barla M., Bartalos P., et al.: Ontology as an Information Base for a Family of Domain Oriented Portal Solutions. Submitted to ISD 2006.

[19] Barla M., Bartalos P., et al.: Dynamic Ontology Based Form Generation for Portal Solutions. IIT.SRC 2006, Bratislava, pp. 113-120.

[20] Ambler, S. W.: Mapping Objects to Relational Databases: O/R Mapping In Detail. (2006), http://www.agiledata.org/essays/mappingObjects.html (25.5.2006)

[21] Battle S, Jiménez D., Kalyanpur A., Padget J.: Automatic Mapping of OWL Ontologies into Java. In: Frank Maurer and Günther Ruhe (eds) Proc. of Title Suppressed Due to Excessive Length 11 the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004), (2004).

Príloha A Pravidlá pre návrh portálu pre zrakovo postihnutých používateľov

Aby bol portál použiteľný aj zrakovo postihnutými používateľmi je potrebné dodržať pravidlá 1-9. Dodržanie dodatočných pravidiel uvedených v závere, je vhodné pre zvýšenie efektívnosti a pohodlia práce nevidiacich používateľov pri práci s portálom.

1. Pre všetky obrázky a grafické objekty musí byť definovaná hodnota alt atribútu

Keďže nástroje sprístupňujúce nevidiacim používateľom web nie sú schopné reprezentovať obrázky a grafické objekty, je potrebné aby objekty slúžiace k navigácii mali nastavenú hodnotu alt (alternatíva) s opisom cieľovej stránky. Aj ostatné grafické objekty, ktoré nie sú odkazmi musia mať vyplnený atribút alt.

Príklad:

```
<p>Menu:</p>
<p><a href="index.html">
<img src="kino.gif" alt=- " Kino Orbis tu sa o ňom dozviete viac">
</a></p>
```

Aj v prípade, že je objekt definovaný prvkami [INPUT] alebo [APPLET] musí byť atribút alt vyplnený.

2. Všetky časti stránky musia byť prístupné nezávisle na použitej technológii.

Nové technológie na sprístupnenie a prezentáciu informácií môžu nevidiacemu používateľovi spôsobiť veľké problémy, ak mu znemožnia pokračovať v prezeraní stránky. Príkladom je menu vytvorené v JavaScripte, na ktoré je treba kliknúť aby sa rozbalilo. Odporúčaným pravidlom je vytvoriť ku každému podobnému objektu aj časť ktorou sa dá obísť (napríklad jednoduché menu v spodnej časti stránky).

3. Vyhýbanie sa použitiu tabuliek a pravidlá ich použitia.

Zrakovo postihnutí používateľ nemá stálu predstavu o celej tabuľke, lebo v danom momente vníma iba jednu jej časť. Nevidiaci len veľmi ťažko dokáže porovnať dve hodnoty v tabuľke. Reprezentácii údajov pomocou tabuľky sa treba vyhnúť, pokiaľ je to možné.

Ak je nutné použiť tabuľku:

- všetko čo patrí do jednej bunky treba definovať v jednej bunke
- čo patrí do viacerých buniek, treba vložiť do viacerých buniek
- tabuľka je pre používateľa linearizovaná po riadkoch
- tabuľka by mala byť orámovaná dostatočne výrazným rámom

4. Každý odkaz musí opisovať svoj cieľ bez okolitého kontextu.

Pri každom odkaze musí byť jasné kam odkazuje aj bez toho aby sa používateľ musel oboznámiť s celým obsahom stránky. Cieľ odkazu by mal byť zrejmý už z textu odkazu. Ak to nie je možné treba použiť atribút *title* HTML tagu <a>.

5. Stránka by mala byť prístupná aj pri rôznych farebných schémach.

Pri voľbe farebných kombinácií stránok je potrebné zohľadniť skutočnosť, že slabozrakí používatelia môžu mať nastavenú aj inú ako štandardnú farebnú schému.

6. Farby písma a podklad musia byť dostatočne kontrastné.

Farby písma a podkladu musia byť definované tak, aby boli dostatočne kontrastné. Kombinácie farby popredia (napr. písma) a farby alebo vzoru pozadia musia mať dostatočný vzájomný kontrast. Ak je podklad tvorený obrázkom, nesmie tento znižovať čitateľnosť textu.

7. Hodnoty atribútov a štýlov musia byť zadané pomocou relatívnych jednotiek.

Ak by bolo písmo pre čitateľa príliš malé môže si ho zväčšiť. Preto je potrebné napríklad pri písme používať v štýloch preddefinované hodnoty alebo relatívne jednotky. Takisto treba používať relatívne jednotky pri definíciách tabuliek a orámovania.

8. Nepoužívať efekty textu.

Pre zrakovo postihnutého používateľa môže byť nepríjemné používanie rôznych efektov v texte ako animácie alebo blikanie. Preto je lepšie sa takýmto prvkom vyhnúť.

9. Hlavný obsah stránky je na jej začiatku.

Zrakovo postihnutí používateľ vníma stránky lineárne. Ak sa pred hlavným obsahom stránky nachádza zdĺhavé menu, musí prejsť celým menu skôr ako sa dostane k obsahu stránky. Pomocou štýlov sa dá dosiahnuť primerané rozmiestnenie prvkov tak, aby nebolo potrebné všetky pomocné prvky dávať na začiatok stránky.

Ďalšie zásady

- V kóde je nevyhnutné ako prvé uvádzať hlavné prvky (navigačné menu, obsah stránky):

```
<p class="non-visual">
    <a href="#content" accesskey="1">skočiť na obsah</a>
    <a href="#menu">skočiť na menu</a>
</p>

.non-visual {
    position: absolute;
    width: 100px;
    left: -200px;
    }
```

- Navigačné menu by malo byť tvorené výhradne odrážkovými zoznamami <ul> aby nevidiaci získali zrozumiteľný obraz o hierarchickom rozdelení stránky.

```
<ul id="menu">
    <li><a href="…">Produkty</a>
        <ul id="menu-1">
            <li><a href="…">CMS</li>
            <li><a href="…">DMS</li>
        </ul>
    </li>
    <li><a href="…">Služby</a>
        <ul id="menu-2">
            <li><a href="…">Webaplikácie</a></li>
            <li><a href="…">Webhosting</a></li>
        </ul>
    </li>
</ul>
```

- Ak je hlavné menu je tvorené rôznymi „vyskakovacími“ odkazmi pomocou JavaScriptu je nutné aby aj hlavné položky boli odkazmi, pretože takto tvorené menu môže byť pre nevidiaceho mätúce.
- Otváranie nových okien je pre nevidiaceho často mätúci jav. Ak je potrebné otvárať nové okno, malo by byť už z názvu okna jasné o aké okno sa jedná.
- Pri formulárových elementoch je pre správnu orientáciu nutné naučiť sa správne používať tag <label>. Sú dve možné použitia:

```
<label for="name">Meno</label> <input type="text" id="name">
<label>Meno <input type="text" id="name"></label>
```

Príloha B Nízko-úrovňové rozdiely rámca Cocoon a prezentačného serveru Orbeon

		Orbeon 2.8	Cocoon 2.1
Dátovodový jazyk (XPL)	Podmienky	Áno, zabudovaný a postavený na XPath výrazoch.	S akciami (potrebuje Javu), selektory a tok.
	Iterácie	Áno	Nie
	XML Validácia	Áno, zabudovaná W3C Schéma a Relax NG validácia na každom vstupe a výstupe procesoru.	Len pri parsovaní XML.
	Agregácia	Áno, zabudovaná	Iba na úrovni generovania; cocoon: protokol môže byť použitý na agregáciu výsledkov iných dátovodov.
	Poddátovody	Áno, dátovody môžu byť volané ako XML komponenty, aj ako funkcie v iných programovacích jazykoch.	Protokol dovoľuje volanie iných dátovodov.
	XPointer Podpora	Áno	Nie v dátovodovom jazyku, ale XInclude transformer podporuje XPointer.
	Nelineárne dátovody	Áno, plná podpora	Limitovane, cez cocoon: protokol.
Dátovod Engine	SAX základ	Áno	Áno
	Caching	Áno	Áno
Webové Aplikácie	Tok stránok	Áno, deklaratívny XML tok s plnou separáciou formy, modelu pohľadu, akcie a prezentácie v rámci kontrolera toku stránok.	Áno, s Cocoon Control Flow skriptovaním (JavaScript).
	XForms podpora	Áno, zabudovaná implementácia na strane servera kompletne integrovaná s kontrolerom toku stránok.	Áno, pomocou XMLForms alebo Chicoon. Cocoon teraz používa jeho vlastný engine (Cocoon Forms).
	EJB Podpora	Áno, pomocou delegation procesora môžu byť session beans volané bez písania Java kódu. EJBs môžu byť rovnako volané aj z Java kódu použitím Java Processoru, vlastných XML komponentov a JSP.	Písaním Java kódu v XSP, akcií alebo vlastných komponentov.

Jazyky	Java	Áno, pomocou Java Processoru alebo vlastných XML komponentov.	Áno, pomocou XSP alebo vlastných komponentov.
	XSLT 1.0	Áno, zabudovaný Xalan, XSLTC a Saxon.	Ano, zabudovaný Xalan, XSLTC a Saxon.
	XSLT 2.0	Áno, zabudovaný	Možné pomocou TrAX.
	XQuery	Áno, zabudovaný	Nedokumentované
	XUpdate	Áno, zabudovaný	Nedokumentované
	JSP	Áno, v Modeli 2X konfigurácii alebo použitím Servlet Include Generatora.	Áno. Použitím Cocoon akcií a XSP.
	XSP	Nie.	Áno.
Databázová Podpora	SQL	SQL Processor.	SQL Transformer, SQL logicsheet.
	LDAP	LDAP Processor.	LDAP Transformer.
	XML:DB	Áno	Áno
Iné	Web Služby	Áno	Áno
	Plánovanie	Áno	Áno
	WebDAV	Áno, len na čítanie	Áno
	Operácie príkazového riadku	Áno	Áno
	Zapuzdrené operácie	Áno	Je to možné.
	Open Source	Áno, LGPL.	Áno, Apache licencia.

Príloha C Technická dokumentácia k prototypu

Ukážka kódu flowscriptu pre zobrazenie formulára s inštanciami ponúk a následné volanie dátovodu na zobrazenie vybranej ponuky.

```
cocoon.load("resource://org/apache/cocoon/forms/flow/javascript/Form.js");
function test(form)
{
	var instance  = new Packages.hachiban.JobOfferPortal.support.GenForm.OntologyExaminer();
	var list = instance.getClassInstances
		("http://nazou.fiit.stuba.sk/nazou/ontologies/v0.6.15/offer-job#JobOffer");
	var xml = new Packages.hachiban.JobOfferPortal.support.GenForm.
		GenForm.GenerateSelectionList(list);
	var file = new java.io.FileWriter
		("webapps/cocoon/prototype/models/offerListChoices.xml");
	var printWriter = new java.io.PrintWriter(file);

	printWriter.print(xml);
	printWriter.flush();
	printWriter.close();
	form.showForm("offerList-display-pipeline");
	cocoon.sendPage("offer-display");
}
```

Ukážka kódu flowscriptu pre naviazanie údajov z fomulára do objektu Java Bean, následné uloženie do ontologického úložiska a zobrazenie nasledovného formulára.

```
cocoon.load("resource://org/apache/cocoon/forms/flow/javascript/Form.js");
function offer2bean(form)
{
	var offerid = cocoon.request.getParameter("offerList");
	java.lang.System.out.println("OfferID:" + offerid);
	var repository = new	Packages.nazou.support.jobofferportal.repository.SesameRepositoryAccess
		("http://147.175.98.225:8080/sesame", "DomainOntology", "user", "password");
	repository.connect();
	var offer = repository.downloadOffer(offerid);

	form.load(offer);
	form.showForm("offer-display-pipeline");
	form.save(offer);
	repository.uploadOffer(offer);
	cocoon.sendPage("test.jx", {"result" : "Ponuka bola uspesne zapisana."} );
}
```

Ukážka časti modelu formulára pre zobrazenie inštancie ponuky.

```
  <fd:widgets>
<!-- SALARY CLASS -->
	<fd:class id="salary">
		<fd:widgets>
			<fd:field id="SalaryMinAmount" required="true">
				<fd:label>Plat od</fd:label>
				<fd:datatype base="string"/>
				<fd:validation>
					<fd:regexp pattern="[0-9]*">
						<fd:failmessage>Musi obsahovat len cisla!
						</fd:failmessage>
					</fd:regexp>
				</fd:validation>
```

```
                        </fd:field>
                        <fd:field id="SalaryMaxAmount" required="true">
                                <fd:label>Plat do</fd:label>
                                <fd:datatype base="string"/>
                                <fd:validation>
                                        <fd:regexp pattern="[0-9]*">
                                                <fd:failmessage>Musi obsahovat len cisla!
                                                </fd:failmessage>
                                        </fd:regexp>
                                </fd:validation>
                        </fd:field>
                        <fd:field id="SalaryBonus" required="false">
                                <fd:label>Odmeny</fd:label>
                                <fd:datatype base="string"/>
                        </fd:field>
                        <fd:field id="SalaryText" required="false">
                                <fd:label>Text</fd:label>
                                <fd:datatype base="string"/>
                        </fd:field>
                </fd:widgets>
        </fd:class>
<!-- OFFER CLASS -->
        <fd:class id="offer">
                <fd:widgets>
                        <fd:field id="OfferName" required="true">
                                <fd:datatype base="string"/>
                                <fd:label>Pracovna pozicia</fd:label>
                                <fd:validation>
                                        <fd:length min="2">
                                        <fd:failmessage>Musi mat aspon 2 znaky!!</fd:failmessage>
                                   </fd:length>
                                </fd:validation>
                        </fd:field>
                        <fd:new id="organization"/>
                        <fd:new id="salary"/>

                        <fd:repeater id="benefits">
                          <fd:widgets>
                                <fd:output id="id">
                                  <fd:datatype base="long"/>
                                </fd:output>
                                <fd:new id="benefit"/>
                                <fd:booleanfield id="select">
                                  <fd:label>Select</fd:label>
                                </fd:booleanfield>
                          </fd:widgets>
                        </fd:repeater>
                        <fd:repeater-action id="addbenefit" command="add-row" repeater="benefits">
                          <fd:label>Add benefit</fd:label>
                        </fd:repeater-action>
                        <fd:repeater-action id="removebenefits" command="delete-rows"
                repeater="benefits" select="select">
                          <fd:label>Remove selected benefits</fd:label>
                        </fd:repeater-action>
                </fd:widgets>
        </fd:class>
        <fd:new id="offer"/>
  </fd:widgets>
</fd:form>
```

Ukážka časti šablóny pre zobrazenie formulára pracovnej ponuky.

```
<?xml version="1.0"?>
<html xmlns:ft="http://apache.org/cocoon/forms/1.0#template"
  xmlns:fi="http://apache.org/cocoon/forms/1.0#instance"
  xmlns:i18n="http://apache.org/cocoon/i18n/2.1"
  xmlns:jx="http://apache.org/cocoon/templates/jx/1.0">

<jx:import uri="resource://org/apache/cocoon/forms/generation/jx-macros.xml"/>

<head>
  <title>Offer form</title>
</head>
<body>
   <ft:form-template action="#{$cocoon/continuation/id}.continue" method="POST">
         <!-- SALARY CLASS -->
               <ft:class id="salary">
                     <tr>
                           <td valign="top"><ft:widget-label
                           id="SalaryMinAmount"/>:</td>
                           <td valign="top"><ft:widget id="SalaryMinAmount"/></td>
                     </tr>
                     <tr>
                           <td valign="top"><ft:widget-label
                           id="SalaryMaxAmount"/>:</td>
                           <td valign="top"><ft:widget id="SalaryMaxAmount"/></td>
                     </tr>
                     <tr>
                           <td valign="top"><ft:widget-label id="SalaryBonus"/>:</td>
                           <td valign="top"><ft:widget id="SalaryBonus"/></td>
                     </tr>
                     <tr>
                           <td valign="top"><ft:widget-label id="SalaryText"/>:</td>
                           <td valign="top"><ft:widget id="SalaryText"/></td>
                     </tr>
               </ft:class>

         <!-- OFFER CLASS -->
               <ft:class id="offer">
                     <table border="1">
                           <tr>
                             <td valign="top"><ft:widget-label id="OfferName"/>:</td>
                             <td valign="top"><ft:widget id="OfferName"/></td>
                           </tr>
                           <ft:new id="organization"/>
                           <ft:new id="salary"/>
                     </table>

                     <ft:repeater id="benefits">
                           <table border="1">
                                 <tbody>
                                       <tr>
                                             <th>Benefit</th>
                                             <th><ft:repeater-widget-label widget-
                                             id="select"/></th>
                                       </tr>
                                       <ft:repeater-rows>
                                             <tr class="forms-row-
                                             ${repeaterLoop.index % 2}">
                                                   <td><ft:new id="benefit"/></td>
```

```
                                                    <td align="center"><ft:widget
                                                    id="select"/></td>
                                                </tr>
                                            </ft:repeater-rows>
                                        </tbody>
                                    </table>
                                </ft:repeater>
                                <tr>
                                    <td colspan="2" align="right">
                                        <ft:widget id="addbenefit"/>
                                        <ft:widget id="removebenefits"/>
                                    </td>
                                </tr>
                            </ft:class>
                            <ft:new id="offer"/>
                            <br/>
                            <input type="submit"/>
    </ft:form-template>
  </body>
</html>
```

Ukážka definície mapovania formulára na objekt Java Bean alebo XML súbor pomocou Cocoon Binding Framework.

```
<?xml version="1.0"?>

<fb:context xmlns:fb="http://apache.org/cocoon/forms/1.0#binding" path="/" >
        <fb:value id="OfferName" path="offerName"/>
        <fb:value id="offeredByProfile" path="offeredByProfile"/>
        <fb:value id="offeredByName" path="offeredByName"/>

        <fb:repeater id="benefits" parent-path="." row-path="benefit">
                <fb:identity>
                        <fb:value id="id" path="@id"/>
                </fb:identity>

                <fb:on-bind>
                        <fb:value id="benefit-text" path="text"/>
                        <fb:value id="benefit-class" path="type"/>
                </fb:on-bind>

                <fb:on-delete-row>
                        <fb:delete-node/>
                </fb:on-delete-row>

                <fb:on-insert-row>
                        <fb:insert-bean
                                classname="nazou.support.jobofferportal.model.Benefit"
                                addmethod="addBenefit"/>
                </fb:on-insert-row>
        </fb:repeater>
</fb:context>
```

Príloha D Technická dokumentácia k produktu

D.1 Integrácia formulárov rámca CForms do portálu

D.1.1 Štýly a skripty

Rámec CForms používa pri svojej práci množstvo JavaScript knižníc a css štýlov. Pri jednoduchom použití formulárov mimo portálu sú tieto knižnice a štýly pre používateľa transparentné, pretože sa pridávajú do HTML kódu stránky transformáciou pred samotným serializovaním stránky a odoslaním klientovi.

V prípade portálového riešenia a použitia CForms formulárov v coplete však klasický (a prototypovaný) postup tvorby formulárov neprináša želaný efekt. Formuláre sú načítané bez štýlov a JavaScriptov. To spôsobuje nefunkčnosť tabiek, pop-up kalendára atď. Pre ich správnu funkcionalitu je potrebné pridať do šablóny *portal/skins/<skin_name>/styles/ portal-page.xsl* vo vzore <xsl:template match="/"> v obsahu elementu *head* nasledovné riadky:

```
<link title="Default Style" href="/cocoon/styles/main.css" rel="stylesheet"/>
<script type="text/javascript" src="resources/forms/js/forms-lib.js"/>
<script type="text/javascript" src="resources/ajax/js/cocoon-ajax.js"/>
<script type="text/javascript" src="resources/forms/js/cforms.js"/>
<link href="{$base}css/forms.css" type="text/css" rel="stylesheet"/>
<script type="text/javascript" src="resources/forms/mattkruse-
lib/AnchorPosition.js"/>
<script type="text/javascript" src="resources/forms/mattkruse-
lib/PopupWindow.js"/>
<script type="text/javascript" src="resources/forms/mattkruse-
lib/OptionTransfer.js"/>
<script type="text/javascript" src="resources/forms/mattkruse-
lib/selectbox.js"/>
<script xmlns:i18n="http://apache.org/cocoon/i18n/2.1" type="text/javascript"
src="resources/forms/mattkruse-lib/CalendarPopup.js"/>
<script type="text/javascript" src="resources/forms/mattkruse-lib/date.js"/>
<script type="text/javascript">
// Setup calendar
var forms_calendar = CalendarPopup();
forms_calendar.setWeekStartDay(1);
forms_calendar.showYearNavigation();
forms_calendar.showYearNavigationInput();
forms_calendar.setCssPrefix("forms_");
</script>
<link href="resources/forms/css/forms-calendar.css" type="text/css"
rel="stylesheet"/>
<script type="text/javascript">
_editor_url = "resources/forms/htmlarea/";
_editor_lang = "en";
 </script>
<script src="resources/forms/htmlarea/htmlarea.js" type="text/javascript"/>
```

To zabezpečí korektné zobrazovanie formulárov v rámci copletov portálu.

D.1.2 Funkcionalita AJAX vo formulároch

Rámec CForms pri použití mimo portálu poskytuje možnosť asynchrónnej komunikácie so serverom pomocou technológie AJAX. V našom riešení AJAX využívame pri zobrazovaní

prvkov typu *TreeWidget*, alebo pri zaškrtávacom prvku *startDateASAP* (v prípade, že sa zvolí táto možnosť, znemožní sa vypísanie dátumu do poľa pre dátum nástupu).

Využitie technológie AJAX vo formulároch, ktoré sa zobrazujú v copletoch si vyžaduje nasledovné úpravy:

V šablóne formulára (*portal/coplets/AjaxJobOffer/forms/offer_template.xml*) je potrebné upraviť riadok s akciami formulára nasledovne:

```
<ft:form-template action="#{$cocoon/continuation/id}.continuation"
method="POST" ajax="true" ajax-
action="/cocoon/portal/coplets/AjaxJobOffer/#{$cocoon/continuation/id}.continu
e">
```

Následne sú potrebné úpravy sitemap-y (*portal/coplets/AjaxJobOffer/sitemap.xmap*):

Do časti *map:transformers* sa pridá riadok:

```
<map:transformer name="browser-update"
src="org.apache.cocoon.ajax.BrowserUpdateTransformer"/>
```

Do časti map:selectors sa pridá riadok:

```
<map:selector name="ajax-request"
src="org.apache.cocoon.ajax.AjaxRequestSelector"/>
```

Samotné zobrazenie formulára sa potom vykoná nasledovným dátovodom:

```
<map:match pattern="*-display-pipeline">
  <map:generate type="jx" src="forms/{1}_template.xml">
    <map:parameter name="locale" value="{flow-attribute:locale}"/>
  </map:generate>
  <map:transform type="browser-update"/>
  <map:transform type="i18n">
    <map:parameter name="locale" value="{flow-attribute:locale}"/>
  </map:transform>
  <map:call resource="simple-page2html">
    <map:parameter name="file" value="forms/{1}_template.xml"/>
  </map:call>
  <map:transform src="resources/forms-styling.xsl"/>
  <map:select type="ajax-request">
    <map:when test="true">
      <map:serialize type="xml"/>
    </map:when>
    <map:otherwise>
      <map:serialize type="html"/>
    </map:otherwise>
  </map:select>
</map:match>
```

D.2 Integrácia existujúcich aplikácií do portálu

Portálové riešenie Cocoonu umožňuje integrovať teoreticky ľubovoľnú webovú aplikáciu vo forme copletu do vytváraného portálu. Pre vytvorenie nového copletu je potrebné zadefinovať ho ako typ, vytvoriť inštanciu copletu a zadefinovať jeho umiestnenie na stránke portálu. My sme túto možnosť Cocoonu využili a vytvorili portál z viacerých relatívne samostatných aplikácií, ktoré sa (s výnimkou portálových nástrojov *Tools*) integrovali do portálu vo forme copletov a navzájom prepojili pomocou flowscriptu a zadefinovaných dátovodov.

Postup integrácie aplikácie do portálu si ukážeme na príklade nástroja Factic do portálu.

D.2.1 Zadefinovanie copletu

Cocoon obsahuje viacero základných typov copletov. Každý coplet musí mať pevne stanovenú URI, ktorú zobrazuje.
Pre integráciu aplikácie tretej strany do portálu je výhodne použiť *CachingUriCoplet*. Ten má uri pevne nastavenú na „dummy“ aplikáciu a má ďalší atribút *temporary:application-uri*, ktorým je nasmerovaný na skutočnú aplikáciu, ktorej rozhranie zobrazuje. *CachingUriCoplet* má, ako už názov napovedá, vyrovnávaciu pamäť. Pri obnove stránky portálu (po vykonanej akcii) sa tak v prípade, že sa akcia daného copletu netýka nemusí volať aplikácia, ale použije sa výstup z vyrovnávacej pamäte copletu. Ak by sa musela aplikácia zakaždým znovu vyvolať, mohlo by to spôsobiť neželané stavy aplikácie a teda nekorektné správanie.
Ukážka definície copletu Factic v súbore *portal/profiles/copletdata/portal.xml*:

```
<coplet-data id="Factic" name="standard">
  <title>Faceted Semantic Browser</title>
  <coplet-base-data>CachingURICoplet</coplet-base-data>
    <attribute>
      <name>buffer</name>
      <value xsi:type="java:java.lang.Boolean">true</value>
    </attribute>
    <attribute>
      <name>handleParameters</name>
      <value xsi:type="java:java.lang.Boolean">true</value>
    </attribute>
    <attribute>
      <name>uri</name>
      <value xsi:type="java:java.lang.String">cocoon:/coplets/html/application
  </value>
    </attribute>
    <attribute>
      <name>temporary:application-uri</name>
      <value
xsi:type="java:java.lang.String">cocoon://portal/coplets/Factic/factic
  </value>
    </attribute>
</coplet-data>
```

D.2.2 Vytvorenie inštancie copletu

Po zadefinovaní copletu je potrebné vytvoriť jeho inštanciu s jedinečným identifikátorom. Teoreticky je možné vytvoriť viacero inštancií jedného copletu.
Ukážka vytvorenia inštancie Factic-1 copletu Factic v súbore : *portal/profiles/copletinstancedata/portal.xml*. Je dôležité si uvedomiť, že v tomto okamihu začíname rozlišovať inštancie copletov pre prihláseného a neprihláseného (anonymného používateľa). Inštancie copletov pre neprihláseného používateľa sa nachádzajú v súbore *portal-user-anonymous.xml.*

```
<coplet-instance-data id="Factic-1" name="standard">
<coplet-data>Factic</coplet-data>
</coplet-instance-data>
```

D.2.3 Zadefinovanie umiestnenia copletu

Vytvorenú inštanciu copletu je potrebné umiestniť do *layoutu* portálu. Portálové riešenie cocoon v tejto časti umožňuje coplety ľubovoľne zoskupovať, vytvárať rôzne menu a pod. Rozmiestnenie copletov sa definuje v súbore: *portal/profiles/copletinstancedata/portal.xml* (resp. *portal-user-anonymous.xml* pre neprihláseného používateľa).

V nasledujúcej ukážke je zobrazená časť tohto súboru, v ktorej sa definuje lišta s jednotlivými položkami menu, ktorým sú priradené príslušné inštancie copletov. Coplety sú definované v štýle *"nowindow"*, takže nebudú mať okraje a nebudú mať ovládacie prvky na ich minimalizáciu, maximalizáciu a zatvorenie.

```
<item>
  <composite-layout name="column">
  <item>
    <parameter name="width" value="85%"/>
    <composite-layout id="tab" name="tab">
      <named-item name="Add Offer">
        <parameter name="width" value="70%"/>
    <coplet-layout name="coplet" layout-renderer-name="nowindow">
          <coplet-instance-data>AjaxJobOffer-1</coplet-instance-data>
    </coplet-layout>
     </named-item>
     <named-item name="Factic">
      <coplet-layout name="coplet" layout-renderer-name="nowindow">
        <coplet-instance-data>Factic-1</coplet-instance-data>
      </coplet-layout>
    </named-item>
    <named-item name="About">
      <coplet-layout name="coplet" layout-renderer-name="nowindow">
        <coplet-instance-data>About-1</coplet-instance-data>
  </coplet-layout>
    </named-item>
    <named-item name="Help">
      <coplet-layout name="coplet" layout-renderer-name="nowindow">
        <coplet-instance-data>Help-1</coplet-instance-data>
      </coplet-layout>
    </named-item>
  </composite-layout>
</item>
          </composite-layout>
    </item>
```

Po tejto konfigurácii sa coplet zobrazí na stránke portálu a dá sa s ním pracovať.

D.3 Prepojenie aplikácií navzájom

Pre účely nášho riešenia sme potrebovali prepojiť dve aplikácie – nástroj *Factic* slúžiaci na prehľadávanie ponúk a základnú prácu s nimi (zmazanie, deaktivovanie) a aplikáciu na zobrazenie formulára. Aplikácie sme prepojili na úrovni cocoon sitemap a flowscriptu. Nástroj *Factic* má v rozhraní zadefinované tlačidlo, ktoré smeruje na lokálne URL *edit*, pričom v parametroch je posielaný celý aktuálny stav nástroja. URL *edit* sa namapuje na tento dátovod:

```
<map:match pattern="edit">
  <map:redirect-to uri="cocoon://portal/coplets/AjaxJobOffer/offer.form"/>
</map:match>
```

Uvedený dátovod iba jednoducho presmeruje požiadavku do druhej aplikácie, ktorá má zobraziť formulár. V ňom sa požiadavka namapuje na tento dátovod:

```
<map:match pattern="*.form">
    <map:call function="handleForm">
      <map:parameter name="function" value="anyFormDisplay"/>
      <map:parameter name="definitionURI" value="forms/{1}_model.xml"/>
      <map:parameter name="bindingURI" value="bindings/{1}_bind_bean.xml"/>
      <map:parameter name="renderMode" value="{1}"/>
      <map:parameter name="contextPath" value="{request:contextPath}"/>
    </map:call>
</map:match>
```

Uvedený dátovod zabezpečí spustenie obslužného flowscriptu. Spúšťa sa interná funkcia cocoonu *handeForm*, ktorej sa predá ako parameter model formulára, pravidlá na prepojenie formulára s objektom Java bean. Funkcia *handleForm* zabezpečí vytvorenie objektu formulára, ktorý predá funkcii *anyFormDisplay*, ktorej názov dostane ako parameter. Táto funkcia využíva aj ďalšie parametre – kontextovú cestu a mód spustenia. Ten slúži na rozlíšenie druhu formulára, ktorý zobrazujeme, keďže vzor daného dátovodu je písaný všeobecne (**.form*), aby sa dokázal namapovať na zobrazenie ponuky (*offer.form*) alebo formulárov registrácie (*organization.form*). Skutočná hodnota namiesto znaku '*' je namapovaná na sekvenciu „{1}".

Ďalšie spracovanie požiadavky je teda v réžii funkcie anyFormDisplay:

```
function anyFormDisplay(form) {
var map = new Packages.java.util.HashMap();
var e = cocoon.request.getParameterNames();
while(e.hasMoreElements() )
{
var el = e.nextElement();
map.put(el, cocoon.request.getParameter(el));
}
var contextPath = cocoon.parameters["contextPath"];
var renderMode = cocoon.parameters["renderMode"];
```

V prvej časti funkcie sa spracujú všetky parametre získané z pôvodného requestu nástroja *Factic* a dodatočné parametre získané zo sitemapy. Pristupuje k nim pomocou objektu *cocoon*.

V ďalšej časti sa pripravuje obsah formulára a pripojenie na ontologické úložisko pomocou triedy *SesameRepositoryAccess*. Rozlišujeme stav, keď bola metóda volaná nástrojom *Factic* (premenná *isFactic*) a bude sa editovať existujúca ponuka alebo sa vytvára úplne nová ponuka prípadne organizácia.

Konštruktoru triedy *SesameRepositoryAccess* sa ako parameter predá kontextová cesta k portálu a názov súboru, v ktorom sa nachádzajú pravidlá pre objektovo ontologický mapovač.

```
var isFactic = false;
var FormBindingBean = null;
var repository = new
Packages.sk.fiit.nazou.jop.crud.repository.SesameRepositoryAccess("./webapps"
+ contextPath + "/portal","GraphBeanTransformerMetadata.xml");

repository.connect();
var copletid = map.get("copletid");//id of a coplet which called this function

if(copletid != null && copletid.equals("Factic-1"))
{
//we were called by factic ==> we edit an offer
    isFactic = true;
```

```
    var namespace = map.get("namespace"); //ns of ontology individual
    var uri = map.get("uri"); //localName of ontology individual
    FormBindingBean = repository.downloadOffer(namespace+"#"+uri);
}
else
{
//we were not called by factic ==> new offer or registration
    if(renderMode.equals("offer"))
    {
           //we create an empty offer bean
           FormBindingBean = new
Packages.sk.fiit.nazou.jop.crud.model.JobOffer();
    }
    else if(renderMode.equals("organization"))
    {
           //we create an empty organization bean
           FormBindingBean = new
Packages.sk.fiit.nazou.jop.model.Organization();
    }
    else
    {
           //weird renderMode - we create an empty offer bean
           FormBindingBean = new Packages. sk.fiit.nazou.jop.model.JobOffer();
    }
}
```

V tomto momente máme vytvorený alebo naplnený objekt Java bean. Môžeme ho prepojiť s formulárom a zobraziť.

```
form.load(FormBindingBean);
form.showForm(renderMode + "-display-pipeline");
form.save(FormBindingBean);
```

Pre zobrazenie samotného formulára sa volá dátovod *-display-pipeline, opísaný v kap. A.1.2. Toto volanie zablokuje ďalšie vykonávanie funkcie až kým nie je na server odoslaný korektne vyplnený formulár, ktorý prejde všetkými validátormi. Následne sa formulár uloží späť do objektu Java bean.

Zmeny sa musia uložiť do ontologického úložiska:

```
if(renderMode.equals("organization"))
{
//if we render organization form we have to recreate repository with different
metadata file
repository = new
Packages.sk.fiit.nazou.jop.crud.repository.SesameRepositoryAccess("./webapps"
+ contextPath + "/portal","OrganizationMetadata.xml");
    repository.connect();
}
//upload of offer (or organization)
var offerid = repository.uploadOffer(FormBindingBean);
repository.disconnect();
```

V nasledujúcej časti sa v prípade, že bola funkcia volaná nástrojom *Factic* resetne vyrovnávacia pamäť copletu a *temporary-uri* atribút sa nastaví späť na pôvodný stav *Factic*-u (to znamená poslať naspäť všetky parametre, ktoré *Factic* poslal pri volaní a pridať parameter *message* s výsledkom akcie). Parametre *Factic*-u sú uložené v mape vytvorenej na začiatku vykonávania funkcie.

Ak by sa coplet neresetoval, ostala by mu v pamäti už neplatná *temporary-uri*, čo by spôsobilo pád copletu. Resetovanie copletu je realizované pomocou procedúry *resetCoplet*, ktorej sa ako parameter zadá meno inštancie copletu a URI, ktoré sa má nastaviť.

V tejto časti sa po uložení ontologických dát pristupuje k uloženiu neontologických údajov o ponuke do HSQL databázy. Pre samotné uloženie sa volá pomocná procedúra *storeHSQLData*.

```
if(isFactic)
{
var message = "Offer with id: " + offerid + " was successfully uploaded!";
var FacticState = "/portal/coplets/Factic/factic?message=" + message + "&";

var keySetIterator = map.keySet().iterator();
//we iterate through all request parameters and pass it back to Factic where
it came from
while(keySetIterator.hasNext())
{
var key = keySetIterator.next();
    FacticState +=key;
    FacticState += "=";
    FacticState += map.get(key);
    if(keySetIterator.hasNext())
    {
           FacticState +="&";
    }
}
resetCoplet("Factic-1", "cocoon:/" + FacticState);
storeHSQLData(Packages.java.lang.Boolean.TRUE, offerid, contextPath);
cocoon.sendPage(FacticState);
}
```

V prípade, že funkcia nebola volaná nástrojom Factic, ostávajú stále dve možnosti: vypĺňala sa nová ponuka alebo sa jedná o registráciu.

Ak sa vypĺňala nová ponuka, zobrazíme správu o úspešnom uložení. Následne musíme resetovať coplet pre ponuku tak ako v prípade copletu *Factic*. Ak by sme to nespravili, používateľ by neustále videl iba hlásenie o úspešnom uložení a nevedel by zadať ďalšiu novú ponuku.

```
else
{
  if(renderMode.equals("offer"))
  {
     //we were not called by factic but render mode is offer
 cocoon.sendPage("success.jx", {"result" : offerid } );
     if(copletid != null)
     {
       storeHSQLData(Packages.java.lang.Boolean.FALSE, offerid, contextPath);
       resetCoplet("AjaxJobOffer-1",
"cocoon://portal/coplets/AjaxJobOffer/offer.form");
       //we reset also the factic coplet, to ensure new offer is displayed in
factic
       resetCoplet("Factic-1", "cocoon://portal/coplets/Factic/factic");
     }
  }
```

V prípade, že sa jedná o registráciu, je potrebné zobraziť ešte jeden formulár (stále ten istý dátovod ako všetky ostatné formuláre), v ktorom používateľ vyplní údaje o sebe. Dáta z tohto formulára treba následne uložiť ako atribúty nového používateľa, ktorý bude zatiaľ neaktívny.

Predtým však treba skontrolovať už existujúce loginy v systéme. V prípade, že si používateľ zvolil login, ktorý je už obsadený, je potrebné vyznačiť to ako chybu vo formulári a nepustiť ďalšie spracovávanie.

```
  else if(renderMode.equals("organization"))
  {
    var form = new Form("./forms/user_model.xml");
    var authMan =
cocoon.getComponent("org.apache.cocoon.webapps.authentication.AuthenticationMa
nager");
    var child = callAuthPipeline(authMan, "load-users");
    var resource = getChildUri(child);
    var params = getChildParams(child);
    params.setSingleParameterValue("type", "users");
    var doc = loadResource(resource, params);
    var obj = new Packages.java.util.ArrayList();
    createKeys(doc, 0, obj);
    var flag = 0;
    while(flag == 0)
    {
      form.showForm("user-display-pipeline");
      var login = form.getChild("login").getValue();
      print("login: " + login);
      flag = 1;
      for(var i = 0; i < obj.size(); i++)
      {
        if(obj.get(i).getItemWithKey("name").getValue().equals(login))
        {
          var error = new
Packages.org.apache.cocoon.forms.validation.ValidationError("login_already_in_
use", true);
          form.getChild("login").setValidationError(error);
          flag = 0;
        }
      }
    }
    //login is validated
    child = callAuthPipeline(authMan, "new-user");
    var newUserResource = getChildUri(child);
    params = getChildParams(child);

    params.setSingleParameterValue("type", "user");
    params.setSingleParameterValue("ID", login);
    params.setSingleParameterValue("company", offerid);
    params.setSingleParameterValue("active", "false");
    params.setSingleParameterValue("role", "orgadmin");
    params.setSingleParameterValue("password",
form.getChild("password").getValue());
    params.setSingleParameterValue("firstname",
form.getChild("FirstName").getValue());
    params.setSingleParameterValue("lastname",
form.getChild("LastName").getValue());
    params.setSingleParameterValue("email",
form.getChild("email").getValue());

    cocoon.releaseComponent(authMan);
    invokeResource(newUserResource, params);
//we display an empty registration form
cocoon.sendPage("/portal/coplets/AjaxJobOffer/organization.form");
```

```
//we reset coplet back to its original state
resetCoplet("Registration-1",
"cocoon://portal/coplets/AjaxJobOffer/organization.form");
}//end else if(renderMode.equals("organization"))
} //end else (was not invoked by Factic
} //function end
```

Pri vytváraní a editovaní ponúk je potrebné ukladať aj neontologické dáta o ponuke – čas, kedy bola naplnená, či je aktívna a kto ju naplnil. Procedúra *storeHSQLData* si zistí aktuálneho prihláseného používateľa a vytvorí, prípadne aktualizuje záznam týkajúci sa ponuky s daným URI.

```
function storeHSQLData(isUpdate, offerid, contextPath)
{
  var login = "unknown";
  var contextMan =
cocoon.getComponent(Packages.org.apache.cocoon.webapps.session.ContextManager.
ROLE);
  var context = contextMan.getContext("authentication");
  var authenticationManager =
cocoon.getComponent("org.apache.cocoon.webapps.authentication.AuthenticationMa
nager");
    /* get current user data */
    var context = authenticationManager.getState().getHandler().getContext();
    var grabber = new
Packages.org.apache.cocoon.portal.tools.userManagement.ContextGrabber();
    var userBean = grabber.grab(context);
    login = userBean.getContextItem("name");
    var date = new Packages.java.util.Date();
    var timeStamp = new Packages.java.sql.Timestamp(date.getTime());
    var isActive = new Packages.java.lang.Boolean("false");

    var OfferTableAdapter = new
Packages.sk.fiit.nazou.jop.hsql.OfferTableAdapter("./webapps" + contextPath +
"/portal/DB/OfferDB", "sa", "");
    var NotOntoBean = null;
    if(isUpdate.equals(Packages.java.lang.Boolean.TRUE))
    {
      //performing update
  NotOntoBean = OfferTableAdapter.getOffer(offerid);
      if(NotOntoBean == null)
      {
        //we perform an update, but there is no entry for such offerid
        NotOntoBean = new Packages.sk.fiit.nazou.jop.hsql.Offer();
        isActive = Packages.java.lang.Boolean.TRUE;
       }

    }
    else
    {
          //performing create
          NotOntoBean = new Packages.sk.fiit.nazou.jop.hsql.Offer();
          isActive = Packages.java.lang.Boolean.FALSE;
    }
    /* setting bean fields */
    NotOntoBean.setUri(offerid);
    NotOntoBean.setAuthor(login);
    NotOntoBean.setCreationTime(timeStamp);
    NotOntoBean.setIsActive(isActive);
```

```
    try
    {
           OfferTableAdapter.tryUpdate(NotOntoBean);
           OfferTableAdapter.close();
    }
    catch (e)
    {
           print("exception occured: " + e);
    }
}
```

Metóda *resetCoplet* zabezpečí vyresetovanie vyrovnávacej pamäti copletu, nastavenie application uri na zadanú hodnotu a prešírenie tejto zmeny:

```
function resetCoplet(copletId, tempURI)
{
  var service = null;
  try
  {
    // get portal service/profile manager/coplet
    service =
cocoon.getComponent(org.apache.cocoon.portal.PortalService.ROLE);
    service.setPortalName("portal");
    var componentManager = service.getComponentManager();
    var profileManager = componentManager.getProfileManager();
    var coplet = profileManager.getCopletInstanceData(copletId);

//reset the coplet's cache and temporary:application-uri
    coplet.setTemporaryAttribute(org.apache.cocoon.portal.coplet.adapter.impl.
CachingURICopletAdapter.DO_NOT_CACHE,"true");
    coplet.setTemporaryAttribute("application-uri", tempURI);

    //create an event to make changes visible :) (magic)
    var path = "temporaryAttributes/application-uri";
    var value = tempURI;
    var event = new
Packages.org.apache.cocoon.portal.event.impl.CopletJXPathEvent(coplet, path,
value);
    service.getComponentManager().getEventManager().getPublisher().publish(eve
nt);
}
catch ( e )
{
       cocoon.log.info("Error", e);
    }
    finally {
       cocoon.releaseComponent( service );
    }
}
```

D.3.1 Spracovanie prístupových práv vo Facticu

Do fazetového prehliadača Factic sme doplnili podporu pre spracovanie prístupových práv používateľov pomocou XSL šablóny, ktorá ako parameter získa údaje o prihlásenom používateľovi. Na základe týchto údajov vyhodnotí prístup používateľa k jednotlivým inštanciám ponúk a upraví používateľské rozhranie na zobrazenie vybraných ovládacích prvkov (napr. nezobrazí voľbu edituj).

Ukážka XSLT šablóny na získanie údajov o prihlásenom používateľovi:

```
<!-- Root Factic element. -->
<xsl:template match="factic">
  <xsl:copy>
    <xsl:copy-of select ="@*"/>
    <xsl:element name ="user">
      <login>
        <session:getxml context="authentication" path="/authentication/ID"/>
      </login>
      <role>
        <session:getxml context="authentication"
path="/authentication/data/role"/>
      </role>
      <organization>
        <session:getxml context="authentication"
path="/authentication/data/company"/>
      </organization>
    </xsl:element>

  <xsl:apply-templates/>
  </xsl:copy>
</xsl:template>

<!-- This template simply copies stuff that doesn't match other -->
<!-- templates and applies templates to any children.           -->
<xsl:template match="@*|node()" priority="-1">
 <xsl:copy>
  <xsl:apply-templates select="@*|node()"/>
 </xsl:copy>
</xsl:template>
```

Získané údaje o prihlásenom používateľovi sú ďalej využité pri spracovaní výstupu z jadra Facticu, ktorý obsahuje inštancie nájdených ponúk práce. Pomocou šablóny XSLT sa do výstupu pridávajú ovládacie prvky, resp. sa ich pridanie preskočí na základe role, ktorú má prihlásený používateľ pridelenú v systéme. Administrátor systému má prístup ku všetkým ponuká, administrátor organizácie ku ponukám z vlastnej organizácie, bežný používateľ len ku svojim ponukám, neprihlásený používateľ nemá prístup k operáciám so žiadnymi ponukami.

```
<!-- Retrieve user credentials - access rights. -->
<xsl:variable name="user"
select="attributes/attribute/label[node()='User']/../value"/>
<xsl:variable name="organization"
select="attributes/attribute/label[node()='Company']/../value"/>

<!-- Proces user roles - access rights. -->
<xsl:choose>
  <!-- Role: admin ==> all item actions -->
  <xsl:when test="//user/role[node()='admin']">
    <xsl:call-template name="ItemActions"/>
  </xsl:when>

  <!-- Role: orgadmin ==> all item actions -->
  <xsl:when test="//user/role[node()='orgadmin']">
    <xsl:choose>
      <!-- This check should be made against the organization URI, which is
not yet available so we check against the organization's name. -->
      <xsl:when test="//user/organization[node()=$organization]">
```

```
        <xsl:call-template name="ItemActions"/>
      </xsl:when>
      <xsl:when test="//user/login[node()=$user]">
        <xsl:call-template name="ItemActions"/>
      </xsl:when>
      <xsl:otherwise>
        <td/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:when>

  <!-- Role: user ==> all item actions -->
  <xsl:when test="//user/role[node()='user']">
    <xsl:choose>
      <xsl:when test="//user/login[node()=$user]">
        <xsl:call-template name="ItemActions"/>
      </xsl:when>
      <xsl:otherwise>
        <td/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:when>

  <!-- Role: other ==> no item action -->
  <xsl:otherwise>
    <xsl:if test="//user/login[not(node()='anonymous')]">
      <td/>
    </xsl:if>
  </xsl:otherwise>
</xsl:choose>
```

D.4 Manažment používateľov

Portálové riešenie Cocoon obsahuje podporu manažmentu používateľov. Nanešťastie táto podpora nie je zatiaľ plne implementovaná a preto bolo potrebné pridať chýbajúce časti.

D.4.1 Konfigurácia

Všetky operácie nad používateľmi a rolami sú vykonávané pomocou dátovodov, ktoré sú definované v autentifikačnom manažéri ako konfigurácia „single-role-user-management“. Názvy týchto dátovodov sú presne stanovené (*new-user* pre vytvorenie nového užívateľa a pod.). Príklad zápisu takejto konfigurácie znázorňuje nasledujúca časť sitemapy.

```
<map:component-configurations>
  <authentication-manager>
    <handlers>
      <handler name="portal-handler">
      <redirect-to uri="cocoon:/login"/>
      <authentication uri="cocoon:raw:/sunrise-authuser"/>
        <applications>
          <application loadondemand="true" name="portal">
            <configuration name="single-role-user-management">
              <load-users uri="cocoon:raw:/sunrise-loaduser"/>
              <load-roles uri="cocoon:raw:/sunrise-roles"/>
              <new-user uri="cocoon:raw:/sunrise-newuser"/>
              <new-role uri="cocoon:raw:/sunrise-newrole"/>
´             <change-user uri="cocoon:raw:/sunrise-changeuser"/>
```

```
              <delete-role uri="cocoon:raw:/sunrise-delrole"/>
              <delete-user uri="cocoon:raw:/sunrise-deluser"/>
            </configuration>
```

Hore uvedené dátovody sú konfigurované nasledovne:

```
<map:match pattern="sunrise-changeuser">
    <map:generate src="resources/sunrise-changeuser.xml"/>
    <map:transform type="session"/>
    <map:transform type="cinclude"/>
    <map:transform src="styles/changeuser.xsl"/>
    <map:transform type="write-source"/>
    <map:transform src="styles/portal.xsl"/>
    <map:serialize type="xml"/>
</map:match>
<map:match pattern="sunrise*">
    <map:generate src="resources/sunrise{1}.xml"/>
    <map:transform type="session"/>
    <map:transform type="cinclude"/>
    <map:transform type="write-source"/>
    <map:transform src="styles/portal.xsl"/>
    <map:serialize type="xml"/>
</map:match>
```

Bližší opis autentifikačného rámca je uvádza domovská stránka Cocoonu[8]. Naše riešenie pracuje nad užívateľmi a rolami uložených v XML súboroch a operácie nad nimi sú vykonávané pomocou XSLT transformácií. Nasledujúca ukážka znázorňuje zápis používateľov v XML súbore:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<authentication>
    <users>
          <user>
                <name>cocoon</name>
                <password>cocoon</password>
                <role>admin</role>
                <title>Mr.</title>
                <firstname>Walter</firstname>
                <lastname>Cocoon</lastname>
                <company>testCompanyURI</company>
                <phone>N/A</phone>
                <fax>N/A</fax>
                <email>N/A</email>
                <active>true</active>
          </user>
    </users>
```

[8] Cocoon Authentication Framework, http://cocoon.apache.org/2.1/developing/webapps/authentication.html

Nad týmto súborom pracujú spomínané dátovody. Ako generátor pre dátovod *load-users* slúži nasledujúci XML súbor.

```
<?xml version="1.0"?>
<loaduser xmlns:session="http://apache.org/cocoon/session/1.0"
          xmlns:cinclude="http://apache.org/cocoon/include/1.0">
    <info>
          <ID><session:getxml context="request" path="/parameter/ID"/></ID>
          <role><session:getxml context="request"
path="/parameter/role"/></role>
          <type><session:getxml context="request"
path="/parameter/type"/></type>
          <company><session:getxml context="request" path
="/parameter/company"/></company>
    </info>
    <cinclude:includexml ignoreErrors="true">
          <cinclude:src>resources/sunrise-user.xml</cinclude:src>
    </cinclude:includexml>
</loaduser>
```

Následne sa volá XSLT transformácia, ktorá vykoná potrebné úpravy (čítanie, zmena, zápis) do XML súboru s používateľmi a rolami. Vybranú časť tejto transformácie, pre načítanie používateľov, uvádzame ako príklad:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="loaduser">
    <xsl:apply-templates select="authentication/users"/>
</xsl:template>

<xsl:template match="authentication/users">
    <users>
          <xsl:for-each select="user">
                <xsl:call-template name="includeuser"/>
          </xsl:for-each>
    </users>
</xsl:template>

<xsl:template name="includeuser">
    <xsl:variable name="type"><xsl:value-of select="normalize-
space(ancestor::loaduser/info/type)"/></xsl:variable>
    <xsl:variable name="name"><xsl:value-of select="normalize-
space(ancestor::loaduser/info/ID)"/></xsl:variable>
    <xsl:variable name="role"><xsl:value-of select="normalize-
space(ancestor::loaduser/info/role)"/></xsl:variable>
    <xsl:variable name="company"><xsl:value-of select="normalize-
space(ancestor::loaduser/info/company)"/></xsl:variable>

    <xsl:choose>
          <xsl:when test="normalize-space(role) = $role and ($type='users' or
normalize-space(name) = $name)">
                <xsl:message terminate="no">bol som tu - role and
user</xsl:message>
                <user>
                      <ID><xsl:value-of select="name"/></ID>
                      <role><xsl:value-of select="role"/></role>
                      <data>
```

```
                              <password><xsl:value-of
select="password"/></password>
                              <name><xsl:value-of select="name"/></name>
                              <role><xsl:value-of select="role"/></role>
                              <ID><xsl:value-of select="name"/></ID>
                              <user><xsl:value-of select="name"/></user>
                              <title><xsl:value-of select="title"/></title>
                              <firstname><xsl:value-of
select="firstname"/></firstname>
                              <lastname><xsl:value-of
select="lastname"/></lastname>
                              <company><xsl:value-of
select="company"/></company>
                              <phone><xsl:value-of select="phone"/></phone>
                              <fax><xsl:value-of select="fax"/></fax>
                              <email><xsl:value-of select="email"/></email>
                              <active><xsl:value-of select="active"/></active>
                        </data>
                  </user>
            </xsl:when>
            <xsl:when test="$type='user' and $name = normalize-space(name)">
                  <xsl:message terminate="no">bol som tu - orgURI</xsl:message>
                  <user>
                        <ID><xsl:value-of select="name"/></ID>
                        <role><xsl:value-of select="role"/></role>
                        <data>
                              <password><xsl:value-of
select="password"/></password>
                              <name><xsl:value-of select="name"/></name>
                              <role><xsl:value-of select="role"/></role>
                              <ID><xsl:value-of select="name"/></ID>
                              <user><xsl:value-of select="name"/></user>
                              <title><xsl:value-of select="title"/></title>
                              <firstname><xsl:value-of
select="firstname"/></firstname>
                              <lastname><xsl:value-of
select="lastname"/></lastname>
                              <company><xsl:value-of
select="company"/></company>
                              <phone><xsl:value-of select="phone"/></phone>
                              <fax><xsl:value-of select="fax"/></fax>
                              <email><xsl:value-of select="email"/></email>
                              <active><xsl:value-of select="active"/></active>
                        </data>
                  </user>
            </xsl:when>
            <xsl:when test="$type='usersFromOrg' and normalize-space(company) =
$company">
                  <xsl:message terminate="no">bol som tu - orgURI</xsl:message>
                  <user>
                        <ID><xsl:value-of select="name"/></ID>
                        <role><xsl:value-of select="role"/></role>
                        <data>
                              <password><xsl:value-of
select="password"/></password>
                              <name><xsl:value-of select="name"/></name>
                              <role><xsl:value-of select="role"/></role>
                              <ID><xsl:value-of select="name"/></ID>
                              <user><xsl:value-of select="name"/></user>
```

```
                              <title><xsl:value-of select="title"/></title>
                              <firstname><xsl:value-of
select="firstname"/></firstname>
                              <lastname><xsl:value-of
select="lastname"/></lastname>
                              <company><xsl:value-of
select="company"/></company>
                              <phone><xsl:value-of select="phone"/></phone>
                              <fax><xsl:value-of select="fax"/></fax>
                              <email><xsl:value-of select="email"/></email>
                              <active><xsl:value-of select="active"/></active>
                        </data>
                  </user>
            </xsl:when>
            <xsl:when test="$type='users' and string-length($role) = 0">
                  <xsl:message terminate="no">bol som tu - all
users</xsl:message>
                  <user>
                        <ID><xsl:value-of select="name"/></ID>
                        <role><xsl:value-of select="role"/></role>
                        <data>
                              <password><xsl:value-of
select="password"/></password>
                              <name><xsl:value-of select="name"/></name>
                              <role><xsl:value-of select="role"/></role>
                              <ID><xsl:value-of select="name"/></ID>
                              <user><xsl:value-of select="name"/></user>
                              <title><xsl:value-of select="title"/></title>
                              <firstname><xsl:value-of
select="firstname"/></firstname>
                              <lastname><xsl:value-of
select="lastname"/></lastname>
                              <company><xsl:value-of
select="company"/></company>
                              <phone><xsl:value-of select="phone"/></phone>
                              <fax><xsl:value-of select="fax"/></fax>
                              <email><xsl:value-of select="email"/></email>
                              <active><xsl:value-of select="active"/></active>
                        </data>
                  </user>
            </xsl:when>
    </xsl:choose>
</xsl:template>

<xsl:template match="@*|node()">
    <xsl:copy>
          <xsl:apply-templates select="@*|node()" />
    </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

D.4.2 Volanie dátovodov

Keďže rámec Cocoon nemá kompletne implementovanú správu používateľov, je potrebné volať spomenuté dátovody ručne. Nasledovná funkcia v JavaScripte zavolá rúru s názvom *resource* a predá jej parametre *parameters*.

```
function invokeResource(resource, parameters)
{
var resolver = cocoon.getComponent(
"org.apache.excalibur.source.SourceResolver");
var source = org.apache.cocoon.components.source.SourceUtil.getSource(
resource,
null,
parameters,
resolver);

org.apache.cocoon.components.source.SourceUtil.parse(
new org.apache.avalon.framework.service.DefaultServiceManager(),
    source,
    new org.xml.sax.helpers.DefaultHandler);

resolver.release(source);
cocoon.releaseComponent(resolver);
}
```

V niektorých prípadoch však okrem zavolania dátovodu potrebuje získať aj jeho výstup (napr. zoznam používateľov). Tento problém rieši nasledujúca funkcia:

```
function loadResource(resource, parameters) {
    var source = null;
    try {
           var resolver = cocoon.getComponent(
                  "org.apache.excalibur.source.SourceResolver");
           source = org.apache.cocoon.components.source.SourceUtil.getSource(
             resource,
             null,
             parameters,
             resolver);
           return org.apache.cocoon.components.source.SourceUtil.toDOM(source);
    }
    finally {
           if(source != null) {
                  resolver.release(source);
           }
           cocoon.releaseComponent(resolver);
    }
}
```

Aby bolo možné dané funkcie zavolať, je potrebné najprv zistiť predávané parametre. Parameter *resource* získame pomocou nasledujúcich dvoch funkcií:

```
function callAuthPipeline(authMan, pipeline) {
    var state = authMan.getState();
    var conf = state.getModuleConfiguration("single-role-user-management");
    return conf.getChild(pipeline, false);
}

function getChildUri(child) {
    if (child != null) {
        return child.getAttribute("uri");
    }
    return null;
}
```

Funkcia *callAuthPipeline* dostane na vstup inštanciu autentifikačného manažéra a meno požadovaného dátovodu (napr. *load-users*). Následne, vráti objekt *child*, ktorý v sebe obsahuje informácie o dátovode. Funkcia *getChildUri* dostane na vstup práve tento objekt a vráti úplnú cestu k dátovodu, ktorú môžeme použiť pri volaní funkcií *invokeResource* a *loadResource*.

Dátovody pracujú so zoznamom parametrov, ktoré je možné podľa potreby nakonfigurovať. Časť z nich môže byť povinná a predvyplnená v objekte *child*. Množinu predvyplnených atribútov získame nasledujúcou funkciou, ktorej vstupom je objekt *child*:

```
function getChildParams(child) {
    var newUserResource = null;
    var params = new org.apache.excalibur.source.SourceParameters();
        if (child != null) {
            var newUserResourceParameters =
                  org.apache.excalibur.source.SourceParameters.create(child);
           if (newUserResourceParameters != null)
           {
                  params.add(newUserResourceParameters);
           }
    }
    return params;
}
```

D.4.3 Operácie nad používateľmi

Logika operácií nad používateľmi (pridanie používateľa, zmena profilu a pod.) je realizovaná pomocou Cocoon JavaScriptu. Na nasledujúcom príklade opíšeme funkciu obsluhujúcu editovanie profilu. Volania *params.setSingleParameterValue()* nastavujú parametre, ktoré sa odovzdajú dátovodu *load-users*. V uvedenom príklade dátovod na získanie informácií o používateľovi dostane ako parameter *ID* aktuálne prihláseného používateľa.

```
function showUser() {
    var obj = getContext();
    var authMan = cocoon.getComponent(
        "org.apache.cocoon.webapps.authentication.AuthenticationManager");
    var child = callAuthPipeline(authMan, "load-users");
    cocoon.releaseComponent(authMan);

    var form = new Form("cocoon:/page/model/userData?mode=edit2");

    var resource = getChildUri(child);
    var params = getChildParams(child);
    params.setSingleParameterValue("type", "user");
    params.setSingleParameterValue("ID", obj.getContextItem("name"));

    var doc = loadResource(resource, params);
    var users = new Packages.java.util.ArrayList();
    createKeys(doc, 0, users);

    for(var it = obj.getContext().iterator(); it.hasNext();) {
           var e = it.next();
           var item = users.get(0).getItemWithKey(e.getKey());
           if(item == null) {
                  users.get(0).addContext(e.getKey(), "");
           }
    }
```

```
    users.get(0).addContext("password2", ""); //for retyping

    var form = new Form("cocoon:/page/model/userData?mode=profile");

    form.createBinding("cocoon:/page/binding/userData?mode=profile");
    form.load(users.get(0));

    if(obj.getContextItem("role") != "admin") {
           var repeater = form.getChild("context");
           var widget = getWidgetInRepeater(repeater, "company");

           widget.setState(
             Packages.org.apache.cocoon.forms.formmodel.WidgetState.DISABLED);
           widget = getWidgetInRepeater(repeater, "active");
           widget.setState(
             Packages.org.apache.cocoon.forms.formmodel.WidgetState.DISABLED);
           widget = getWidgetInRepeater(repeater, "role");
           widget.setState(
             Packages.org.apache.cocoon.forms.formmodel.WidgetState.DISABLED);
           widget = getWidgetInRepeater(repeater, "name");
           widget.setState(
             Packages.org.apache.cocoon.forms.formmodel.WidgetState.DISABLED);
    }

    form.showForm("page/form/userData?mode=profile");
    form.save(users.get(0));
    changeUser(users.get(0), null);
    if(obj.getContextItem("role") == "user") {
           cocoon.sendPage("profile-success-user.jx");
           return;
    }
    else {
           cocoon.sendPage("profile-success.jx");
           return;
    }
}
```

Rovnako je dôležité si všimnúť volanie metódy *createKeys()*, ktoré je nutné z toho dôvodu, že metóda *loadResource()* vracia fragment XML súboru. V uvedenom prípade je to informácia o aktuálne prihlásenom používateľovi (metóda môže vrátiť napr. všetkých používateľov zaregistrovaných v portáli). Metóda *createKeys()* naplní zoznam používateľov, pričom každý záznam je typu *sk.fiit.nazou.jop.userManagement.UserBean* a je dynamicky vygenerovaný. Následne user beany aj formulár, v tomto prípade na editovanie profilu, sú vždy dynamicky generované na základe položiek, ktoré sú zadefinované v XML súbore s používateľmi.

```
function createKeys (node, lev, obj) {
    node = node.getFirstChild();
    var nodes = node.getChildNodes();
    for(var i = 0; i < nodes.getLength(); i++) {
           var bean = new Packages.sk.fiit.nazou.jop.userManagement.UserBean();
           var node = nodes.item(i).getFirstChild();
           var childnodes =
                node.getNextSibling().getNextSibling().getChildNodes(); //data
           for(var j = 0; j < childnodes.getLength(); j++) {
                  if(childnodes.item(j).getFirstChild()  != null)
                  {
                         bean.addContext(
```

```
                childnodes.item(j).getNodeName(),
                childnodes.item(j).getFirstChild().getNodeValue());
            }
        }
        obj.add(bean);
    }
}
```

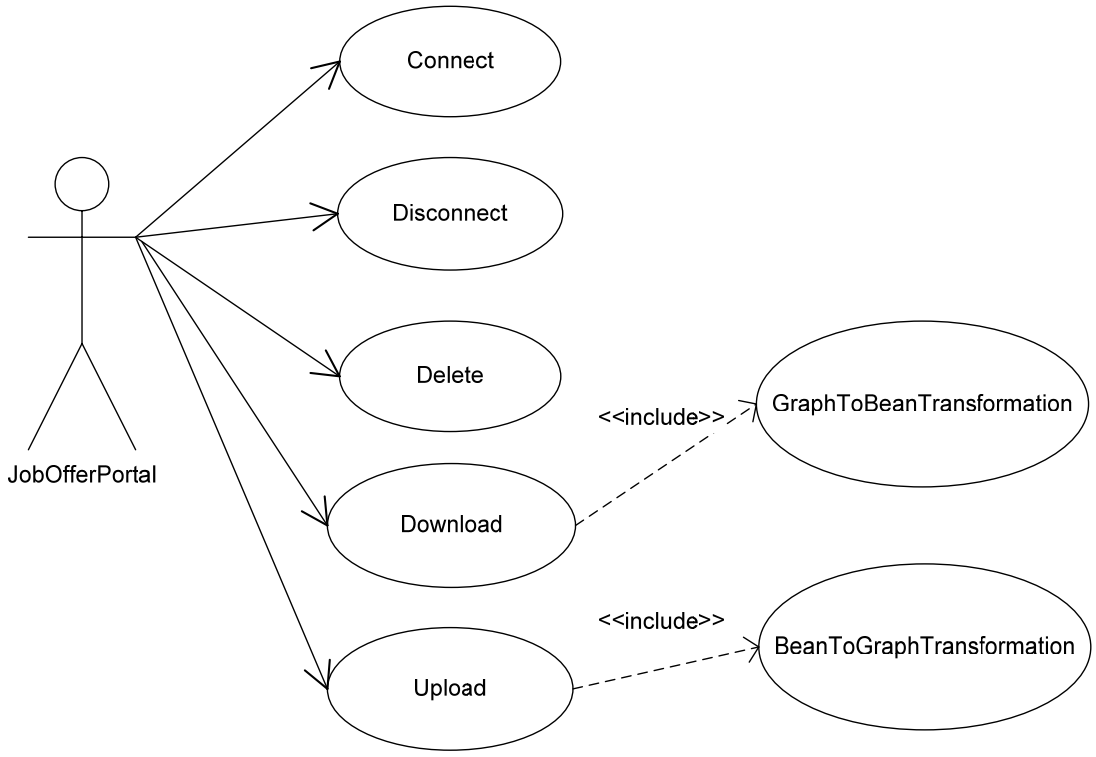
D.5 Realizácia vzoru CRUD pre ponuku práce

Jednou z dôležitých čŕt portálu, je realizácia vzoru CRUD (create, retrieve, update, delete)[9]. Ide o klasické operácie s úložiskom, realizované nad pracovnými ponukami. Na uchovávanie dát je použité ontologické úložisko Sesame (http://www.openrdf.com/). Jednou zo súčastí realizácie vzoru CRUD je mapovanie objektov Java Bean, predstavujúcich vnútornú reprezentáciu ponúk v systéme, na inštancie v ontológii.

Objekty predstavujú inštancie tried Java Bean vytvorené pre potreby vnútornej reprezentácie entít v systéme, kde umožňujú uchovávanie informácií o pracovnej ponuke. Tieto objekty mapujeme na RDF graf, predstavujúci ontologickú reprezentáciu ponuky a uchovávame v ontologickom úložisku.

D.5.1 Prípady použitia

K základným prípadom použitia patrí vytvorenie a zrušenie spojenia s úložiskom. Ďalšie realizujú vzor CRUD – uloženie, stiahnutie a vymazanie ponúk z úložiska. Uloženie a stiahnutie v sebe zahŕňajú príslušnú transformáciu medzi objektovou a ontologickou reprezentáciou.

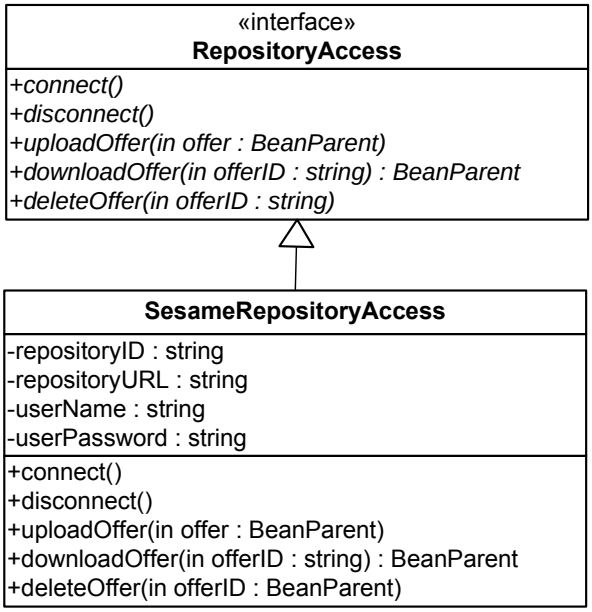


Obrázok 1. Prípady použitia vzoru CRUD v systéme.

[9] Barla, M. et. al.: Dynamic Ontology Based Form Generation for Portal Solutions. In: Bieliková, M. (ed.): Proc. of IIT.SRC 2006: Student Research Conference, apríl 2006, Bratislava.

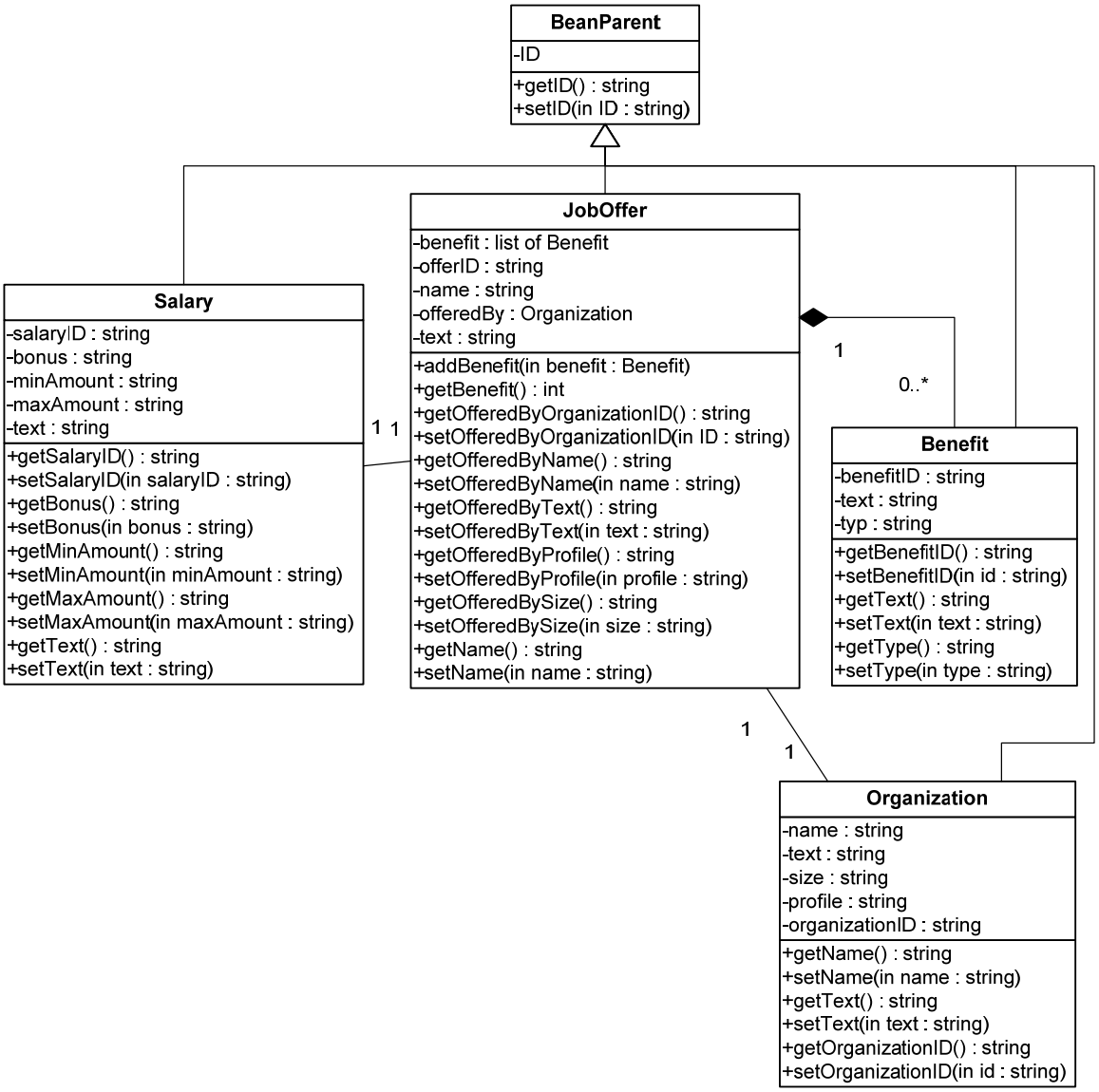
D.5.2 Prístup k úložisku

Na poskytovanie požadovanej funkcionality sme vytvorili balík *sk.fiit.nazou.jop.crud.repository*. Obsahuje triedy poskytujúce API pre realizovanie všetkých potrebných operácií nad úložiskom (pozri Obrázok 2). Nachádza sa v ňom rozhranie *RepositoryAccess*, ktoré musia implementovať všetky triedy pracujúce s konkrétnym úložiskom. Rozhranie obsahuje funkcie na pripojenie a odpojenie sa k/od úložiska, pridanie, získavanie a odobratie dát. V súčasnosti predpokladáme len prístup k úložisku Sesame. Následne sme implementovali len jednu triedu, implementujúca toto rozhranie – *SesameRepositoryAccess*.



Obrázok 2. API pre operácie nad úložiskom.

Objekty, s ktorými vytvorené API pracuje sú inštancie Java Bean tried. Pre potreby tímového projektu sme vytvorili triedy (pozri Obrázok 3), ktoré sú súčasťou balíka *sk.fiit.nazou.jop.crud.model*. Na obrázku nie sú znázornené všetky vytvorené triedy. Pre celkové pokrytie dát v ponuke a taxonómie určitých jej atribútov obsiahnutých v ontológii, bolo potrebné vytvoriť desiatky takýchto tried. Dôležité je však vedieť, že každá trieda je zdedená od rozhrania *BeanParent*. API umožňuje pracovať s inštanciami tried, ktoré sú zdedené od tohto rozhrania.



Obrázok 3. Hierarchia Java Bean tried pre pracovnú ponuku.

Pri práci s ontologickým úložiskom využívame Sesame API verzie 1.2.4. Ide o sadu balíkov, umožňujúcich prácu s ontologickým úložiskom Sesame. Na pripájanie a odpájanie sa k/od úložiska využívame funkcie *connect()* a *disconnect()*, ktoré využívali toto API. Pre konfiguráciu parametrov pripojenie sme používali Java properties súbor. Obsahuje URL a ID úložiska, login používateľa. Na jeho načítania slúži trieda *PropertyReader*.

Na samotnú realizáciu vzoru CRUD boli implementované funkcie *uploadOffer(), downloadOffer(), deleteOffer()*. Realizujú uloženie objektu Java Bean do úložiska, získanie objektu Java Bean pre konkrétnu ponuku z úložiska na základe jej ID (URI) a napokon vymazanie niektorej ponuky z úložiska.

Funkcia *uploadOffer()* v sebe zahŕňa volanie funkcie transformujúcej objekt Java Bean do RDF grafu. Funkcia *downloadOffer()* naopak obsahuje volanie transformácie RDF grafu do objektu Java Bean. Obe transformácie sú však realizované transparentne. Pri prechodoch z jednej do druhej transformácie sú nevyhnutné určité metadáta, ktoré hovoria o previazaní jednotlivých premenných v Java Bean triedach a príslušnou vlastnosťou ontologických tried. Metadáta uchovávame vo forme XML súboru. Nachádzajú sa v ňom pravidlá pre jednotlivé previazania. Na základe týchto metadát môžeme čiastočne konfigurovať postup mapovania. Pravidlá obsahujú:

1. *BeanFieldName* – názov premennej v Java Bean triede (k premenným objektov agregovaných tried sa dostávame pomocou bodkovej notácie)
2. *OntologyPropertyURI* – predikát z ontológie pre príslušnú vlastnosť
3. *Type* – typ vlastnosti (či ide o dátový alebo objektový typ, v prípade dátového typu je uvedený konkrétny typ)
4. *Multiple* – multiplicita, ktorá hovorí o tom, či môže byť daná vlastnosť viacnásobná
5. *Recursive* – rekurzívnosť, ktorá hovorí, či majú mapovacie algoritmy postupovať rekurzívne
6. *RecursivelyDelete* – vymazávanie, ktoré určuje, či je potrebný pri vymazávaní rekurzívny postup
7. *DoubleRange* – multiplicita cieľovej triedy, ktorá určuje, či daný predikát môže smerovať do viacerých rôznych tried, alebo len do jednej
8. *BeanName* – názov Java Bean triedy, ktorá reprezentuje príslušnú ontologickú triedu

Príklad jedného mapovacieho pravidla je nasledovný:

```
<MappingRule>
    <BeanFieldName>.name</BeanFieldName>
    <OntologyPropertyURI>http://nazou.fiit.stuba.sk/nazou/ontologies/v0.6.16/o
ffer#name</OntologyPropertyURI>
    <Type>string</Type>
    <Multiple>false</Multiple>
    <Recursive>false</Recursive>
    <RecursivelyDelete>false</RecursivelyDelete>
    <DoubleRange>false</DoubleRange>
    <BeanName></BeanName>
</MappingRule>
```

Pre načítavanie týchto metadát, bola vytvorená trieda *GraphBeanTransformerMetadataXMLReader*. Na načítanie XML súboru používame štandard W3C DOM (http://www.w3.org/DOM/).

D.5.3 Ontologicko-objektové mapovače

Samotné mapovanie medzi objektami Java Bean tried a RDF grafom, a spracovanie metadát zabezpečujú triedy *BeanGraphTransformer, GraphBeanTransformerMetadataXMLReader* a *OfferFieldMetadataObject* (pozri Obrázok 4). Na realizáciu samotných transformácií slúžia metódy *transformBeanToGraph()* a *transformGraphToBean()* v triede *BeanGraphTransformer* .

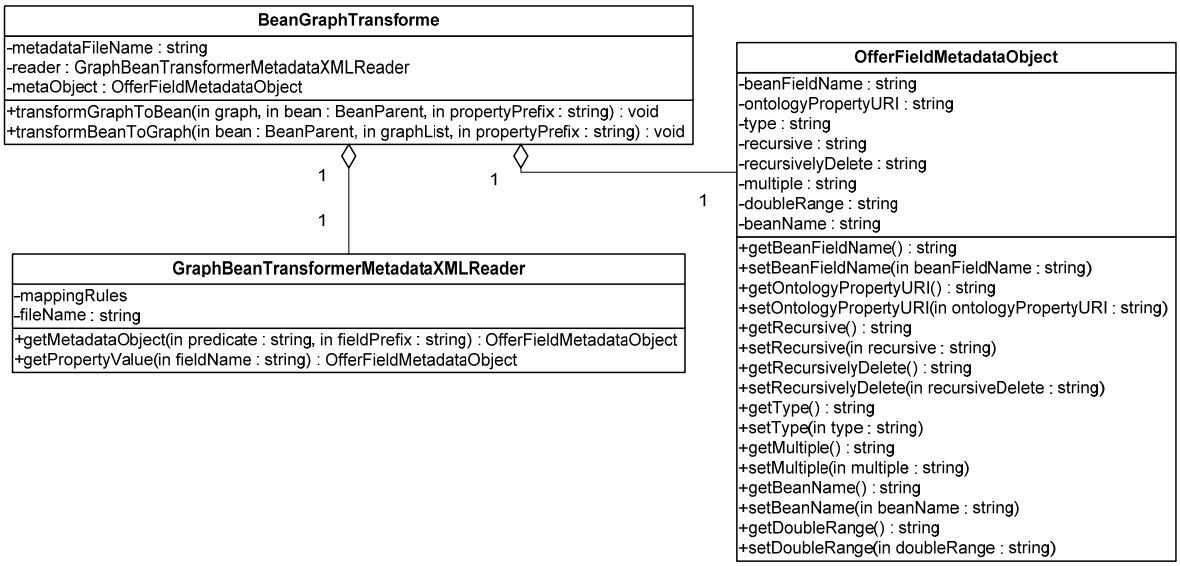
Metóda *transformBeanToGraph()* vytvára na základe naplnených atribútov Java Bean triedy (uvedenej ako parameter) postupne v rekurzívnom volaní grafy jednotlivých objektových vlastností a pridáva ich do zoznamu, ktorý je vstupným parametrom funkcie.

Metóda *transformGraphToBean()* napĺňa atribúty objektu Java Bean tried (parameter) na základe dát z RDF grafu (parameter).

Obe transformácie sú rekurzívne. V jednom kroku rekurzie sa vykonáva mapovanie medzi jedným objektom Java Bean tried a inštanciou ontologickej triedy. Transformácie sa spúšťajú na inštancie tried predstavujúcich vrchol v hierarchii (inštancia triedy JobOffer pri objektovej reprezentácii a inštancia http://nazou.fiit.stuba.sk/nazou/ontologies/v0.6.16/offer-job#JobOffer v prípade ontologickej triedy). Pri prístupe k atribútom objektov Java Bean tried sa využíva reflekcia.

Na čítanie XML súboru obsahujúceho metadáta, predstavujúce pravidlá potrebné pri transformáciách slúži trieda *GraphBeanTransformerMetadataXMLReader*. Informácie o jednom pravidle sú v pamäti uchovávané v podobe inštancie triedy *OfferFieldMetadataObject*. Trieda *GraphBeanTransformerMetadataXMLReader* obsahuje metódy na získavanie týchto objektov. Keďže viacero tried môže obsahovať atribút s rovnakým názvom, ale pritom majú určité odlišnosti, je ich potrebné v metadátach odlišovať. Príkladom je názov ponuky a názov organizácie. Na odlíšenie je použitá bodková notácia. Ku každému atribútu sa dostaneme cez triedu, v ktorej sa nachádza, začínajúc od triedy na najvyššej úrovni (jej názov sa však vynecháva).

Názov organizácie, ktorá ponúka prácu je potom určená reťazcom „*isOfferedBy.name*“. Prefixy týchto reťazcov (podreťazec pred poslednou bodkou) sú potrebné aj ako parametre funkcií vykonávajúcich transformáciu pri jednotlivých volaniach. Samotné prefixy sú však vytvárané automaticky. Prvotné volanie transformácií sa realizuje s týmto parametrom nastaveným na prázdny reťazec.



Obrázok 4. Triedy realizujúce mapovanie

Príloha E Článok na ISD 2006

Príloha obsahuje článok, ktorý sme vypracovali a zaslali na konferenciu ISD 2006.

Ontology as an Information Base for a Family of Domain Oriented Portal Solutions

No Author Given

No Institute Given

Abstract. Ontologies are becoming increasingly accepted as a form for information representation of distributed web-based information and its processing. In our paper we build a web-based application around an ontology, which serves both as a data base and metadata source used in processing of the data. We propose techniques and methods that exploit this metadata to provide an easy and flexible implementation of the CRUD pattern. We identify patterns in the ontological representation of domain entities and transform them into web-based forms for data management. Based on this flexible framework we can model variations to adapt the application for specific sub-domains.

1 Introduction

A significant amount of resources has already been invested into the development of web-based applications that use ontologies for data storage [6]. Among other advantages, ontologies support the execution of semantic queries on an ontological database, explicit and inherent access to metadata and reasoning based on the formal representation of both the data and metadata.

By definition, an ontology is an explicit formal specification of a shared conceptualization (of a domain) [4]. Ontological languages provide means for the basic modeling of concepts such as classes, relations and various properties. Based on the corresponding language constructs, domain specific concepts can be defined and their semantics specified. Consequently, the model of a domain consists of both concepts defined by ontological languages, which are reused in other domain models and domain-specific concepts unique to a particular domain or set of domains.

If we create a model of a specific domain represented by an ontology, this model will necessarily have to be changed at some time in the future to compensate for changing requirements. Ontologies are especially suited for such change and provide flexible means of data storage.

However, the flexible data storage provided by ontologies would be of limited use unless applications are flexible enough to accommodate a similar

degree of change. Otherwise, for any change in the ontology, the corresponding application code would have to be manually updated, which is unpractical for real-world applications.

Thus the apparent flexibility of ontologies results in the need for flexible applications. If we consider forms as the main mean of communication between portal and its users we come to the need for flexible form generation tools. These would be based not only on data stored in ontologies but also on additional (meta)data, which would be needed because domain ontologies shall not contain the information about the desired visual data representation, preserving their generocity and delegating application-specific information to other sources.

2 Proposed approach

In our approach, we take advantage of the native availability of metadata in ontologies, which make the data self-descriptive and allows for effective searching in the stored data. Based on the assumption that an ontology may represent a formal model of an information domain, we propose the use of such metadata to build a domain oriented web portal solution for a particular domain.

In order to process ontologically structured data by the state of the art GUI frameworks, we implement mapping tools that transform data between its graph representation and an object-oriented representation. Consequently, we define the mapping of modeling concepts between these two modeling paradigms.

To design a flexible framework, we identify patterns as repeating structures (sets of concepts and their relations) in ontological representations and define consecutive data and processing around these patterns. We assume that similar patterns are shared between ontologies with the patterns themselves being defined using different types of ontological concepts – classes, relations between them, properties, instances and restrictions.

We map identified patterns onto sets of visual elements (graphical user interface widgets) that correspond to the data stored in a specific ontology. Based on this approach we implement the CRUD pattern (Create-Retrieve-Update-Delete) for a particular entity in a domain ontology.

Since the metadata available in ontologies are not always sufficient to fully create a satisfactory user interface, we define additional metadata, which describe the arrangement of visual components on web-based forms.

Finally, we automatically generate the forms corresponding to ontological concepts, where first the respective ontological patterns are identified. Next, the proper graphical representation is determined and lastly the form descriptions are saved and used during operation.

3 Object–ontology mapping

The creation of an object-oriented representation of ontological concepts introduces new challenges due to the fundamental differences of both representations. These come from the differences between description logic and object-oriented systems and lie primarily in the completeness and satisfiability. Ontologies have a significantly higher expressivity compared to object-oriented approaches [3].

We automatically generate a set of java bean classes, each corresponding to an integral part of an entity described by the ontology. We represent literals by a simple data type field of the appropriate type and each object property by a separate java bean.

In order to process these java beans and store the values in an ontological repository we need to generate additional metadata for the mapping between objects and classes and properties (RDF graphs) of the ontology. These metadata allow us to bind the ontological class to java bean fields and include the name of the java bean, the names of its fields, information about the corresponding OWL properties (e.g. multiplicity, data type) and the type of the object in the RDF triple (object type or a data type – literal).

The mapping itself is performed by a pair of graph-to-bean and bean-to-graph transformers (analogical to O/R mapping in relational databases [2]). These transformers work the graph representation of RDF and use reflection to invoke *get* and *set* methods on the generated java bean objects.

The transformation process is performed recursively, in each step performs mapping between one object and an integral part of the RDF graph, i.e. object property. Finally, a simple java bean field is mapped to an RDF triple and vice versa, with the name of the field being used to determine the corresponding ontological property. Thus a simple data type field corresponds to a literal in the ontology and an object property corresponds to an instance of the respective java bean class.

4 Pattern types

We identified two distinct types of patterns that can be applied at different levels of abstraction and are thus useful for a broad set of ontologies:

- *Widget patterns*, which are based on basic ontological language concepts and correspond to relatively simple configurations in ontologies.
- *Visual patterns*, which define the higher-level visual style of forms, the layout of individual widgets and other form controls (close to user interface design).

To achieve independence from the used ontology we base our metadata processing on basic ontological language concepts. Moreover, we specify a model for the representation of variability in information sub-domains, e.g.

specializing the structure and behavior of an instance of the CRUD pattern of a domain entity for some specified subset of users.

4.1 Widget patterns

Widget patterns focus on the structure of classes, subclasses, properties and possible restrictions. They determine the widgets used in a form to edit instances of classes.

4.2 Simple widget patterns

Primitive datatype properties

Patterns for *primitive datatype* properties include all properties, whose ranges are literals (string, integer, float, date, boolean etc.). The graphical representation of these patterns is straightforward – the *rdfs:label* of each property is displayed next to the input field for its value. This would generally be a dropdown list for Boolean values (true, false, undefined), a calendar for date values and a *textbox* for text strings. It is more convenient to use *textarea* input field for longer strings as it increases the readability of the form. To distinguish cases where a normal *textbox* and where a *textarea* should be used, we can either define additional metadata about desired form template or we can compare the average length of existing instances to a predefined threshold.

If the respective property has multiple cardinality, the above elements would be placed inside a special control called *repeater*, which enables users to add/remove more values by means of additional buttons for these actions.

Same-range object properties

Patterns for *same-range object properties* identify classes that have several object properties with multiple cardinality and the same range (class or a union of classes). In this case, the fields for the range are displayed only once and an additional component is used to distinguish the specific property that is being edited. Such a component can be either a dropdown list or a set of radio buttons. All these components are wrapped in a repeater to allow multiple values to be added.

For example, the *Prerequisite* class form the Job Offer domain ontology of the NAZOU [5] project. Each prerequisite has two multiple properties: the *Requires* and *Prefers* which have as their range a union of experience and qualification classifications. Thus, the visual representation provides a radio button to select between the *requires* and *prefers* properties while the rest of the widget is common for both properties and is used to assign experience and qualification prerequisites for job candidates.

Enumerations

Patterns for *enumerations* identify object properties, whose range is a class fully defined by its instances. It should not be possible to add new or edit existing instances. Enumerated classes are defined in the ontology itself, e.g. a class representing the days of the week or various time periods (hour, day, month etc.).

Several graphical representations of this pattern exist. If the cardinality of an object property whose range is an enumerated class is single then it can be represented by a dropdown list of its instances. If the cardinality is multiple, the mentioned dropdown list can be wrapped in a repeater or instances can be represented by a multi-choice *listbox*.

Besides enumerated classes, also classes which allow users to create new instances or choose an existing one can be identified. This information must be stored in the metadata for the appropriate class. In this case all fields necessary to create an instance would be displayed and a dropdown list or *listbox* with existing instances would be added as mentioned above.

4.3 Tree hierarchies

Since hierarchies offer a wide range of values, they must be structured in a way that allows users to easily understand and choose amongst them. For example, when users want to choose a country where a company is based, it might be convenient for them to first choose a continent, then a country on that continent, etc.

In ontology, there are two basic ways to represent tree hierarchies. The standard property *rdfs:subclassOf* between classes which represents an *is a* relation and/or custom defined properties between instances can be used, which define relationships between nodes in the tree hierarchy.

One can assume that if a class in an ontology has a property which is transitive and points to instances of the same class (its range and domain are the same), then it is used to represent some form of hierarchy. The job offer ontology of the NAZOU project [5] defined two properties in this way: the *isPartOf* property and the *consistsOf* property, which were mutually inverse and allowed for navigation in a hierarchy of regions.

Additional characteristic of a class is the number of levels of its subclasses and also whether or not a class is fully defined by its subclasses. A class is fully defined by its (direct) subclasses if every individual belonging to that class must belong to at least one of its (direct) subclasses.

If a class is fully defined by its subclasses and only has direct subclasses, we represent it by a dropdown list component that contains labels of these subclasses. Figure 1 and 2 shows the example of the *jo:Benefit* class. Its graphical representation is a simple dropdown list and since *jo:Benefit* is a multiple object property of another class (*jo:JobOffer*), it is wrapped in a repeater (bottom).

In this way, users choose the type of instance they want to create. If the class is not fully defined, the dropdown list will also contain the label of the parent class to enable users to create an instance of it instead of its subclasses.

If there is more than one level of subclasses, their presentation should enable users to browse them. If there are only a small number of classes, it is suitable to display them in a dropdown list as in the previous case and indicate the hierarchy by adding a symbol before the actual name of each class (for instance one dot for each level of hierarchy). This approach might not be suitable to display complex and deep hierarchies, where it is better to create a component which simulates the navigation in a tree. E.g. a *listbox*, which contains all classes of the same level and is redrawn with the classes of the next level when the user chooses one value.

The current location in the tree would be indicated next to the component and users would have the ability to return to a higher level in the hierarchy (e.g. a button, hyperlinks in the path).

The same approach can be applied to a hierarchy created by transitive properties between instances. Whether the user can choose a class or instance which is not a leaf of a tree hierarchy is determined from additional metadata about the class. Metadata are also used when both classes and instances are used to define a hierarchy.

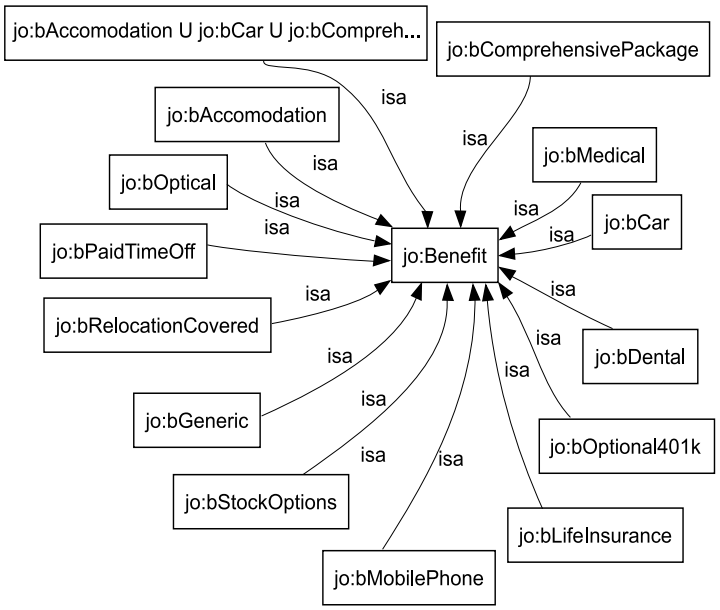


Fig. 1. Example of a class *jo:Benefit* that is fully defined by its direct subclasses.

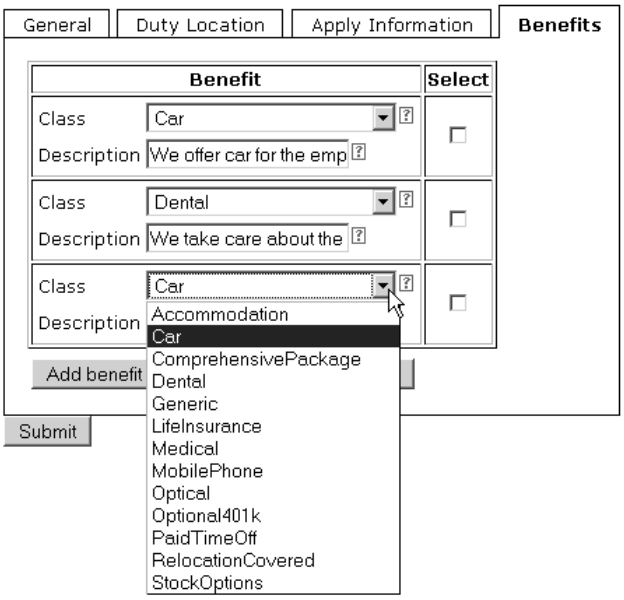


Fig. 2. Visual representation of benefits

4.4 Visual patterns

Visual patterns describe the concepts of sub forms and identify the typical structure, when a class has an object property pointing to another class (which can also have object properties). So, if class A has an object property pointing to class B, there are several possibilities how to represent it in a form for class A:

- Dedicating an area of the form for the properties of class B, usually bounded by a rectangle. This representation is suitable when class B has only few datatype properties.
- Dedicating a special part of the screen to display any object property of class A and creating a button for each object property of class B. Users choose the property, they want to edit by clicking the appropriate button. If a user clicks on a button of a property in class B, the content of the dedicated part would be redrawn and would contain the form for the editing of the instance of class B.
- Creating a button for each object property of class A. If a user clicks a button for class B, a pop-up window would be shown, where the object property could be edited. This solution is not desirable since pop-ups are usually annoying and are often filtered by web browsers.
- Using a tabbed interface, where each tab would represent one object property of class A. Selecting a tab for class B makes it editable. If class B also contained other object properties, a second row of tabs would be available.

5 Form Generation: Mapping patterns to form controls

The basic principle of dynamic form generation lies in the use of data stored in the ontology along with the appropriate metadata to identify ontological patterns. Patterns are used to define a binding between the data in an ontology and their graphical representation to create forms for specific classes and form controls for specific properties.

Since one pattern can have more than one graphical representation, additional metadata must be used to choose the most appropriate one.

Form generation is a recursive process which begins from the given identifier of a class from an ontology for which the form should be generated and continues via its object-type properties. During form generation, the visual description and the data model of the form must be generated. Furthermore, the corresponding implementation related objects such as java classes (e.g., java beans), which store form data must also be generated as well as mapping rules between these classes and the respective ontology. These mapping rules are used by the previously mentioned graph-to-bean and bean-to-graph transformers.

The proposed method matches the structure of each class to patterns described in the previous section. This determines whether the process of form generation is recursively applied to object properties of the class or is terminated by defining a set of simple widgets for display. The matching itself is done by examining the conformance of a selected class to a sequence of patterns in predefined order.

5.1 Form layout

The previously described *widget patterns* map the structure of an ontology to the form controls but do not describe the layout of the form. Although we already discussed the visual representation of object properties between classes (*Visual patterns*), none of the proposed solutions (e.g., tabbed interface) provide information about ordering of the visualization. Moreover, these representations do not allow the personalization of forms preventing us from creating various specialized form templates which fit the needs of individual users. These include hiding of unnecessary form elements or pre-filling forms with default data.

Another problem is that ontologies normally do not contain application-specific information, e.g. whether users are only allowed to choose from existing instances or have to create new ones. Finally, ontologies may contain concepts that should not be displayed on specific forms at all.

Since the lack of flexible form layout support would degrade the proposed solution we defined an additional ontology (fig. 3) which contains information about the order of form tabs and their titles as well as the order of the class properties displayed in these tabs. Additional metadata define whether users are allowed to create new instances of certain classes or are only allowed to

choose from existing ones, or the combination of both. The ontology can also describe predefined values for specific for fields thus allowing for the creation of forms which are more user-friendly compared to forms generated by generic ontology editors such as Protégé[1].

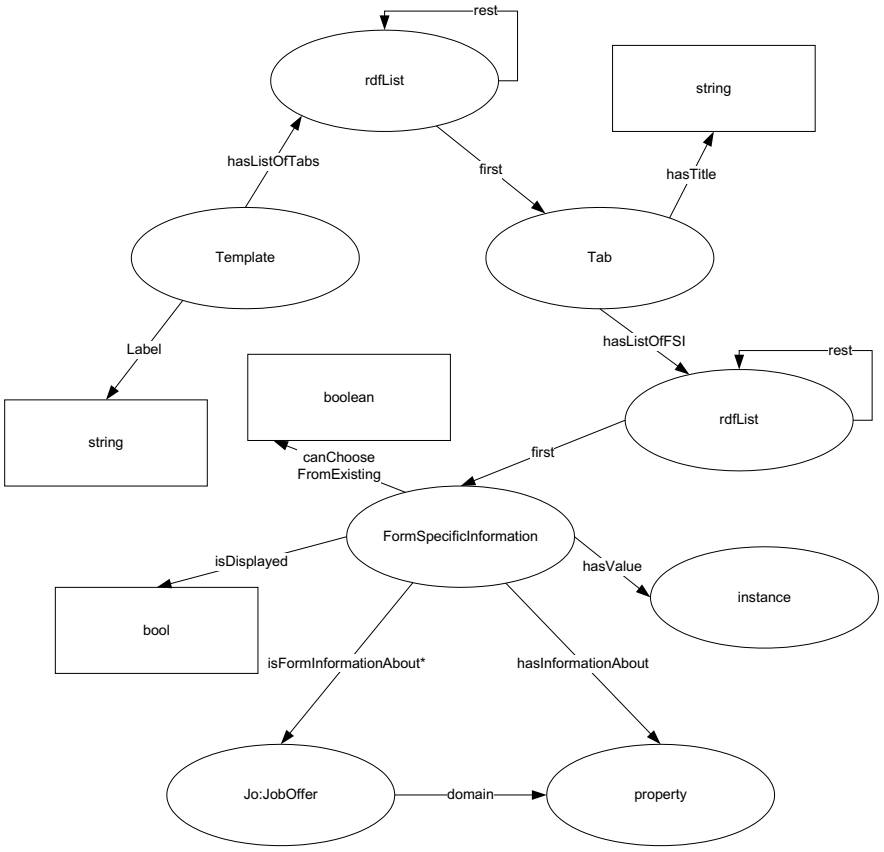


Fig. 3. Meta-model of a form layout information.

The model in fig. 3 shows that our solution supports the reuse of its parts since one Tab instance can be used in multiple lists and one *FormSpecificInformation* instance can be used in multiple tabs. This allows for quick customization of a form template, where some parts of a form can be reused and new layout can be defined for the rest of the form.

The fact that the model is directly connected to a domain ontology allows for easy personalization of a form template to meet the needs of organizations and individuals. If a system determines that a user always fills in the same

[1] Protégé ontology editor, `http://protege.stanford.edu`

value in a field (e.g., duty location or qualification prerequisites) it can create a template for this user which does have these fields pre-filled with the appropriate values (instances in the ontology). On the other hand, if the user is constantly ignoring some field of the template (e.g., level of management or salary bonus) the system can hide these fields from the template while still allowing their use if user explicitly requests the complete form.

The identification of the template that is used for a specific user is stored in a user model which is used for adaptation throughout the whole system if we consider the CRUD pattern as a part of a larger system.

6 Conclusions

We briefly described one of the current trends – the drive towards web applications built around the semantic web principles and technologies. We also explored the problems introduced by the dependency of portal systems on ontologies as means for information storage.

We proposed dynamic form generation as a possible approach which would accommodate changes in ontologies by increasing the flexibility of web portals. We identified and described several patterns in ontologies and their mapping to form controls. With this knowledge we designed an algorithm that dynamically generates form descriptions from ontologies, which can be used by portal solutions to display and process forms for instances from these ontologies.

To verify the feasibility of proposed solution, we created a job offer portal that uses the job offer domain ontology developed as a part of the NAZOU project [5]. The portal enables users to input and publish job offers by filling in various aspects of the respective job offers in appropriately generated web-based forms.

Future work might include the identification of more complex patterns in ontologies and their mapping to form controls. Exploring the possibilities offered by adaptive hypermedia technologies and their relevance to dynamic form generation is another promising direction of research.

References

1. Aldred L, Dumas M, Heravizadeh M, Hofstede A (2002) Ontology Markup for Web Forms Generation. In: Workshop on Real World RDF and Semantic Web Applications
2. Ambler S W (2006) Mapping Objects to Relational Databases: O/R Mapping In Detail.
   `http://www.agiledata.org/essays/mappingObjects.html`
3. Battle S, Jiménez D, Kalyanpur A, Padget J (2004) Automatic Mapping of OWL Ontologies into Java. In: Frank Maurer and Günther Ruhe (eds) Proc. of

the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004)
4. Benjamins V R, Fensel D, Studer R (1998) Knowledge engineering : Principles and methods. Data and Knowledge Engineering, 25(1-2):161-197.
5. Bieliková M, Návrat P, Rozinajová V (2005) Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In: Proc. of International Conference on Computer Systems and Technologies – CompSysTech' 2005. Varna, Bulgaria
6. Dong J S (2004) Software modeling techniques and the semantic Web In: Proc. of 26th International Conference on Software Engineering (ICSE'04), pp. 724-725, IEEE
7. Obrst L (2003) Ontologies for semantically interoperable systems. In: Proceedings of the twelfth international conference on Information and knowledge management (CIKM '03). CM Press 366–369, New Orleans, LA, USA

Príloha F Článok na IIT.SRC 2006

Príloha obsahuje článok, ktorý bol prijatý na konferenciu IIT.SRC 2006 a úspešne prezentovaný účastníkom konferencie.

Dynamic Ontology Based Form Generation for Portal Solutions

Michal Barla, Peter Bartalos, Ján Porubský,
Peter Sivák, Kristián Szobi, Michal Tvarožek [*]

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
tp@atrip.sk

Abstract. Ontological data representation provides several advantages over relational algebra, inherent inference and reasoning among others. It provides many logic-based modeling structures to include metadata describing the ontology itself, which can be used for partial dynamic generation of the web application. In this paper, we propose an approach to transformation of the ontological representation of domain entities into a description of web-based forms for these entities, which is based on identified ontological patterns (sets of RDF/OWL entities and their interconnections corresponding to form controls).

1 Introduction

At present, there are several trends in the design of web-based applications. Perhaps the most prominent is giving machine-understandable meaning (semantics) to the presented content, which is the essential part of the semantic web project [2] and related technologies. Here ontologies and their use for information storage and representation play an important role.

The introduction of ontologies as a mean for information storage provides several advantages over existing approaches. These include the added value of semantic markup [4] and easier incorporation of changes. Ontologies based on first order logic or description logic also support reasoning – drawing conclusions from the facts. An important feature of ontologies is the fact that they natively include metadata, which make them self-descriptive. This allows for effective searching in stored data.

[*] Supervisor: Ing. Roman Filkorn, Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2006, Bratislava, April 26, 2006, pp. 1-1.

The standard language for ontologies is OWL[1] (Web Ontology Language), which is an extension of RDF[2]. RDF represents information in the form of (object, predicate, subject) triples.

The advantages of ontologies have also introduced new challenges. Flexible data storage provided by ontologies would be useless unless applications are flexible enough to accommodate a similar degree of change. One fine example is the task of adding or modifying data stored in an ontology. This can be done by means of a form [1], which allows users to enter all relevant data for a given class instance.

If the ontology changes, the application code corresponding to the form must be manually modified, which is unpractical for real-world applications which would immediately stop working until a patch is issued. Thus the apparent flexibility of ontologies results in the need for flexible applications. In the case of form filling applications this results in the need for flexible form generation tools based not only on data stored in ontologies but also on additional metadata. These are needed because the domain ontology lacks the information about the desired visual data representation. In metadata we define the arrangement of the visual components on the form. Metadata are also necessary in the process of storing the values from the form into the ontology. In our solution both XML files and the ontology are used to store these metadata.

The rest of the paper is structured as follows. Section two introduces the concept of finding and mapping patterns in ontologies to form controls and describes several types of such patterns. Section three describes the proposed approach to dynamic form generation. Section four provides an overview of portal solutions and their relevance to ontologies and dynamic form generation. Section five summarizes the paper and offers insight on future work.

2 Mapping ontology patterns to form controls

We consider the term “Ontology pattern” as the structure of an ontology or its part which is common for a range of ontologies. Ontology patterns can thus be defined on a higher level of abstraction and depict structures that can be found in different ontologies. Furthermore, patterns can be defined using different types of ontological concepts – classes, relations between them, properties, instances and restrictions.

The basic principle of dynamic form generation lies in the use of data stored in the ontology along with the appropriate metadata to identify ontological patterns. Patterns are used to define a binding between the data in an ontology and their graphical representation to create forms for specific classes and form controls for specific properties. Since one pattern can have more than one graphical representation, additional metadata must be used to choose the most appropriate one.

We identified two basic groups of patterns: *top-level* patterns and *detailed* patterns. *Top-level* patterns describe the concepts of sub forms. Patterns identify the typical structure, when a class has an object property pointing to another class (which

[1] Web Ontology Language, `http://www.w3.org/2004/OWL/`
[2] Resource Description Framework, `http://www.w3.org/RDF/`

can also have object properties). So, if class A has an object property pointing to class B, there are several possibilities how to represent it in the form for class A:

- Dedicating an area of the form for the properties of class B, usually bounded by a rectangle. This representation is suitable when class B has only few datatype properties.
- Dedicating a special part of the screen to display any object property of class A and creating a button for each object property of class B. Users choose the property, they want to edit by clicking the appropriate button. If a user clicks on a button of a property in class B, the content of the dedicated part would be redrawn and would contain the form for the editing of the instance of class B.
- Creating a button for each object property of class A. If a user clicks a button for class B, a pop-up window would be shown, where the object property could be edited. This solution is not desirable since pop-ups are usually annoying and are often filtered by web browsers.
- Using a tabbed interface, where each tab would represent one object property of class A. Selecting a tab for class B makes it editable. If class B also contained other object properties, a second row of tabs would be available.

Detailed patterns focus on the structure of classes, subclasses, properties and possible restrictions. We identified several patterns with different graphical representations.

2.1 Simple patterns

Patterns for *primitive datatype properties* include all properties, whose ranges are literals (string, integer, float, date, boolean etc.). The graphical representation of these patterns is straightforward – the *rdfs:label* of each property is displayed next to the input field for its value. This would generally be a dropdown list for Boolean values (true, false, undefined), a calendar for date values and a textbox for text strings. It is more convenient to use a textarea input field for longer strings as it increases the readability of the form. To distinguish the cases where a normal textbox and where a textarea should be used, we can define additional metadata about desired form template or we can compare mean length of existing instances to some predefined threshold.

If the cardinality of the respective property is multiple, the above elements would be placed inside a special control called *repeater*, which enables users to add/remove more values by means of additional buttons for these actions.

Patterns for *same-range object properties* identify classes that have several object properties with multiple cardinality and the same range (class or a union of classes). In this case, the fields for the range are displayed only once and an additional component is used to distinguish the specific property that is being edited. As an example, we can mention *requires* and *prefers* properties of *Prerequisite* class from the domain ontology of the NAZOU [3] project. Such a component can be either a dropdown list or a set of radio buttons. All these components are wrapped in a repeater to allow multiple values to be edited.

Patterns for *enumerations* identify object properties, whose range is a class fully defined by its instances. It should not be possible to add new or edit existing instances. Enumerated classes are defined in the ontology itself, e.g. a class representing the days of the week or various time periods (second, hour, day, week, month etc.).

Several graphical representations of this pattern exist. If the cardinality of an object property whose range is an enumerated class is single then it can be represented by a dropdown list of its instances. If the cardinality is multiple, the mentioned dropdown list can be wrapped in a repeater or instances can be represented by a *listbox*.

Besides enumerated classes, also classes which allow users to create new instances or choose an existing one can be identified. This information must be stored in the metadata for the appropriate class. In this case all fields necessary to create an instance would be displayed and a dropdown list or *listbox* with existing instances would be added as mentioned above.

2.2 Patterns for tree hierarchies

This section describes the most complex patterns we identified in ontologies – hierarchical structures. Since hierarchies offer a wide range of values, they must be structured in a way that allows users to easily understand and choose amongst them. For example, when users want to choose a country where a company is based, it might be convenient for them to first choose a continent, then a country on that continent, etc.

In ontology, there are two basic ways to represent tree hierarchies. The standard property *subclassOf* between classes which represents an *is a* relation and/or custom defined properties between instances can be used, which define relationships between nodes in the tree hierarchy.

One can assume that if a class in an ontology has a property which is transitive and points to instances of the same class (its range and domain are the same), then it is used to represent some form of hierarchy. The job offer ontology we used was created as a part of a larger research project [3] and defined two properties in this way: the *isPartOf* property and the *consistsOf* property, which were mutually inverse and allowed for navigation in a hierarchy of regions.

Additional characteristic of a class is the number of levels of its subclasses and also whether or not a class is fully defined by its subclasses. A class is fully defined by its (direct) subclasses if every individual belonging to that class must belong to at least one of its (direct) subclasses.

If a class is fully defined by its subclasses and only has direct subclasses, we represent it by a dropdown list component that contains labels of these subclasses. On Fig. 1 we can see the example of the *jo:Benefit* class. Its graphical representation is a simple dropdown list and since *jo:Benefit* is a multiple object property of another class (*jo:jobOffer*), it is wrapped in a repeater (shown on the right).

In this way, users choose the type of instance they wants to create or use. If the class is not fully defined, the dropdown list will also contain the label of the parent class to enable users to create an instance of it instead of its subclasses.

If there is more than one level of subclasses, their presentation should enable users to browse them. If there are only a small number of classes, it is suitable to

display them in a dropdown list as in the previous case and indicate the hierarchy by adding a symbol before the actual name of each class (for instance one dot for each level of hierarchy). This approach might not be suitable to display complex and deep hierarchies, where it is better to create a component which simulates the navigation in a

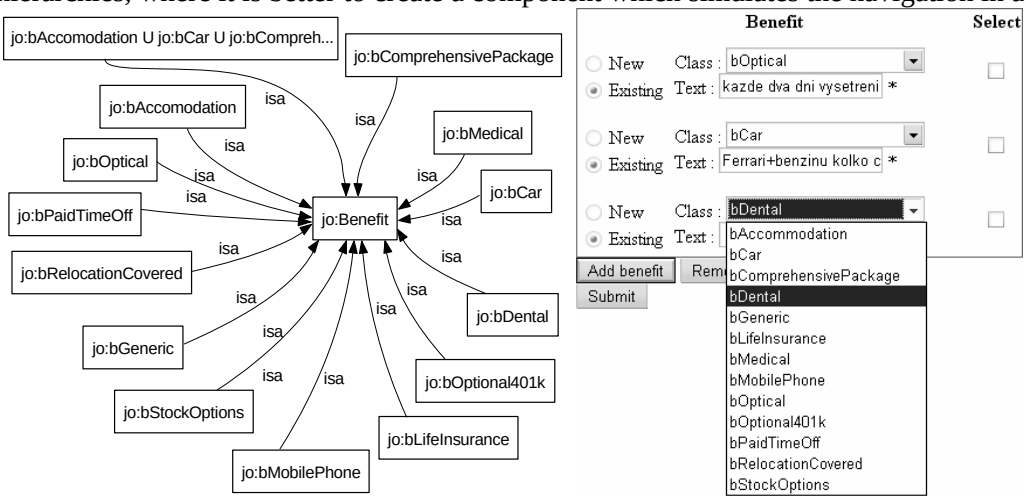


Fig. 1. Example of a class *jo:Benefit* that is fully defined by its direct subclasses.

tree. E.g. a *listbox*, which contains all classes of the same level and is redrawn with the classes of the next level when the user chooses one value. The current location in the tree would be indicated next to the component and users would have the ability to return to a higher level in the hierarchy (e.g. a button, hyperlinks in the path).

The same approach can be applied to a hierarchy created by transitive properties between instances. Whether the user can choose a class or instance which is not a leaf of a tree hierarchy is determined from additional metadata about the class. Metadata are also used when both classes and instances are used to define a hierarchy.

3 An Approach to Form Generation

Form generation is a recursive process which begins from the given identifier in the ontology of a class for which the form should be generated and continues on its object-type properties. Apart from the representation of the form, method produces java bean classes to store the form data as well as additional metadata for mapping objects to ontology. The mapping is performed by a pair of graph-to-bean and bean-to-graph transformers (analogy with O/R mapping in relational databases). Java bean objects are mapped to corresponding RDF graphs (and vice versa) using additional metadata about the ontology and java beans, acquired from the form generation process. These metadata contain information needed for binding the java beans and their fields to the OWL classes and their properties. The metadata comprise the name of the java bean, the names of java bean's fields, information about the OWL class properties (e.g. multiplicity) and the type of the object in the RDF triple (e.g. object type or a literal).

Designed method uses more data about the layout of the form. We defined an additional ontology which holds information about the order of the form tabs and their titles as well as the order of the class properties displayed in these tabs. In addition, metadata define whether user is allowed to create new instances of a certain class or is allowed only to choose from the existing ones, or the combination of both. All this metadata allow for creation of forms which are more user-friendly compared to forms generated by generic ontology editors like Protégé[3].

The proposed method tries to match the structure of each class to patterns described in the previous section. This determines whether the process of form generation is recursively applied to object properties of the class or is finished at this point by defining a set of simple widget to be displayed.

4 Experimental evaluation

We decided to verify the assertions mentioned in previous sections and implement a form generation algorithm used by a portal web site operating with an ontological repository. We consider a portal to be a virtual place, where various categories of people can interact by using portal's services and functionality.

Portal in our solution stands for an online labor market where organizations would put offers on their free positions which would be retrieved by people looking for a job. It is intended to become a part of the project [3] and use its domain ontology.

Because of the requirements of the NAZOU project, the solution should be based on free open source software (FOSS) and J2EE platform.

When considering the J2EE platform one of the most common components of portal systems are portlets. Portlets are parts of the portal itself each embedding one functionality of the portal (e.g. User management). Portlets are very flexible – they can be easy added or removed from a web site according to user preferences. Moreover, one feature of portlets enables users to visually maximize or minimize the portlet itself.

We tried to avoid creating the whole portal solution from scratch and we chose Apache Cocoon as our portal framework. Cocoon is not only a portal framework since it can also be used as a publishing framework thanks to a revolutionary solution of content generation based on pipeline architectural model. Pipelines are used for portal form content generation that can be, for instance, displayed in the portlets afterwards [5]. Definition of pipelines is central. The data in a pipeline is produced by a generator component. Consequently, the data can be transformed by a series of transformers. The last pipeline component is a serializer that outputs the transformed data in the required format.

Knowledge of the framework and its features is important when considering the form generation algorithm. We need to know how the framework handles the forms, what are the forms definition like etc.

[3] Protégé ontology editor, `http://protege.stanford.edu`

4.1 Cocoon and forms handling

Cocoon has an advanced forms framework - CForms. It is based on MVC architectural pattern where a form definition is divided into several XML files. A *form model* is a definition of form widgets with corresponding datatypes. Based on the form definition a form instance can be created. A *form template* defines the layout of the form by defining tags referencing *widgets* from the form model in places where *widgets* are supposed to appear. These tags are processed by the Forms Template Transformer, and replaced by the XML representation of *widgets.* Moreover, CForms contains binding framework that allows for binding of the form data to an xml structure or a java bean class (we will use the term object to refer to either of these). Before showing the form, the data are copied from the object to the form, and after the form has been validated, the data in the form are copied back to the object and can be further processed [6].

4.2 Form generation for Cocoon Forms

The life-cycle of an input/output form is shown in Fig. 2. First, data are loaded from the ontology into internal representation, which corresponds to an XML file that is used as the source for pipeline generators. Every pipeline outputs appropriate files as required by the Cocoon Forms framework as described in previous section. Having all the necessary files, the form itself is displayed at which point the user fills the form with relevant information.

By submitting the form, the values are stored in the corresponding java bean classes. This allows for storing the values into the ontology. To save the data into the ontology, it is necessary to know the corresponding identifiers in the ontology of the properties in the ontology, which are stored in metadata that hold this information during the process of transformation of ontology data into the internal representation.

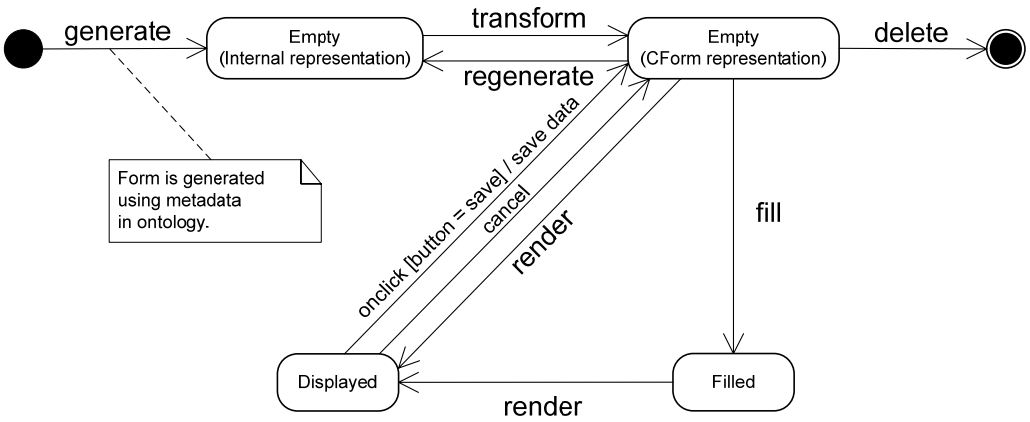


Fig. 2. The life-cycle of the form

5 Conclusions

We described one of the current trends in software design. It is the drive towards web applications that is also influenced by the semantic web. We also explored the problems introduced by the dependency of portal systems on ontologies as means for information storage.

We proposed dynamic form generation as a possible approach which would accommodate changes in ontologies by increasing the flexibility of web portals. We identified and described several patterns in ontologies and their mapping to form controls. With this knowledge we designed an algorithm that dynamically generates form descriptions from ontologies, which can be used by portal solutions to display and process forms for instances from these ontologies.

Future work might include the identification of more complex patterns in ontologies and their mapping to form controls. Exploring the possibilities offered by adaptive hypermedia technologies and their relevance to dynamic form generation is another promising direction of research.

Acknowledgment: This work was partially supported by the State programme of research and development "Establishing of Information Society" under the contract No. 1025/04.

References

[1] Aldred, L., Dumas, M., Heravizadeh, M., Hofstede, A.: Ontology Markup for Web Forms Generation. In Workshop on *Real World RDF and Semantic Web Applications*, 2002.

[2] Dong, J.S.: Software Modeling Techniques and the Semantic Web, In: *Proc. of 26th International Conference on Software Engineering (ICSE'04)*, pp. 724-725, IEEE.

[3] Návrat, P., Bieliková, M. and Rozinajová, V.: Methods and Tools for Acquiring and Presenting Information and Knowledge in the Web. In *Proc. of International Conference on Computer Systems and Technologies - CompSysTech' 2005*. June 2005, Varna, Bulgaria. Available on the Web `http://ecet.ecs.ru.acad.bg/cst05/Docs/cp/SIII/IIIB.7.pdf` (April 2006)

[4] Obrst, L.: Ontologies for Semantically Interoperable Systems. In: *Proc. of 12th international conference on Information and knowledge managements,* pp. 366 – 369 ACM Press, 2003.

[5] *Apache Portals Project*, `http://portals.apache.org` (April 2006)

[6] *Cocoon Forms Introduction*, `http://cocoon.apache.org/2.1/userdocs/basics/index.html` (April 2006)

Príloha G Používateľská dokumentácia

Príloha používateľskú dokumentáciu k portálu pracovných príležitostí.

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

POUŽÍVATEĽSKÁ DOKUMENTÁCIA

JOBS
PORTÁL PRACOVNÝCH PRÍLEŽITOSTÍ

Tím č. 8: Hachiban
Vedúci tímu: tvarozek@gmx.net
Mailový alias: tp@atrip.sk
Predmet: Tvorba softvérového systému v tíme
Pedag. vedúci: Ing. Roman Filkorn
Ak. rok: 2005/2006

Členovia tímu:
Bc. Michal Barla
Bc. Peter Bartalos
Bc. Ján Porubský
Bc. Peter Sivák
Bc. Kristián Szobi
Bc. Michal Tvarožek

Obsah

Úvod

Ďakujeme Vám za zakúpenie produktu od tímu č. 8 Hachiban. Zakúpený produkt predstavuje webový informačný systém (IS) – Portál pracovných príležitostí, určený primárne pre napĺňanie údajov o ponukách pracovných príležitostí aktívnymi producentmi.

Kapitola 1 bližšie opisuje inštaláciu systému. Kapitola 2 opisuje hlavné koncepty systému, ktorých pochopenie je nevyhnutné k úspešnému použitiu systému. Kapitola 3 opisuje postup pri vykonávaní jednotlivých úloh, ktoré je možné v rámci systému realizovať. Kapitola 4 obsahuje často kladené otázky a opisuje postupy pri riešení najčastejších problémov.

Inštalácia produktu

Táto časť opíše postup pri inštalácii informačného systému Portál pracovných príležitostí.

Minimálne a odporúčané požiadavky

Hardvér (minimálne požiadavky)

- Dvojjadrový procesor AMD Athlon 64 X2 4400+ alebo kompatibilný
- 1 GB operačnej pamäte
- 2 GB voľného miesta na disku
- pripojenie do Internetu

Hardvér (odporúčaná konfigurácia)

- Štvorprocesorový systém s dvojjadrovými procesormi AMD Opteron 885 alebo kompatibilný
- 16 GB ECC REG operačnej pamäte
- 2 TB voľného miesta na disku na RAID 6 diskovom poli
- Dvojité 1GB/s pripojenie do Internetu

Softvér

Pre inštaláciu systému sú nevyhnutné minimálne nasledovné softvérové komponenty:

- Operačný systém s podporou Java Virtual Machine (Windows XP/2003, Linux)
- Java Virtual Machine 1.5
- Apache Tomcat v5.5.12
- Apache Cocoon v2.1.8
- Sesame v1.2.4

Inštalácia systému zaškoleným pracovníkom (odporúčané)

Inštaláciu produktu na cieľový server vykonáva zaškolený pracovník – člen tímu č. 8 Hachiban na mieste u zákazníka, pričom zabezpečí dostupnosť potrebného softvéru od tretích strán, súborov pre samotný portál ako aj konfiguráciu jednotlivých súčastí tak, aby systém fungoval.

Na klientských zariadeniach je potrebné mať nainštalovaný grafický webový prehliadač s podporou JavaScriptu a technológie AJAX.

Inštalácia systému vo vlastnej réžii zákazníka

Systém Portál pracovných príležitostí môže inštalovať aj zákazník samostatne, avšak tento postup nie je odporúčaný kvôli komplexnosti konfigurácie jednotlivých súčastí systému a nevyhnutnosti podrobných znalostí o jednotlivých komponentoch.

Rámcový postup inštalácie

1. Nainštalovať Java Virtual Machine
2. Nainštalovať aplikačný server Apache Tomcat
3. Nasadiť Sesame do Tomcatu
4. Nasadiť Apache Cocoon do Tomcatu
5. Nakonfigurovať ontologické úložisko v Sesame
    a. Vytvoriť používateľov a RDFS úložisko

   b. Importovať do úložiska doménovú ontológiu pracovných ponúk NAZOU a ontológiu so zápisom vzhľadu formulárov
6. Nasadiť portál do Cocoonu
7. Nakonfigurovať portál
   a. Nakonfigurovať umiestnenie portálu
   b. Nakonfigurovať použitie ontologického úložiska v Sesame
   c. Inicializovať vygenerovanie formulárov pre pracovné ponuky

Koncept systému

Portál pracovných príležitostí predstavuje webovú aplikáciu, ktorá s používateľmi komunikuje prostredníctvom webového rozhrania v bežnom webovom prehliadači. Samotný portál je primárne určený aktívnym producentom ponúk, ktorí v portáli pracujú s údajmi o zverejnených pracovných ponukách, avšak nie je vylúčené ani jeho použitie používateľmi, ktorí si hľadajú zamestnanie – hľadajú pracovné ponuky.

Z hľadiska koncepcie systému sú dôležité štyri entity – systém samotný a jeho údržba, ponuka voľného pracovného miesta, organizácia, ktorá ponúka pracovné miesta prostredníctvom zverejňovania pracovných ponúk a používatelia, ktorí za jednotlivé organizácie vkladajú pracovné ponuky do systému.

Portál pracovných príležitostí

Údržbu a prevádzku samotného systému realizuje administrátor systému (špeciálny typ používateľa), ktorý vykonáva všetky činnosti nevyhnutné k fungovaniu systému. Predvolený používateľský účet administrátora systému má prihlasovacie meno *cocoon* a heslo *cocoon*.
Činnosti vykonávané *administrátorom systému*:

- Registrácia nových a správa existujúcich organizácií
- Globálna správa používateľov
- Zálohovanie a archivácia údajov
- Generovanie a správa formulárov
- Správa šablón formulárov a metadát potrebných ku generovaniu formulárov
- Aktualizácia softvérového vybavenia

Registrácia nových organizácií je riešená dvojkrokovým procesom. Nová organizácia sa zaregistruje, čím ju systém pridá do zoznamu organizácií a vytvorí používateľa – administrátora za organizáciu ako neaktívneho používateľa. V druhom kroku administrátor systému overí organizáciu a potvrdí jej registráciu aktiváciou používateľského účtu jej administrátora.

Ponuka

Ponuka prestavuje údaje o voľnej pracovnej príležitosti uložené v systéme. Ponuky do systému zadávajú prihlásení používatelia – personalisti, pričom prístup k zverejneným ponukám majú aj neprihlásení používatelia a teda aj verejnosť. Samotná ponuka môže obsahovať rôzne údaje o ponúkanom pracovnom mieste ako sú:

- Názov pracovného miesta
- Názov zamestnávateľa
- Miesto vykonávania práce
- Ponúkaný plat a výhody (benefity)
- Dátum nástupu do pracovného pomeru
- Typ a dĺžka pracovného pomeru
- Požiadavky kladené na uchádzačov (vzdelanie, prax, schopnosti)
- Informácie pre uchádzačov o spôsobe prihlásenia sa na pracovné miesto

Organizácia

Organizácia ponúka voľné pracovné miesta buď priamo alebo sprostredkovane pomocou agentúry. Organizácia v rámci systému nevystupuje priamo ale prostredníctvom špeciálneho používateľa – administrátora za danú organizáciu.
Činnosti vykonávané *administrátorom za organizáciu*:

- Zabezpečenie aktuálnosti údajov o organizácii v systéme
- Registrácia nových a správa existujúcich používateľov za danú organizáciu
- Globálna správa pracovných ponúk za danú organizáciu

Používatelia

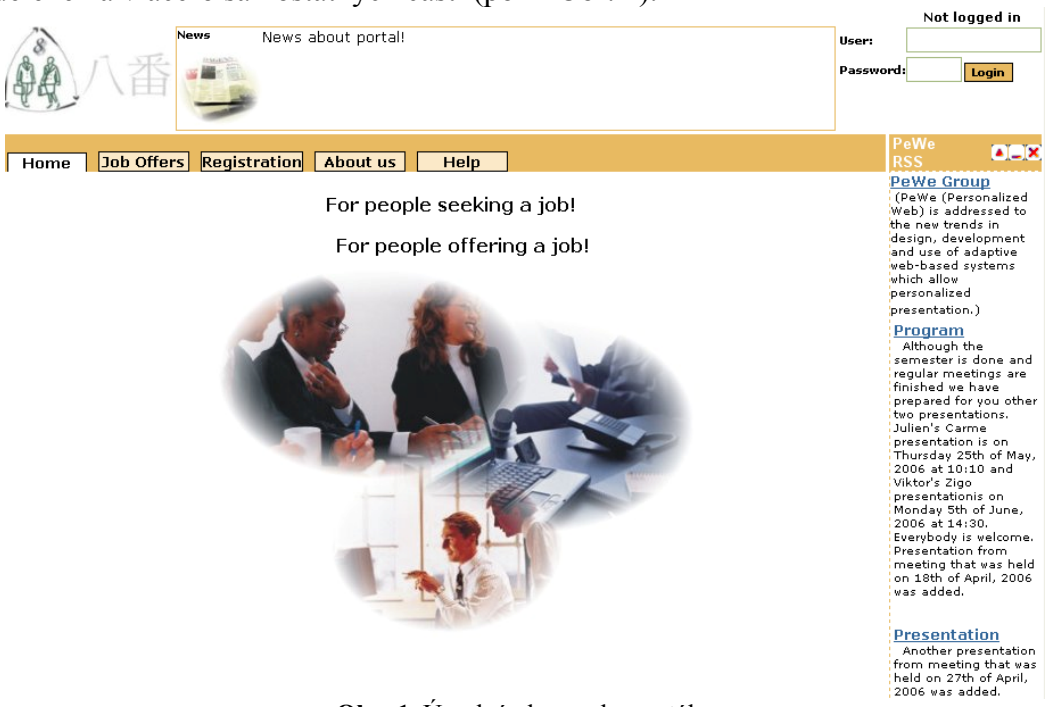
Používatelia predstavujú pracovníkov organizácií – aktívnych producentov (personalistov), ktorí budú so systémom pracovať najčastejšie a ktorých hlavnou úlohou je do systému vkladať údaje o nových pracovných miestach, resp. udržiavať a spravovať údaje o existujúcich ponukách pracovných príležitostí.

Na prácu s ponukami je nevyhnutné aby používateľ bol prihlásený do systému, pričom neprihlásení používatelia majú prístup k všetkým zverejneným ponukám na čítanie. Systém podporuje tri typy prihlásených používateľov:

- *Základný používateľ* (personalista) môže pracovať len s ponukami, ktoré sám vytvoril, pričom ponuky ostatných používateľov sú mu neprístupné (má k nim prístup len ako neprihlásený používateľ).
- *Administrátor* môže pracovať so všetkými ponukami za danú organizáciu.

Používateľské rozhranie

V tejto časti je opísané používateľské rozhranie systému a postup vykonania jednotlivých úloh používateľmi. Portál pracovných príležitostí používa grafické webové rozhranie, ktoré je možné používateľsky prispôsobiť potrebám konkrétnych používateľov. Predvolené rozhranie je rozdelené na viacero samostatných častí (pozri Obr. 1).



Obr. 1. Úvodná obrazovka portálu.

Hlavné menu sa nachádza v hornej časti, pole pre prihlásenie sa používateľa sa nachádzajú v pravej hornej časti. Aktuálne informácie o dianí na portáli sú zobrazené v ľavej hornej časti obrazovky. Hlavná časť rozhrania v strede je určená pre zobrazenie ovládacích prvkov podľa aktuálnej polohy používateľa v portáli. Prispôsobenie používateľského rozhrania je možné vykonať pomocou tlačidiel („full screen“, „minimize“, „delete“) v pravom hornom rohu vybraných častí rozhrania – aktívnych oblastí (copletov). Na ukážku konfigurovateľnosti a možností prispôsobenia portálu je v pravej časti umiestnený RSS kanál skupiny Personalized Web.

Položky menu (neprihlásený používateľ)

- *Home* – domovská stránka
- *Job Offers* – prehľad ponúk uložených v systéme
- *Registration* – registrácia novej organizácie
- *About us* – informácie o systéme a jeho autoroch
- *Help* – pomoc; základné informácie o používaní portálu

Položky menu (prihlásený používateľ)

- *Add Offer* – pridanie novej ponuky
- *Job Offers* – prehľad ponúk uložených v systéme
- *About us* – informácie o systéme a jeho autoroch
- *Help* – pomoc; základné informácie o používaní portálu

Položky menu nástroje – „Tools“ (neprihlásený používateľ)

- *Coplet Management* – nastavenie používateľského rozhrania
  - *Edit Layout* – nastavenie rozloženia copletov
  - *Select Skin* – nastavenie vzhľadu portálu
- *User Management* – správa používateľov
  - *Show User* – zobrazenie informácií o používateľovi
  - *Edit User* – zmena profilu používateľa
  - *Add User* – pridanie nového používateľa (len administrátori)

Prehliadanie ponúk

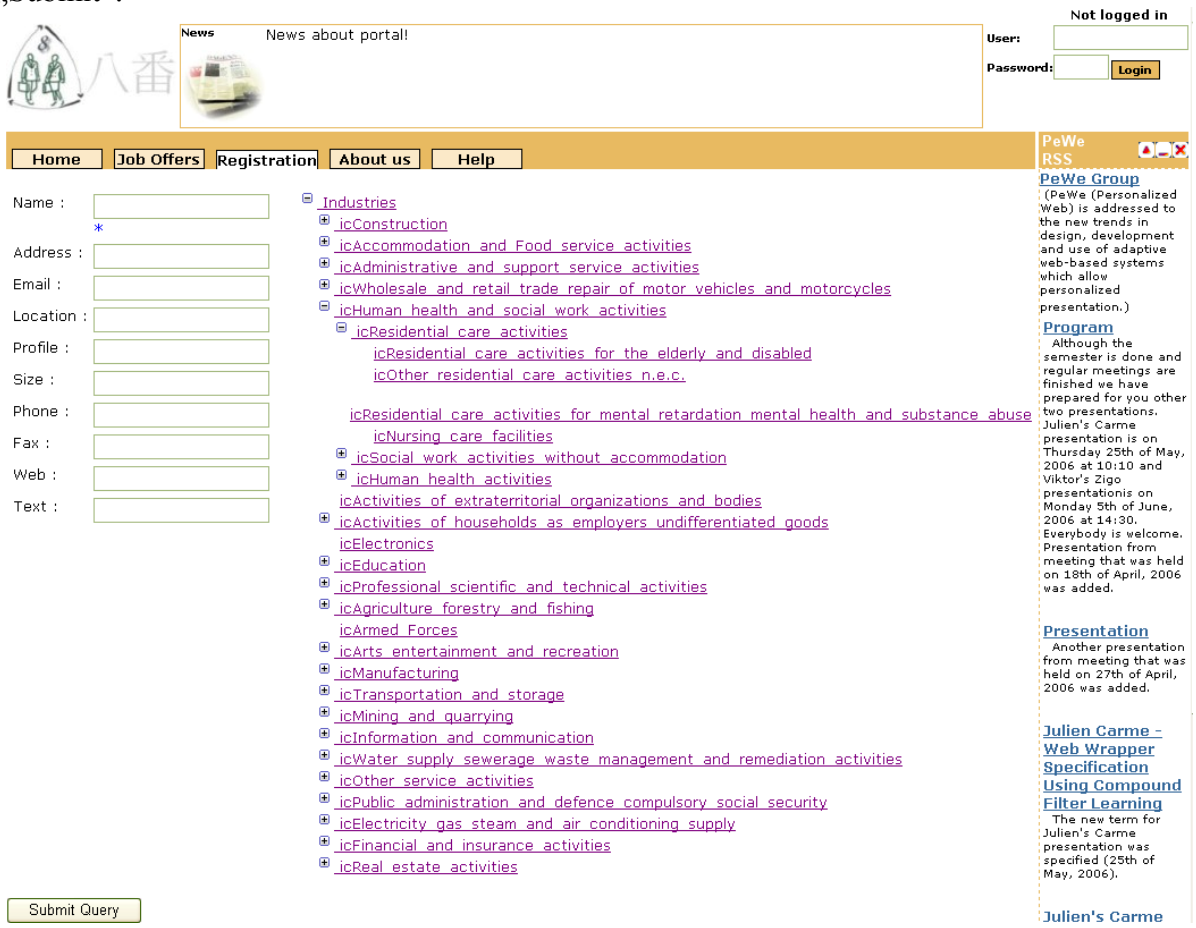
Vyberte v hlavnom menu voľbu pre prehliadanie ponúk „Job Offers“, zobrazí sa prehľad ponúk cez rozhranie fazetového prehliadača (Obr. 2). V ľavej časti je možné definovať ohraničenia na zobrazené ponuky podľa miesta vykonávania práce, odvetvia a profesie. Navigácia medzi jednotlivými stranami výsledkov sa nachádza v strednej hornej časti hlavnej obrazovky. Pod ňou je umiestnená skupina odkazov na definovanie usporiadania výsledkov vyhľadávania (vzostupne/zostupne).

Výsledky vyhľadávania sa nachádzajú v strednej časti obrazovky. Po nasledovaní odkazu – názvu ponuky sa zobrazia detaily o zvolenej ponuke. Červenou farbou pozadiu sú zobrazené nezverejnené ponuky, zelenou farbou zverejnené ponuky. Po prihlásení sa je možné vykonávať operácie nad zvolenou ponukou pomocou tlačidiel v pravej časti obrazovky pri každej ponuke (editovať, zverejniť/skryť, zmazať). Ak tlačidlá nie sú zobrazené, používateľ nie je prihlásený alebo nemá dostatočné privilégiá na vykonanie danej operácie.

Obr. 2. Prehľad ponúk v rozhraní fazetového prehliadača.

Požiadanie o registráciu organizácie

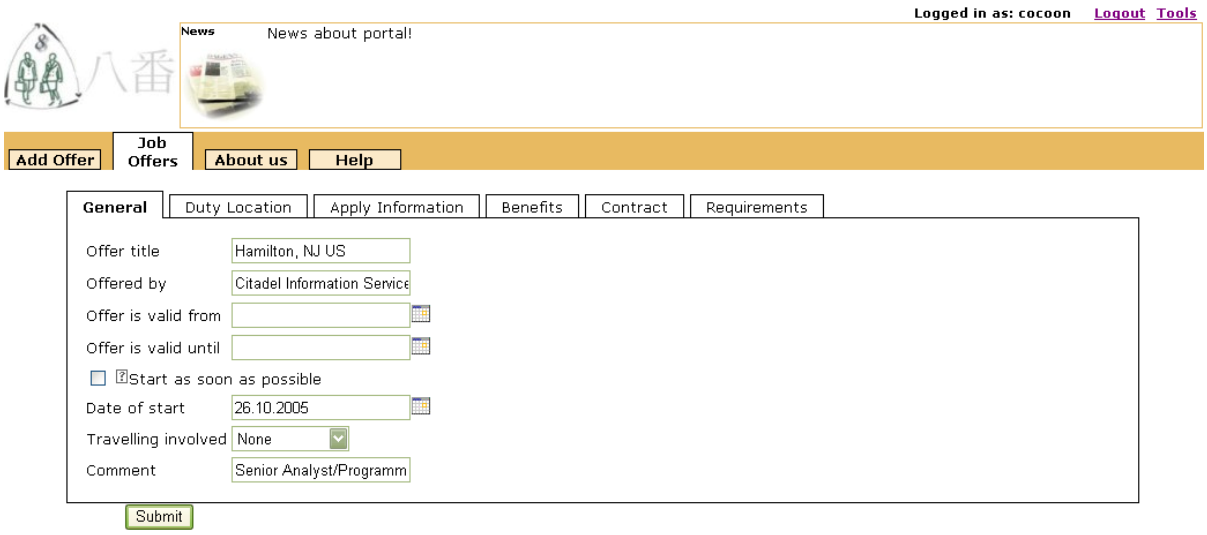
V hlavnom menu vyberte voľbu „Registration“, zobrazí sa formulár pre registráciu organizácie (Obr. 3). Vyplňte požadované údaje o organizácii do zobrazeného formulára, v pravej časti vyberte odvetvie v ktorom organizácia pôsobí a pokračujte pomocou voľby „Submit“. Následne vyplňte údaje o používateľovi – administrátorovi za organizáciu a stlačte „Submit“.



Obr. 3. Registrácia novej organizácie.

Vytvorenie novej editovanie existujúcej ponuky

Pre vytvorenie novej ponuky zvoľte v menu „Add Offer“, zobrazí sa formulár pre editovanie ponuky (Obr. 4). Vyplňte jednotlivé polia formulára v každej záložke (tabke) a pokračujte voľbou „Submit“. V prípade nesprávneho vyplnenia niektorého poľa vás portál vyzve na opravu a odmietne ponuku uložiť.

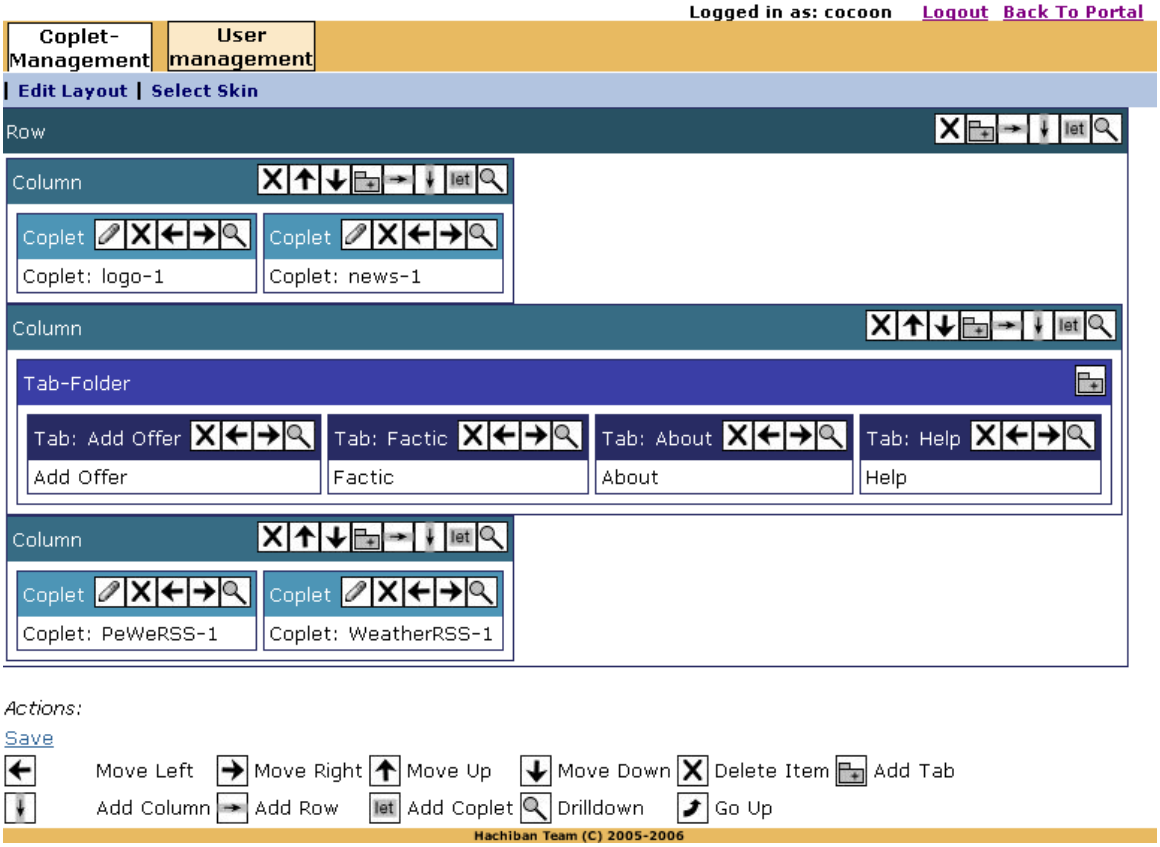


Obr. 4. Formulár pre editovanie ponuky.

Zmena a prispôsobenie grafického rozhrania

Používateľ si môže prispôsobiť rozhranie portálu v menu „Tools“ pod voľbou „Coplet Management“ (Obr. 5). Portál podporuje zmenu rozmiestnenia aktívnych oblastí (copletov) a výber štýlu zobrazovania (skin).

Podľa legendy v dolnej časti obrazovky je možné upraviť rozmiestnenie jednotlivých copletov v strede obrazovky (napr. zmena polohy, zobrazenie/skrytie). Výsledné nastavenie potvrďte voľbou „Save“.

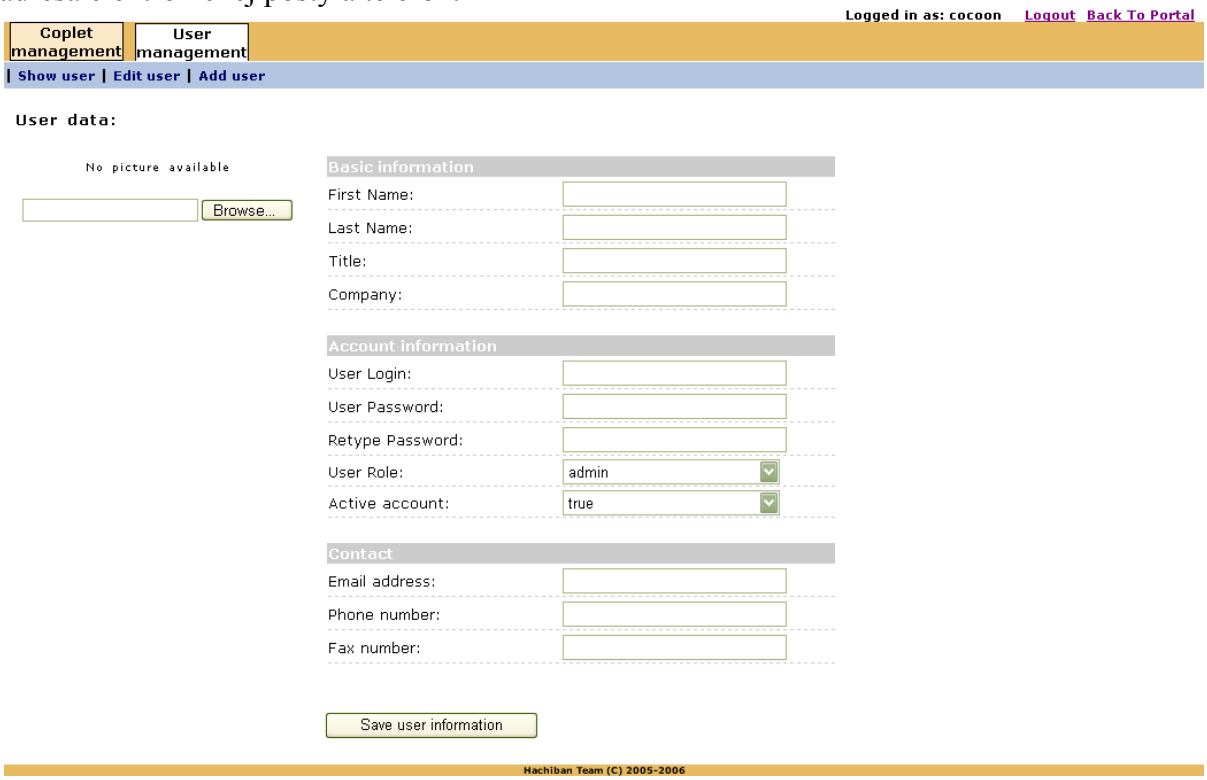


Obr. 5. Prispôsobenie grafického rozhrania portálu zmenou usporiadania copletov.

Pridanie nového používateľa a editovanie používateľských nastavení

Pre správa používateľov zvoľte voľbu „Tools“, „User Management“. Následne si vyber pridanie používateľa (Obr. 6), prezeranie vlastného profilu alebo editovanie iných používateľov.

Pre každého používateľa je možné zadať základné informácie ako meno, priezvisko a spoločnosť do ktorej patrí. Ďalej potrebné zvoliť login a heslo, úroveň prístupových práv. Pre potreby komunikácie s používateľom sa odporúčajú vyplniť kontaktné informácie – adresa elektronickej pošty a telefón.



Obr. 6. Pridanie nového používateľa do systému.

Často kladené otázky a riešenie problémov

Táto časť obsahuje často kladené otázky používateľov súvisiace s Portálom pracovných príležitostí a riešenia najčastejších problémov, s ktorými sa môžu používatelia počas prevádzky systému stretnúť.

Často kladené otázky

Otázka: Je potrebné mať nainštalovaný Java Virtual Machine na spustenie systému?
Odpoveď: Áno.

Riešenie problémov

Problém: Systém indikuje nedostatok operačnej pamäte.
Riešenie: Vypnutie a opätovné spustenie systému (Tomcat spotrebúva nadmerné množstvo pamäte, ktorú nie vždy uvoľní).

Problém: Prehľadávanie ponúk nefunguje, systém hlási chybu „Coplet Factic-1 is not available".
Možná príčina: Nesprávne nakonfigurovaný prístup k úložisku.
Riešenie: Nastavte správne parametre pre spojenie s úložiskom Sesame v súbore portal/coplets/Factic/factic.properties.

Problém: Prehľadávanie ponúk nefunguje, systém hlási chybu „Coplet Factic-1 is not available".
Možná príčina: Úložisko Sesame nie je spustené alebo je prázdne.
Riešenie: Spustite Sesame, presvedčte sa, či úložisko obsahuje aktuálnu verziu doménovej ontológie (v0.6.16 ku 25.5.2006).

Problém: na monitore nič nevidno.
Možná príčina: nemáte zapnutý počítač (monitor).
Riešenie: Problém sa bohužiaľ nedá opraviť, vráťte CD s produktom do najbližšej predajne, po preukázaní sa platným pokladničným dokladom Vám bude refundovaná cena tovaru.

ČASŤ II
–
RIADENIE

Obsah

1. Úvod

Predložená dokumentácia obsahuje zbierku dokumentov súvisiacich s riadením tímového projektu, ktoré vznikli počas jeho riešenia. Obsiahnuté dokumenty sme vytvorili s cieľom sprehľadniť a zefektívniť riadenie projektu a zdokumentovať spôsob fungovania tímu.

V jednotlivých častiach dokumentácie sú postupne za sebou zaradené jednotlivé dokumenty. Prvým v poradí je ponuka na tému *JOBS – portál pracovných príležitostí* a priesvitka, použitá pri prezentácii ponuky. Nasledujú plány projektu na nasledujúce obdobie, resp. rámcové plány do konca riešenia projektu, ktoré predstavujú základ pre vyhodnocovanie stavu projektu.

Ďalej uvádzame informácie o organizácii tímu, rozdelenie rolí medzi jednotlivých riešiteľov a zápisy z formálnych stretnutí tímu s pedagogickým vedúcim. Nasledujú posudky a vyjadrenia k posudkom, vyjadrenie k manažmentu verzií a preberacie protokoly.

Forma zápisnice sa postupne od začiatku projektu menila, reagujúc tak na nové požiadavky.

2. Ponuka

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

Ponuka tímu č. 8

JOBS
Portál pracovných príležitostí

Mailový alias: tp@atrip.sk

Členovia tímu: Michal Barla
Peter Bartalos
Ján Porubský
Peter Sivák
Kristián Szobi
Michal Tvarožek

Október, 2005

Obsah

Zadanie – Portál pracovných príležitostí

Počet tímov: 1

Vedúci tímu: Ing. Roman Filkorn

Portál pracovných príležitostí je moderným nástrojom pre úspešnú výmenu informácií dvoch zúčastnených strán: zamestnávateľov a uchádzačov o zamestnanie. Portál na jednej strane poskytuje zamestnávateľom možnosť zverejniť ponuku pracovnej príležitosti širokej cieľovej komunite, na strane druhej umožňuje vyhľadávať medzi pracovnými ponukami a vyberať si na základe vlastného profilu a záujmov uchádzača o prácu. Portál pomáha spravované informácie kategorizovať, sledovať aktuálnosť, poskytovať previazanie s podobnými portálmi v iných (geografických) oblastiach, uľahčiť opakované činnosti najmä uchádzačom o prácu.

Portál bude integrovanou súčasťou väčšieho projektu. Bude vhodné, keď bude portálové riešenie vychádzať z existujúcich analýz, bude v čo najväčšej miere využívať existujúce modely domény a aj celkový zámer integrálneho projektu - znovupoužiteľnosť riešenia pre iné informačné domény. Takéto ciele kladú vyššie nároky na úroveň návrhu architektúry a celkovej implementácie riešenia - s ohľadom na konfigurovateľnosť a prispôsobiteľnosť vytvoreného produktu.

Vašou úlohou bude na základe zozbierania požiadaviek a primeraného naštudovania modelov a technológii integrálneho projektu analyzovať, navrhnúť, implementovať a overiť portál pracovných príležitostí tak, aby pokrýval nie len základné funkcionálne požiadavky, ale aby bolo riešenie v čo najväčšej miere znovupoužiteľné pri implementácii iných podobných portálových riešení.

1. Úvod

Tento dokument obsahuje ponuku tímu č. 8 na tému Portál pracovných príležitostí vypracovanú v rámci predmetu Tvorba softvérového systému v tíme. Kapitola 2 sa venuje našej motivácii, ktorá zohrala významnú úlohu pri výbere témy a stručnému opisu skúseností členov tímu zo zameraním sa na kontext zvolenej témy.

Kapitola 3 sa venuje navrhovaným možnostiam riešenia portálu pracovných príležitostí a prípadným ohraničeniam, ktoré sme identifikovali. V závere stručne zhrnieme našu motiváciu a hlavné vlastnosti navrhovaného riešenia.

Preferencie tímu vzhľadom na ďalšie témy projektov sa nachádzajú v prílohe A, rozvrh jednotlivých členov tímu spolu s navrhovanými termínmi stretnutí je uvedený v prílohe B.

2. Motivácia a zloženie tímu

V tejto kapitole bližšie opíšeme našu motiváciu k riešeniu témy Portál pracovných príležitostí v rámci predmetu Tvorba softvérového systému v tíme. Následne predstavíme jednotlivých členov tímu so zameraním sa na skúsenosti súvisiace so zvolenou témou.

2.1. Motivácia

Pri výbere témy pre tímový projekt zavážilo mnoho skutočností, ktoré viedli k výberu témy Portál pracovných príležitostí. Významným dôvodom tohto výberu bol záujem všetkých členov tímu o technológie webu a všetko, čo s ním súvisí. K tomu sa pridala aktuálnosť problematiky webových portálov, webu so sémantikou a prispôsobovania. Práca na tomto projekte nám dáva príležitosť získať množstvo cenných skúseností z oblasti analýzy, návrhu a implementácie portálov s využitím najnovších prístupov a súčasne nám umožňuje zdokonaliť sa v tímovej spolupráci pri riešení projektov.

Druhým významným dôvodom, prečo sme sa rozhodli pre túto tému je skutočnosť, že projekt portálu je zasadený do širšieho kontextu väčšieho systému, ktorý práci na projekte dodáva ďalší rozmer. Kvalitné vypracovanie projektu otvára dvere jeho reálnemu nasadeniu do praxe v rámci štátneho programu výskumu a vývoja „Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov“. Možnosť pracovať na projekte od začiatku až po jeho reálne nasadenie do prevádzky je pre nás veľmi zaujímavá.

V neposlednom rade je zaujímavá aj konkrétna doména pracovných ponúk. Čoraz viac ľudí si hľadá prácu pomocou rôznych portálov, ktoré sprostredkúvajú pracovné ponuky a čoraz viac zamestnávateľov hľadá nových zamestnancov práve prostredníctvom takýchto portálov. Príležitosť navrhnúť a implementovať systém, ktorý bude dostatočne univerzálny, ale pritom lepší ako v súčasnosti používané riešenia predstavoval veľkú motiváciu pri výbere zadania.

Doviesť projekt uvedeného rozsahu do úspešného konca nie ľahké a vyžaduje si značné množstvo skúseností z rôznych oblastí. Pestrosť skúseností jednotlivých členov tímu, prechádzajúca naprieč celým spektrom najčastejšie používaných technológií v softvérovom inžinierstve, nám umožní nahliadať na riešenie z viacerých strán, hľadať alternatívy a úspešne ukončiť projekt.

2.2. Členovia tímu

Michal Barla

Je riešiteľom projektu „Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov“ štátneho programu výskumu a vývoja „Budovanie informačnej spoločnosti“, kde pôsobí v skupine „Personalized information presentation“. Podstatným spôsobom sa podieľal na vývoji doménovej ontológie pracovných ponúk pre tento projekt a teda dobre pozná doménu, v ktorej má produkt pôsobiť.

Ako diplomový projekt rieši zachytenie záujmov používateľa na webe, s overením v oblasti pracovných ponúk. Súčasne má v inžinierskom štúdiu zapísaný predmet Princípy webového inžinierstva, čo dokazuje jeho záujem o problematiku súvisiacu so zadaním. Priebežné výsledky diplomového projektu a nadobudnuté znalosti o webe by rád využil aj pri tvorbe produktu v rámci tímového projektu.

V rámci záverečného projektu bakalárskeho štúdia sa zúčastnil spolu s ďalšími troma študentmi medzinárodnej súťaže CSIDC, organizovanej CS IEEE, pri riešení ktorej získal

cenné skúsenosti s prácou v tíme a naučil sa okrem iného pracovať s webovými službami a databázovými technológiami. Ovláda programovacie jazyky C/C++ a C#, má skúsenosti s UML, XML, RDF a vytváraním ontológií v prostredí Protégé.

Peter Bartalos

Má skúsenosti s vývojom aplikácií v programovacích jazykoch Java, C a C++ a tiež s prácou v typografickom systéme TeX (LaTeX). K svojej práci pristupuje zodpovedne a s vysokým nasadením. Je schopný naštudovať veľké množstvo netriviálnych teoretických aj praktických poznatkov z rôznych oblastí, ktoré vie následne efektívne využiť k riešeniu problémov. Na konci bakalárskeho štúdia mu bol udelený pochvalný list dekana za vynikajúco vypracovanú záverečnú prácu. Mal príspevok na študentskej vedeckej konferencii IIT-SRC 2005 organizovanej na FIIT STU, kde získal ocenenie „One of The Best Papers“ a postúpil na medzinárodnú súťaž ACM Student Research Competition 2005. Navštevuje predmet Teória fuzzy systémov. Vedomosti z tohto predmetu chce využiť k zlepšeniu vyhľadávacích kritérií pre pracovné ponuky pomocou metódy neostrých (fuzzy) množín. V rámci predmetu Odborné praktikum 1, ktorý má zapísaný v rovnakom čase ako predmet Tvorba softvérového systému v tíme 1 bude mať prístup do softvérového štúdia, čo umožní tímu voľnejší prístup k jeho zdrojom.

Ján Porubský

Ukončil bakalárske štúdium v odbore Informatika. Počas školských projektov získal skúsenosti s návrhom internetových aplikácií, najmä ich databázovej časti (MySQL) a s vývojom aplikácií na platforme MS Windows. V mimoškolských projektoch sa zameriava na vývoj aplikácií v PHP a jazyk HTML. Ďalej má skúsenosti s jazykmi C/C++, Javascript a pracoval aj s Javou a Perlom. Ovláda prácu v prostrediach Borland Delphi, Borland C++ Builder. Zaujíma sa aj o návrh grafického dizajnu pre internetové aplikácie (programy GIMP, Corel Draw). Pri svojej práci sa snaží uprednostňovať platformu Unix, presnejšie Linux. Je schopný naštudovať potrebné informácie pre rôzne problémové oblasti, čo vychádza z jeho záujmu o oblasť informačných technológií.

Peter Sivák

Pracoval na projekte vývoja systému založenom na protokole XMPP, ktorý slúži na prenos správ vo formáte XML medzi softvérovými komponentmi. Projekt vypracoval na platforme J2EE. Výstupné údaje boli prezentované pomocou vektorovej grafiky vo formáte SVG. Ďalej napísal sériu článkov pre známy český webový portál o možnostiach, ktoré vývojárom poskytuje protokol XMPP. V týchto článkoch prezentoval opisované princípy na príkladoch v programovacom jazyku Java. Má veľmi dobrú znalosť programovacích jazykov C/C++, C#, technológie webových služieb, databázového servera MS SQL 2000, technológie ADO.NET a programovania MS CRM servera.

Minulý rok pracoval v tíme študentov riešiacich zadanie medzinárodnej súťaže CSIDC, organizovanej CS IEEE. Na školskej vedeckej konferencii získal tento tím ocenenie „Best paper“ medzi príspevkami študentov bakalárskeho štúdia a dostal sa do finále súťaže ACM Student Research Competition 2005. Počas riešenia projektu nadobudol hlbšie znalosti o platforme .NET.

V diplomovom projekte sa zaoberá návrhovými vzormi v prostriedkoch objektovo-relačného mapovania a pracuje s nástrojom NHibernate. V súčasnosti pracuje ako vývojár portálových riešení na platforme .NET.

Kristián Szobi

Je absolventom bakalárskeho štúdia na Fakulte informatiky a informačných technológii v študijnom odbore Softvérové inžinierstvo. Popri štúdiu pracoval vo viacerých softvérových spoločnostiach a získal bohaté skúsenosti s celým životným cyklom softvéru a pracou v tíme. Dôraz kladie na návrh modulárneho softvéru s využitím objektovo-orientovaného prístupu. Počas 4-ročnej praxe sa stretol s rôznymi technológiami (Java, webové služby, JDBC, .NET, nízkoúrovňové služby, mobilné zariadenia, MS SQL Server 2000). Množstvo skúseností získal aj na stáži v spoločnosti Microsoft v americkom Redmonde v lete tohto roku. V súčasnosti pracuje ako softvérový inžinier pre známu viedenskú spoločnosť.

Michal Tvarožek

Podieľa sa na riešení projektu „Nástroje pre získavanie, organizovanie a udržovanie znalostí v prostredí heterogénnych informačných zdrojov“, kde ako člen skupiny „Personalized information presentation“ pracuje na prezentácii a prispôsobovaní sa používateľovi. V rámci práce na projekte významným spôsobom prispel k tvorbe doménovej ontológie pre oblasť pracovných príležitostí, čím získal prehľad v problémovej oblasti a skúsenosti z oblasti ontológií.

Ako diplomový projekt rieši tému „Personalizovaná navigácia v priestore webu so sémantikou“, pričom skúsenosti z práce na projekte chce využiť aj v rámci tímového projektu. Navštevuje predmet Princípy webového inžinierstva, čo potvrdzuje jeho záujem o problematiku webu a webových portálov.

Bakalárske štúdium absolvoval s vyznamenaním a bolo mu udelené pochvalné uznanie dekana. Bol členom tímu študentov, ktorí reprezentovali fakultu na medzinárodnej súťaži CSIDC 2005 organizovanej CS IEEE a získal tak množstvo skúseností s prácou v tíme, s riadením a s tvorbou dokumentácie.

Popri štúdiu pracoval takmer štyri roky na viacerých projektoch v softvérovej spoločnosti, kde nadobudol praktické skúsenosti s tvorbou softvéru, prácou v tíme a rozličnými nástrojmi. Zameral sa najmä na platformu Win32 a PocketPC, ovláda programovacie jazyky C/C++ a C#.

3. Portál pracovných príležitostí

V tejto kapitole opíšeme nami navrhované riešenie portálu pracovných príležitostí. Vyjadríme sa ku komunikácii portálu s používateľmi a ku službám, ktoré im chceme prostredníctvom portálu poskytnúť. Ďalej sa budeme venovať zasadeniu portálu do širšieho kontextu systému nástrojov vytváraných v rámci štátneho programu, ich spolupráci a súladu použitých technológií a prostriedkov. Kapitolu zakončíme hrubým návrhom portálu.

3.1. Vlastnosti navrhovaného riešenia

Úspešné portálové riešenia sa sústreďujú na používateľa a prispôsobujú sa jeho potrebám a preferenciám. Používateľovi poskytujú jednoduchý a najmä efektívny prístup ku všetkým potrebným funkciám, pomocou ktorých je možné ľahko realizovať jednotlivé operácie. Celkovo sme identifikovali viacero vlastností a služieb portálu, ktoré chceme používateľovi sprístupniť. Uvádzame tie z nich, ktoré pokladáme za najpodstatnejšie a ktoré môžu najviac odlíšiť naše riešenie od klasických portálov pracovných ponúk.

Základnou funkciou portálu je sprostredkovanie rozhrania medzi používateľom a nástrojmi, ktoré tvoria samotné jadro systému. Používateľovi chceme dať možnosť vybrať si spôsob prezentácie ponúk a navigácie v nich, ktorý mu najviac vyhovuje, pričom predpokladáme spoluprácu s viacerými nástrojmi (fazetový prehliadač, mapy komunít). Na určenie konkrétnych údajov, ktoré budú prezentované a spôsobu, ktorým budú prezentované chceme použiť prezentačnú ontológiu (Fresnel). Architektúru systému chceme navrhnúť tak, aby bolo možné podporovať viacero výstupných formátov (XHTML, PDF, PS, WML, SVG, ZIP).

Portál konzumentovi ponúk poskytne možnosť vyhľadávania v štruktúrovaných dátach, uložených v ontológii, pravdepodobne s využitím nástroja TopK agregátor alebo pomocou fazetového vyhľadávania. Uvažujeme tiež nad možnosťou vyhľadávania s využitím teórie neostrých (fuzzy) množín. Samozrejmou je možnosť fulltextového vyhľadávania, buď nad celou množinou ponúk alebo nad niektorou jej časťou (určenou napr. pomocou fazetového prehliadača).

Významnou vlastnosťou, ktorú chceme používateľovi poskytnúť nad rámec doteraz uvedených možností je adaptabilita a adaptivita portálu na základe ontológie používateľa. Portál chceme prepojiť s nástrojmi na získavanie znalostí o používateľovi a následnú tvorbu modelu používateľa, ktorý využijeme na lepšie prispôsobenie sa portálu potrebám jednotlivých používateľov. Chceme riešiť najmä personalizované usporiadanie a triedenie prezentovaných pracovných ponúk avšak model používateľa možno využiť aj k prispôsobeniu vzhľadu portálu a prispôsobeniu navigácie medzi jednotlivými stránkami. Uvažovali sme tiež o personalizovanom informovaní používateľa o relevantných zmenách na portáli s využitím emailu alebo SMS.

Významnou časťou používateľov portálu budú aj tvorcovia ponúk, ktorým chceme poskytnúť aj alternatívny spôsob prístupu k portálu pomocou webových služieb, ktoré umožnia jednoduchšie prepojenie portálu s ich vlastnými informačnými systémami. Očakávame, že prístup k portálu prostredníctvom webových služieb bude zaujímavý najmä pre pracovné agentúry alebo väčšie firmy.

Využitie ontológií a prispôsobovania dáva priestor pre tvorbu rôznych štatistík, ktoré môžu byť užitočné pre producentov aj konzumentov ponúk. Možností je naozaj veľa (napr. „priemerný“ profil používateľa, ktorý si prezeral ponuku, štatistiky pre rôzne skupiny ponúk, konzumentov, producentov).

3.2. Portál v kontexte väčšieho systému

Portál pracovných príležitostí je len jednou časťou väčšieho systému, ktorý okrem samotného portálu zahŕňa aj celú sadu ďalších nástrojov na získavanie, udržovanie a organizovanie znalostí. Pri návrhu a následnej implementácii portálu preto bude zohrávať veľkú úlohu aj súhra a spôsob komunikácie s týmito nástrojmi. Z tohto dôvodu sme sa rozhodli vytvoriť portál v rovnakom prostredí v akom sú vyvíjané ostatné nástroje a pomocou podobných vývojových prostriedkov.

Celý portál chceme riešiť v prostredí Linux, pričom za implementačný jazyk sme si vybrali jazyk Java verzie 5, v ktorom budú implementované aj jednotlivé nástroje. Komunikácia s nimi bude prebiehať cez dohodnuté rozhrania a prostredníctvom relačnej a ontologickej databázy. Na správu zdrojových súborov a dokumentácie budeme využívať nástroj Subversion, ako dátové úložiská chceme použiť Sesame pre ontologické dáta a PostgreSQL alebo MySQL pre relačné dáta. Na zabezpečenie kvality systému budeme pri vývoji používať testovaciu metodológiu „unit testing“ s využitím nástroja JUnit. Webovú časť by sme chceli riešiť pomocou nástrojov Apache, Tomcat a rámca Cocoon. Súčasne pre dostatočne rýchle fungovanie portálu predpokladáme využitie rovnakých hardvérových prostriedkov ako vyžadujú ostatné časti systému.

Cieľom štátneho programu je vytvoriť všeobecné nástroje na prácu so znalosťami, pričom ich charakter sa navyše môže (bude) časom meniť, čo sa prejaví najmä v zmenách doménovej ontológie. V tomto duchu chceme portál vyvinúť modulárne s dostatočne všeobecnými a modifikovateľnými komponentmi, aby bol využiteľný aj v iných problémových oblastiach len s malými zmenami. Zároveň chceme navrhnúť prezentačnú časť tak, aby v maximálnej možnej miere využívala údaje z ontológie a vedela sa prispôsobiť aspoň menším zmenám aj bez priameho zásahu do zdrojových kódov.

3.3. Hrubý návrh

V návrhu portálu počítame s jeho rozdelením na **administračnú** a **prezentačnú** časť. Administračná časť bude určená na úpravy v štruktúre portálu, na zmeny šablón, podľa ktorých sa bude generovať používateľské rozhranie a na manažment používateľov. Administračná časť ďalej umožní zvýšiť nezávislosť navrhovaného portálového riešenia na doméne pracovných príležitostí tým, že umožní zmenu zobrazovaných informácii bez zmeny kódu aplikácie.

Portál sa dá na najvyššej úrovni abstrakcie dekomponovať na dva podsystémy. Jeden poskytuje funkcionalitu **zadávateľovi ponuky** a druhý jej **konzumentovi**. Zadávateľ ponuky môže byť zamestnávateľ, ktorý ponúka prácu, ale takisto aj uchádzač o zamestnanie, ktorý ponúka svoje vedomosti a zručnosti.

Architektúru navrhovaného systému je najjednoduchšie opísať pomocou toku dát. Dáta sú uložené v dátovom úložisku, ktoré tvorí back-end aplikácie. Pri požiadavke na zobrazenie webovej stránky sa z dátového úložiska vyberú relevantné informácie a pomocou série transformácií sa prevedú do podoby vhodnej pre daného používateľa. Systém bude pri spracovaní požiadavky používať **architektonický vzor dátovod**, ktorý je implementovaný napr. v prezentačnom rámci Apache Cocoon.

Dôležitou vlastnosťou navrhovaného systému je dôkladné oddelenie samotných dát od ich spôsobu zobrazenia. Kým dáta sa získajú z dátového úložiska, spôsob zobrazenia bude definovaný **transformačnými šablónami**.

Oddelenie komponentov používateľského rozhrania, ktoré zabezpečujú spôsob reakcie na vstup používateľa od samotných dát, systém zabezpečí pomocou návrhového vzoru **Pozorovateľ** (Observer). Subjektom bude v tomto prípade dátové úložisko a pozorovateľmi

komponenty používateľského rozhrania. Pri zmene dát v úložisku jedným komponentom budú upozornené aj ostatné, ktoré sú prihlásené na odber daných informácií.

Na definovanie zobrazovaných formulárov bude použitý jazyk **XForms**, vďaka ktorému sa tiež zníži závislosť systému na doméne.

Informácie zobrazované portálom sa budú často meniť. Preto je nevyhnutné zabezpečiť, aby mal používateľ zobrazené vždy tie najaktuálnejšie. Obmedzenie prenosového protokolu webových aplikácii (HTTP), ktorý priamo neumožňuje asynchrónne vyžiadanie zmeny zobrazovaných informácií zo strany servera, chceme riešiť pomocou technológie **AJAX** (Asynchronous JavaScript and XML).

4. Záver

Počas stretnutí tímu sme identifikovali množstvo podnetných nápadov a možností riešenia témy Portál pracovných príležitostí, z ktorých mnohé sa do tohto dokumentu nedostali kvôli jeho obmedzenému rozsahu. Spomedzi nich sme uviedli tie črty portálu, ktoré ho robia výrazne pokrokovejším oproti klasickým portálovým riešeniam. Je to najmä využitie rôznych spôsobov prezentácie údajov na základe ich sémantiky, využitie webových služieb ako alternatívneho rozhrania portálu, zameranie sa primárne na potreby používateľov portálu pomocou adaptivity. Nezanedbateľnou vlastnosťou portálu je tiež jeho prevádzka v jednotnom prostredí spolu s ostatnými nástrojmi a dostatočná odolnosť voči zmenám v ontológii.

Náš tím je zložený z motivovaných ľudí, ktorí sú odhodlaní naplno využiť svoje znalosti, nasadenie a tímového ducha v prospech projektu od jeho hrubého návrhu až po jeho úspešné zakončenie. Motiváciou je pre nás nielen možnosť dôverne sa zoznámiť s aktuálnymi technológiami a nástrojmi, ale aj presvedčiť sa, že vieme spolu vytvoriť jednoliaty celok, ktorý dokáže vyrobiť vynikajúci produkt a súčasne plnohodnotnú súčasť štátneho programu.

Príloha A – Preferencie tímu

1. Portál pracovných príležitostí (JOBS)
2. Obaľovač na získavanie pracovných ponúk (WRAPPER)
3. Báza znalostí a zručností študentov (ZNALOSTI)
4. Distribuovaná simulácia rozsiahlych počítačových sietí (SIMUL)
5. Robocup – tretí rozmer (RoboCup 3D)
6. Tvorba rozvrhov (ROZVRH)
7. RoboCup – nové stratégie (RoboCup S)
8. Kandidát na najlepší multimediálny produkt roku 2006 (EuroPrix)
9. Nástroj na modelovanie vlastností (FEATURES)
10. Systém pre analýzu a animáciu chôdze človeka (ANIM)

Príloha B – Rozvrh členov tímu

Hodina	1	2	3	4	5	6	7	8	9	10	11	12
Začiatok	7:20	8:15	9:15	10:10	11:10	12:05	13:05	14:00	15:00	15:55	16:55	17:50
Koniec	8:10	9:05	10:05	11:00	12:00	12:55	13:55	14:50	15:50	16:45	17:45	18:40
Pondelok	APS1 JP			NS MB, KS, MT			OOANS MB, KS, PB, MT		OOANS MB, KS, PB, MT		TP všetci	
Utorok			Stretnutie č. 3			Stretnutie č. 2			PWI MB, PS, MT		PWI MB, PS, MT	
Streda	Stretnutie č. 1			Návrh prekladačov JP			ZK KS	TFS JP, PB			PeWe MB, MT	
Štvrtok			ASS všetci				NS / TFS / ZK MB, MT / JP, PB / KS		MSI všetci		MSI všetci	
Piatok	JP mimo BA, KS má DOS 31.10., 11.11, a 2.12.											

Legenda:

MB Michal Barla

PB Peter Bartalos

JP Ján Porubský

PS Peter Sivák

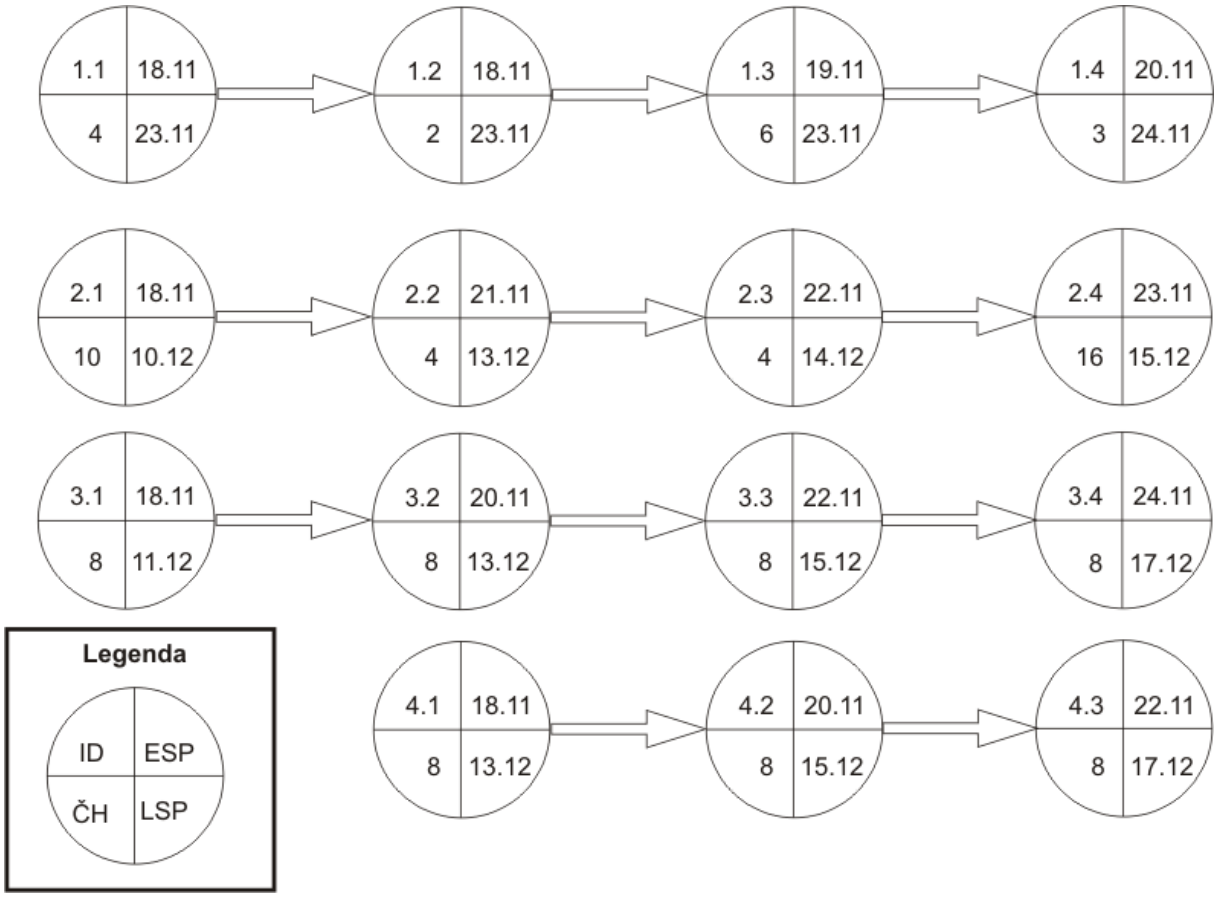
KS Kristián Szobi

MT Michal Tvarožek

Preferujeme stretnutia v uvedenom poradí (najprv č. 1, potom 2 a 3).

3. Plán projektu

Tento dokument predstavuje plán tímu pre obdobie 18.11.2005 – 19.12.2005 a rámcový plán do konca letného semestra. Obdobie 18.11.2005 – 19.12.2005 je určené na vypracovanie posudku projektovej dokumentácie tímu číslo 10 a vytvorenie prototypu. V priloženej tabuľke sú uvedené jednotlivé úlohy a činnosti. Obr. 1 zobrazuje tieto činnosti v grafe.



Obr. 1 Rozvrh činností

Číslo	Dátum	Čas	Michal Barla	Peter Bartalos	Ján Porubský	Peter Sivák	Kristian Szobi	Michal Tvarožek	Výstupy	Úloha
			MB	PB	JP	PS	KS	MT		
S3	26.10.2005	7:30	x	x	x	x	x	x	5" prezentácia	Overenie spôsobu práce a zvyknutie si na týždňový cyklus stretnutí a prezentácie výsledkov
			x	x	x	x	x	x	5" prezentácia	Metodiky na MSI
			x						prezentácia	Prezentácia doménovej ontológie, prvotná analýza domény
				x					prezentácia	Java, editor na naplnenie testovacích údajov
					x				návrh webu	Vytvoriť návrh webu tímu
						x			prezentácia	Prvá verzia analýzy nástrojov Tomcat, Cocoon, Sesame
							x		HowTo	Junit
								x	prezentácia	Prezentácia doménovej ontológie, prvotná analýza domény
								x	doc	Zápis zo stretnutia
								x	xls	Plán projektu
S4	2.11.2005	7:30	x	x	x	x	x	x	doc/visio/...	Odovzdanie prvých verzií príspevkov do dokumentácie pre O2
			x						doc/visio/...	Analýza problémovej oblasti a špecifikácia
				x					editor + prez.	Java, editor na naplnenie testovacích údajov
					x				návrh webu	Vytvoriť návrh webu portálu
						x			doc + prez.	Dopracovanie analýzy + nástroje Jboss, Orbeon, Jena
							x		súbor + prez.	Hrubý návrh architektúry a rozhraní
								x	doc/visio/...	Analýza problémovej oblasti a špecifikácia
IO	7.11.2005	24:00	x	x	x	x	x	x	doc/visio/...	Odovzdanie častí dokumentácie za každého riešiteľa pre S5
S5	9.11.2005	7:30	x	x	x	x	x	x	doc/visio/...	Beta1 dokumentácie pre O2
IO	13.11.2005	24:00	x	x	x	x	x	x	doc/visio/...	RC1 dokumentácie pre O2
S6	16.11.2005	7:30	x	x	x	x	x	x	doc/visio/...	RC2 dokumentácie pre O2
O2	18.11.2005	14:00	x	x	x	x	x	x	?	Odovzdanie dokumentácie analýzy problému, špecifikácie požiadaviek riešenia spolu s hrubým návrhom
S7	23.11.2005	7:30							doc	RC1 dokumentácie pre O3
O3	25.11.2005	14:00	x	x	x	x	x	x	?	Odovzdanie posudku analýzy, špecifikácie a hrubého návrhu iného tímu
IO	27.11.2005	24:00	x	x	x	x	x	x	doc/visio/...	Odovzdanie prvých verzií príspevkov do dokumentácie pre O4
S8	30.11.2005	7:30	x	x	x	x	x	x		Integrácia Alpha1 prototypu, prvá verzia dokumentácie
IO	4.12.2005	24:00	x	x	x	x	x	x	doc/visio/...	Odovzdanie častí dokumentácie za každého riešiteľa pre S9
S9	7.12.2005	7:30	x	x	x	x	x	x		Integrácia Beta1 prototypu, Beta1 verzia dokumentácie
IO	11.12.2005	24:00	x	x	x	x	x	x	doc/visio/...	RC1 dokumentácie pre O4
S10	14.12.2005	7:30	x	x	x	x	x	x		Odskúšanie RC1 prototypu a dokumentácie pre O4
O4	19.12.2005	14:00	x	x	x	x	x	x		Odovzdanie prototypu vybraných častí systému spolu s dokumentáciou
O5	21.12.2005		x	x	x	x	x	x		Používateľská prezentácia prototypu
O6	2.2.2006	14:00	x	x	x	x	x	x		Odovzdanie posudku prototypu iného tímu
	16.2.2006		x	x	x	x	x	x		Produkt, verzia Beta1, kostra dokumentácie
	2.3.2006		x	x	x	x	x	x		Produkt, verzia Beta2, Beta1 dokumentácie
	23.3.2006		x	x	x	x	x	x		Integračné testovanie produktu
	6.4.2006		x	x	x	x	x	x		Odovzdanie produktu a dokumentácie k produktu
	20.4.2006		x	x	x	x	x	x		Odovzdanie celkového výsledku projektu (produkt so zmenami v rámci údržby, dokumentácia)

Číslo	Dátum	Čas	Michal Barla	Peter Bartalos	Ján Porubský	Peter Sivák	Kristian Szobi	Michal Tvarožek	Výstupy	Úloha
			MB	PB	JP	PS	KS	MT		
O6	2.2.2006	14:00	x	x	x	x	x	x		Odovzdanie posudku prototypu iného tímu
	16.2.2006		x	x	x	x	x	x		Produkt, verzia Beta1, kostra dokumentácie
	2.3.2006		x	x	x	x	x	x		Produkt, verzia Beta2, Beta1 dokumentácie
	23.3.2006		x	x	x	x	x	x		Integračné testovanie produktu
S16	28.3.2006	12:00								Vyhodnotenie stavu riešenia projektu, definovanie priorít a úloh na najbližšie obdobie Outline dokumentácie (technická, používateľská) Klikateľné sídlo portálu podľa navigačnej štruktúry, hoci aj s prázdnymi stránkami Funkčný aspoň jednoduchý formulár (pracovnej ponuky) s celým životným cyklom Definované meta-dáta pre opis neontologických formulárov Jednoduchá verzia Facticu pre neontologické dáta Štýly pre zobrazovanie jednotlivých stránok portálu (CSS) Prihlasovanie a manažment používateľov (s využitím Facticu pre neontologické dáta?)
S17	4.4.2006	12:00	x	x	x	x	x	x		Vyhodnotenie stavu riešenia jednotlivých častí produktu a dokumentácie
S18	11.4.2006	12:00	x	x	x	x	x	x		Integrácia a testovanie Beta verzie produktu a dokumentácie
S19	18.4.2006	12:00	x	x	x	x	x	x		Integrácia a testovanie RC1 produktu a dokumentácie
S20	25.4.2006	12:00	x	x	x	x	x	x		Integrácia a testovanie RC2 produktu a dokumentácie
O7	28.4.2006	13:00	x	x	x	x	x	x		Odovzdanie produktu a dokumentácie k produktu
S21	2.5.2006	12:00	x	x	x	x	x	x		Vyhodnotenie CR od zadávateľa, pokračovanie v generovaní formulárov
S22	9.5.2006	12:00	x	x	x	x	x	x		Integrácia a testovanie v2RC1 produktu a dokumentácie
S23	16.5.2006	12:00	x	x	x	x	x	x		Integrácia a testovanie v2RC2 produktu a dokumentácie
O8	22.5.2006	13:00	x	x	x	x	x	x		Odovzdanie celkového výsledku projektu (produkt so zmenami v rámci údržby, dokumentácia)
S24	23.5.2006	12:00	x	x	x	x	x	x		Posudok produktu iného tímu - outline, rozdelenie úloh
O9	5.6.2006	13:00	x	x	x	x	x	x		Odovzdanie posudku na produkt iného tímu
S26	30.5.2006		x	x	x	x	x	x		Príprava prezentácie a obhajoby projektu
S27	6.5.2006		x	x	x	x	x	x		Skúška prezenácie a obhajoby projektu
O10	12.6.2006		x	x	x	x	x	x		Prezentácia a obhajoba projektu
Sx	13.6.2006		x	x	x	x	x	x		Oslava úspešného absolvovania TP :)

4. Úlohy členov tímu

4.1. Roly v tíme

Počas riešenia sme v tíme rozdelili zodpovednosti za jednotlivé časti riadenia nasledovným spôsobom:

Vedúci tímu	Michal Tvarožek
Manažér vývoja	Peter Sivák
Manažér kvality	Kristián Szobi
Manažér plánovania	Michal Barla
Manažér PR	Ján Porubský
Manažér podporných činností	Peter Bartalos

Z hľadiska práce na projekte sme v období do prvého kontrolného bodu rozdelili úlohy nasledovným spôsobom (zachytené je len primárne rozdelenie úloh):

Analýza	Všetci
Špecifikácia	Michal Barla
Návrh	Peter Sivák, Kristián Szobi, Ján Porubský
Implementácia	Peter Bartalos
Dokumentácia	Všetci
Integrácia dokumentácie	Michal Tvarožek
Web	Ján Porubský

4.2. Rozdelenie práce na projektovej dokumentácii

Štruktúra a integrácia dokumentácie: Michal Tvarožek

0.	Úvod	Michal Tvarožek
1.	Opis riešeného problému	Michal Tvarožek
2.	Analýza problémovej oblasti	Michal Barla (2.1, 2.3.1) Michal Tvarožek (2.2) Ján Porubský (2.3.2)
3.	Analýza riešenia	Michal Barla (3.7) Peter Sivák (3.1, 3.2) Peter Bartalos (3.3, 3.6, 3.8) Ján Porubský (3.5) Kristián Szobi (3.4)
4.	Špecifikácia riešenia	Michal Barla
5.	Hrubý návrh riešenia	Kristián Szobi (5.1, 5.6) Peter Sivák (5.1, 5.2) Ján Porubský (5.3, 5.4) Michal Tvarožek (5.5)
A	Pravidlá pre návrh portálu pre zrakovo postihnutých používateľov	Ján Porubský
B	Nízko-úrovňové rozdiely rámca Cocoon a prezentačného serveru Orbeon	Kristián Szobi
	Riadenie	Michal Barla, Michal Tvarožek

4.3. Organizovanie práce v tíme

Okrem pravidelných stretnutí, na ktorých vyhodnocujeme staré a zadávame nové úlohy využívame aj iné spôsoby komunikácie a koordinácie práce v tíme:

- Vytvorili sme mailový alias tímu – pri počte šiestich ľudí je elektronická pošta efektívnym prostriedkom komunikácie.
- Dodržiavame neformálnu dohodu o využití servera `labss2` ako spoločného úložiska súborov.
- Využívame IM technológie na rýchlu komunikáciu
- Využívame podporný nástroj pre manažment projektu `Basecamp` (dostupný z webovej stránky tímu)

4.4. Úlohy v tíme a dokumentácia v etape prototypovania

Z hľadiska práce na projekte sme v období prototypovania rozdelili úlohy nasledovným spôsobom (zachytené je len primárne rozdelenie úloh):

Michal Barla	CForms, generovanie formulárov, flow
Peter Bartalos	Práca s úložiskom Sesame, Java Beans
Peter Sivák	Coplety, Cocoon
Kristián Szobi	Cocoon + Jetspeed
Ján Porubský	Jetspeed
Michal Tvarožek	CForms, dokumentácia

Štruktúra a integrácia dokumentácie: Michal Tvarožek

	Úvod, zhodnotenie	Michal Tvarožek
6.1	Analýza rizík	Michal Tvarožek
6.2	Enterprise Java Bean	Peter Sivák
6.3	Sesame	Peter Bartalos
6.4	Reprezentácia formulárov	Michal Barla Peter Sivák
6.5	Zobrazenie formulárov	Kristián Szobi Peter Sivák Michal Barla
6.6	Zhodnotenie	Michal Tvarožek
C	Technická dokumentácia	Michal Barla, Michal Tvarožek
	Riadenie	Michal Barla, Michal Tvarožek

4.5. Zhrnutie procesov riadenia tímu v letnom semestri

V letnom semestri sme pokračovali v riešení s pôvodným rozdelením manažérskym úloh v tíme, aj napriek tomu, že viaceré z nich sa v minulosti nepodarilo dostatočne zabezpečiť. Viacerí členovia tímu prejavili záujem pokračovať v plnení svojich predchádzajúcich úloh a zlepšení doterajších výsledkov (napr. manažment vývoja a kvality). Približne mesiac pred koncom semestra z tímu odišiel Ján Porubský bez bližšieho udania dôvodov.

4.5.1. Manažment v projekte

Pri riešení tímového projektu sme získali množstvo skúseností s riadením (a neriadením) projektu. Počas projektu riešili veľké množstvo implementačných záležitostí a následne na manažment nezostalo potrebné množstvo zdrojov. Celkovo sa nám nepodarilo v potrebnom rozsahu zabezpečiť manažment plánovania, vývoja a kvality.

4.5.2. Práca na vývoji aplikácie

Práce na vývoji portálu sme po cca. štyroch týždňoch rozdelili na dve skupiny – samotný portál a generovanie formulárov. Pri vývoji sa obe skupiny dopĺňali, ale hlavný dôraz sa kládol na základnú portálovú časť riešenia (Ján Porubský, Kristián Szobi, Michal Tvarožek). Skupina generovania formulárov sa sústredila (polo)automatickú prípravu opisu formulárov pre portál a prípravu algoritmu generovania formulárov (Michal Barla, Peter Bartalos, Peter Sivák).

S blížiacim sa odovzdaním produktu sa spolupráca skupín zintenzívnila, práce na generovaní boli utlmené a získané zdroje boli investované do dokončenia portálovej časti. Po odovzdaní prvej verzie produktu sme opäť zintenzívnili prácu na generovaní formulárov (Michal Barla, Michal Tvarožek, Peter Sivák). Ostatní riešitelia sa sústredili na dopracovanie funkcionality a nedostatkov objavených v portáli (Peter Bartalos, Kristián Szobi).

Z hľadiska úloh sa každý člen tímu podieľal na návrhu, implementácii a dokumentovaní jednotlivých častí riešenia.

4.5.3. Práca na dokumentácii

Na dokumentovaní riešenia sa zúčastnili všetci riešitelia. Každý riešiteľ v rámci svojich možností zdokumentoval „svoje“ časti (technická dokumentácia) a tiež vypracoval časť do dokumentácie o riešení projektu. Jednotlivé časti dokumentácie prešli viacerými iteráciami pripomienok a dopracovaní od autorov, posudzovateľa (Michal Barla) a integrátora (Michal Tvarožek).

Výslednú dokumentáciu integroval vedúci integrátor (Michal Tvarožek) do finálnej podoby. Používateľskú dokumentáciu vytvoril Michal Tvarožek, integráciu technickej dokumentácie o zdrojových kódoch – JavaDoc zabezpečil Michal Barla.

4.5.4. Organizovanie práce v tíme

Počas riešenia sme sa pravidelne stretávali s pedagogickým vedúcim na spoločných stretnutiach tímu. Na organizáciu práce sme využili aj stretnutia v menších skupinách, a nasledovné dodatočné spôsoby:

- Dodržiavame neformálnu dohodu o využití servera `labss2` ako spoločného úložiska súborov, ktorého použitie s nasadením SVN prestalo byť významné. Problémy vznikli počas dlhej doby nefunkčnosti servera `labss2`.
- Intenzívne sme využívali IM technológie na rýchlu komunikáciu.
- Po vyhodnotení použitia nástroja pre manažment projektu `Basecamp` sme chceli vyskúšať aj iné nástroje. Pokúsili sme sa použiť `DotProject`, ale členovia tímu ho nevyužívali v potrebnom rozsahu takže bol pre manažment projektu nepoužiteľný.
- Používali sme `Subversion` na ukladanie a verziovanie súborov.

5. Zápisy zo stretnutí

Zápis zo stretnutia k tímovému projektu 1/2005

Dátum: 3.10.2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek

Miesto: FIIT

Zapísal: Tvarožek

Doplnil:

Náplň stretnutia a závery z diskusie

Na spoločnom stretnutí všetkých tímov sme získali dodatočné informácie k tímovým projektom a ponukám. Dohodli sme sa na definitívnom poradí tém tímových projektov o ktoré máme záujem. Konzultovali sme tému projektu „JOBS“ – portál pracovných príležitostí s Romanom Filkornom.

Následne sme na samostatnom stretnutí členov tímu diskutovali o webovom portály pracovných príležitostí a o prostredí v ktorom bude umiestnený. Diskutovali sme tiež viaceré možnosti riešenia projektu a ďalší postup pri vypracovaní ponuky.

Plán ďalšej činnosti

Termín ukončenia riešitelia	Opis úlohy
4.10.2005 všetci	Príprava na písanie ponuky; spísanie vlastných skúseností, myšlienok a nápadov
4.10.2005 Barla, Tvarožek	Vypracovanie prvej verzie ponuky
4.10.2005-10.10.2005 všetci	Práca na ponuke, príprava prezentácie
10.10.2005	Odovzdanie a prezentácia ponuky

Ďalšie stretnutie

Neformálne stretnutie je predbežne naplánované na stredu 5.10.2005, presný čas bude upresnený neskôr.

Zápis zo stretnutia k tímovému projektu 2/2005

Dátum: 18.10.2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio, CD35

Zapísal: Tvarožek

Doplnil:

Náplň stretnutia a závery z diskusie

Toto bolo prvé stretnutie s vedúcim projektu po pridelení témy. Na začiatku vedúci projektu okomentoval vypracovanú ponuku, s následnou diskusiou. V ďalšom sme diskutovali o organizačných záležitostiach a výstupoch, ktoré bude potrebné v priebehu riešenia projektu vytvoriť. V tejto súvislosti sme diskutovali aj o metodikách, ktoré jednotliví členovia tímu vypracúvajú v rámci predmetu Manažment v softvérovom inžinierstve. Podľa povahy jednotlivých úloh sme ich naplánovali na jeden, resp. dva týždne. Výsledkom vypracovania jednotlivých úloh bude 1-2 stranová analýza úlohy a krátka prezentácia výsledkov ostatným členom tímu.

Predbežne sme si zadelili roly v tíme a úlohy na ďalšie obdobie. Riešili sme aj otázky komunikácie a riadenia tímu. Každý účastník dostal prihlasovacie údaje do webového systému určeného na organizáciu tímu.

Plán budúcich úloh

Termín ukončenia riešitelia	Opis úlohy
27.10.2005 Všetci	Vypracovanie metodiky na MSI a jej krátka prezentácia ostatným členom tímu
26.10.2005 2.11.2005 Ján Porubský	Vytvoriť návrh webu tímu Vytvoriť návrh webu portálu
19.10.2005 26.10.2005 2.11.2005 Michal Barla	Prezentácia doménovej ontológie ostatným členom tímu Prvotná analýza problémovej oblasti Analýza problémovej oblasti a špecifikácia
19.10.2005 26.10.2005 2.11.2005 Michal Tvarožek	Prezentácia doménovej ontológie ostatným členom tímu Vytvoriť plán projektu, pripraviť zápis zo stretnutia, prvotná analýza problémovej oblasti Analýza problémovej oblasti a špecifikácia

26.10.2005 2.11.2005 Peter Sivák	Prvá verzia analýzy Java servletov (Tomcat/JBoss) a ontologického úložiska (Sesame/Jena) Hrubý návrh systému + dopracovanie analýzy
26.10.2005 2.11.2005 Kristián Szobi	Analýza JUnit, práca na hrubom návrhu architektúry a rozhraní Hrubý návrh architektúry a rozhraní, pozrieť si Subversion
26.10.2005 2.11.2005 Peter Bartalos	Analýza nástroja Cocoon Analýza nástroja Orbeon + porovnanie s Cocoonom

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na budúci týždeň v utorok alebo stredu. Termín stretnutia bude ešte do pondelka upresnený vedúcim projektu.

Zápis zo stretnutia k tímovému projektu 3/2005

Dátum: 26.10.2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Barla

Doplnil:

Vyhodnotenie úloh z predchádzajúceho stretnutia

Termín ukončenia riešitelia	Opis úlohy	Stav plnenia
27.10.2005 Všetci	Vypracovanie metodiky na MSI a jej krátka prezentácia ostatným členom tímu	Splnená 100%
26.10.2005 2.11.2005 Ján Porubský	Vytvoriť návrh webu tímu Vytvoriť návrh webu portálu	Splnená 100%
19.10.2005 26.10.2005 2.11.2005 Michal Barla	Prezentácia doménovej ontológie ostatným členom tímu Prvotná analýza problémovej oblasti Analýza problémovej oblasti a špecifikácia	Splnená 100% Splnená 100%
19.10.2005 26.10.2005 2.11.2005 Michal Tvarožek	Prezentácia doménovej ontológie ostatným členom tímu Vytvoriť plán projektu, pripraviť zápis zo stretnutia, prvotná analýza problémovej oblasti Analýza problémovej oblasti a špecifikácia	Splnená 100% Splnená 100%
26.10.2005 2.11.2005 Peter Sivák	Prvá verzia analýzy Java servletov (Tomcat/JBoss) a ontologického úložiska (Sesame/Jena) Hrubý návrh systému + dopracovanie analýzy	Splnená 100%
26.10.2005 2.11.2005 Kristián Szobi	Analýza JUnit, práca na hrubom návrhu architektúry a rozhraní Hrubý návrh architektúry a rozhraní, pozrieť si Subversion	Splnená 100%
26.10.2005 2.11.2005 Peter Bartalos	Analýza nástroja Cocoon Analýza nástroja Orbeon + porovnanie s Cocoonom	Splnená 100%

Náplň stretnutia a závery z diskusie

Na začiatku stretnutia sa prejednalo plnenie jednotlivých úloh, vyplývajúcich z predchádzajúceho stretnutia. Všetky úlohy, vytýčené na toto stretnutie boli splnené.

Každý člen tímu v potrebnej miere prezentoval svoju metodiku na predmet MSI a spôsob, akým ju využijeme v tímovom projekte.

Kristián pred stretnutím poskytol tímu materiál o unit testovaní s využitím rámca JUnit, Peter Sivák poskytol dokument – úvod do technológií Tomcat, Sesame a Cocoon.

Jano odprezentoval návrh webovej stránky tímu, ktorý bol odsúhlasený všetkými prítomnými.

Michal Tvarožek predstavil plán projektu, ktorý sa skladá z podrobného plánu na najbližšie obdobie a rámcového plánu, určeného termínmi odovzdania častí riešenia.

Roman nás oboznámil s požiadavkami na prvý výstup tímu. Tím musí vypracovať dva dokumenty – správu o projekte spolu s technickou dokumentáciou a správu o riešení projektu. Správa o projekte by mala obsahovať stručnú analýzu domény s prípadnou odvolávkou na analýzu vo výstupnom dokumente fázy 2 štátneho programu. Ďalej by mala byť uvedená analýza portálových riešení a techník adaptivity a adaptability. V dokumente by mala byť ďalej špecifikácia produktu a hrubý návrh systému – ideálnym výsledkom je sada spolupracujúcich nástrojov. Technická dokumentácia by sa mala vzťahovať k prototypu, ktorý s najväčšou pravdepodobnosťou **nebude** na zahodenie a tím si na ňom sprototypuje rizikovejšie časti projektu – generovanie formulárov z ontológie,...

Dohodli sme sa, že výstupy úloh sa budú ukladať do úložiska na server labss2, členom tímu sa vždy pošle mailom notifikácia o novom dokumente v úložisku. Výstupy úloh sa budú písať s vedomím, že sa z nich na konci má poskladať celistvý dokument a budú sa postupne dolaďovať.

Následne sa rozprúdila diskusia k ontológiam všeobecne (význam, výhody) a aj konkrétne k ontológií pracovných ponúk. Z diskusie vyplynuli návrhy na drobné zmeny a vylepšenia ontológie, príp. otázky:

- pridať požiadavku na pohlavie
- pridať požiadavku na vek
- atribút text dať s iným namespace, keďže je viac technický a nepatrí priamo do pracovnej ponuky.
- Vyriešiť usporiadanie enumerations, TravellingInvolved – cez vlastnosti
- Vyriešiť usporiadanie EducationLevel – OWL Full problem obísť vytvorením „dummy“ inštancií EducationLevel
- Je nám jasné, ako nástroj TopK agregátor zoraďuje podľa atribútu plat – ako to však rieši napr. pri atribúte TravelingInvolved? Bolo by vhodné, keby vedel triediť nielen na základe triedy, ktorá predstavuje atribút, ale na základe inštancie danej triedy. Treba sa spýtať autora nástroja v Košiciach.

Diskutovali sme aj o SesameAPI a spôsobe, akým budeme pristupovať k ontologickému úložisku. Využijeme štandardné API pre JAVU a dopytovací jazyk SeRQL.

Plán budúcich úloh

Termín ukončenia riešitelia	Opis úlohy
2.11.2005 Ján Porubský	Sprevádzkovať stránku na serveri labss2 Urobiť analýzu nástrojov na navrhovanie web formulárov (Dreamweaver a pod) Vytvoriť návrh webu portálu – technologický aj vizuálny
2.11.2005 Michal Barla	Analýza problémovej oblasti a špecifikácia – 1. outline dokumentu v úložisku Zmeny v doménovej ontológii podľa záverov diskusie na treťom stretnutí
2.11.2005 Michal Tvarožek	Analýza problémovej oblasti a špecifikácia – 1. outline dokumentu v úložisku Zmeny v doménovej ontológii podľa záverov diskusie na treťom stretnutí
2.11.2005 Peter Sivák	Analýza nástroja Orbeon + porovnanie s Cocoonom, pokračovanie v analýze Java servletov (Tomcat/JBoss) a ontologického úložiska (Sesame/Jena) – verzia v úložisku
2.11.2005 Kristián Szobi	Napísať úvod do architektúry – 1. outline dokumentu v úložisku Spustiť na serveri systém Bugzilla a vytvoriť pre tím (alebo na nejaký nasmerovať) tutoriál o jeho používaní.
2.11.2005 Peter Bartalos	Práca na editore ontológie Vytvoriť malý tutoriál pre tím (alebo na nejaký nasmerovať) o nainštalovaní prostredia Eclipse a používaní nástroja JavaDoc v tomto prostredí.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na budúci týždeň v stredu. Čas sa upresní mailom.

Zápis zo stretnutia k tímovému projektu 4/2005

Dátum: 2.11.2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Bartalos

Vyhodnotenie úloh z predchádzajúceho stretnutia

Termín ukončenia riešitelia	Opis úlohy	Stav plnenia
Ján Porubský 2.11.2005 2.11.2005 2.11.2005	 Sprevádzkovať stránku na serveri labss2 Urobiť analýzu nástrojov na navrhovanie web formulárov (Dreamweaver a pod) Vytvoriť návrh webu portálu – technologický aj vizuálny	 Splnená 100% Splnená 100% Splnená 100%
Michal Barla 2.11.2005 2.11.2005	 Analýza problémovej oblasti a špecifikácia – 1. outline dokumentu v úložisku Zmeny v doménovej ontológii podľa záverov diskusie na treťom stretnutí	 Splnená 100% Splnená 100%
Michal Tvarožek 2.11.2005 2.11.2005	 Analýza problémovej oblasti a špecifikácia – 1. outline dokumentu v úložisku Zmeny v doménovej ontológii podľa záverov diskusie na treťom stretnutí	 Splnená 100% Splnená 100%
Peter Sivák 2.11.2005	 Analýza nástroja Orbeon + porovnanie s Cocoonom, pokračovanie v analýze Java servletov (Tomcat/JBoss) a ontologického úložiska (Sesame/Jena) – verzia v úložisku	 Splnená 50%
Kristián Szobi 2.11.2005 2.11.2005	 Napísať úvod do architektúry – 1. outline dokumentu v úložisku Spustiť na serveri systém Bugzilla a vytvoriť pre tím (alebo na nejaký nasmerovať) tutoriál o jeho používaní.	 Splnená 0% Splnená 0%
Peter Bartalos 2.11.2005	 Práca na editore ontológie	 Splnená 100%

2.11.2005	Vytvoriť malý tutoriál pre tím (alebo na nejaký nasmerovať) o nainštalovaní prostredia Eclipse a používaní nástroja JavaDoc v tomto prostredí.	Splnená 100%

Náplň stretnutia a závery z diskusie

Na začiatku stretnutia sa prejednalo plnenie jednotlivých úloh, vyplývajúcich z predchádzajúceho stretnutia.

Členovia tímu prezentovali výsledky svojej práce v uplynulom týždni. Niektoré výsledky boli aj spísané v dokumentoch:

- Peter Bartalos: „Spojazdnenie Eclipse.doc“, „Javadoc v eclipse.doc“
- Michal Barla: „Analyza_v4.doc“, „Zápis TP-2005-03.doc“
- Kristián Szobi: „Orbeon prezentačný server.pdf“
- Michal Tvarožek: „Ontology v0.5.15.zip“

Bolo dohodnuté, že *to-do* v systéme *projectpath* si bude odškrtávať moderátor sám, aby vedel sledovať, ktoré úlohy boli splnené.

Jano prezentoval návrh webu projektu aj portálu. Bol s ním spokojní každý. Po prezentácii sa diskutovalo o:

- Vytvorení textovej formy webu, aby bolo možné stránku prezerať nie len v grafickej forme
- Vytvorení verzie pre nevidiacich
- Vytvorení verzie pre pomalé sieťové pripojenie
- Optimalizovaní pre určité rozlíšenie, prispôsobovaní sa rozlíšeniu (adaptivita, prezeranie cez PDA)
- Názve portálu
- Odkaze na hachiban (názov odkazu zo stránky projektu)

Peter Sivák rozprával o *JetSpeed*, *portletoch*, ich spolupráci s *cocoonom* a o *JBoss*. Diskutovalo sa o výhodách, nevýhodách *cocoonu* a *orbeonu*. Prikláňame sa k použitiu *cocoonu*. Rozhodovalo sa, o verzii *JetSpeed*, ktorú by sme chceli použiť. Prikláňame sa k použitiu verzie 1. Padol návrh vytvorenia nástroja na vyklikanie si layoutu stránky portálu. Diskutovalo sa aj o použití *portletov*, ktoré by mohli byť použiteľné (prihlásenie sa). Roman rozprával o technológiách JAVA *beam*, EJB, *servletoch*, *JSP*.

Michal Tvarožek nám ukázal zmeny v ontológii ponuky práce, ktoré boli urobené po diskusii na minulom stretnutí (vyriešený problém usporiadania).

Michal Barla nám predstavil prvotný dokument analýzy a následne sme hovorili o jeho ďalšom vývoji. Boli urobené malé úpravy tohto dokumentu.

Zaoberali sme sa aj ontologickými a relačnými úložiskami. Hovorili sme o prípadoch ich použitia.

Dohodli sme sa na adresárovej štruktúre úložiska dokumentov, ktorá bola hneď aj vytvorená. Ukladané dokumenty budú vo formáte „doc“.

Podarilo sa skompilovať *cocoon*.

Peter Bartalos ukázal výsledok práce editora ponúk (aplikácia). Následne diskutoval o editore s Romanom.

Plán budúcich úloh

Termín ukončenia riešitelia	Opis úlohy
Ján Porubský 7.11.2005 9.11.2005 9.11.2005	 Napísať o dreamweaver, NVU, JetSpeed, portletoch do analýzy Pokračovať v technologickom návrhu webu a analýze nástrojov Zistiť možnosti využitia cocoonu a portletov, JetSpeed
Michal Barla 7.11.2005 9.11.2005 9.11.2005	 Pokračovať v písaní analýzy Pokračovanie v analýze a špecifikácii Ďalšia práca na ontológii
Michal Tvarožek 7.11.2005 9.11.2005 9.11.2005	 Pokračovať v písaní analýzy, vytvorenie úvodu analýzy a šablóny dokumentu, integrácia dokumentu Pokračovanie v analýze a špecifikácii Ďalšia práca na ontológii
Peter Sivák Predĺžené 7.11.2005 9.11.2005 9.11.2005	 Analýza nástroja Orbeon + porovnanie s Cocoonom, pokračovanie v analýze Java servletov (Tomcat/JBoss) a ontologického úložiska (Sesame/Jena) – verzia v úložisku Napísať o analyzovaných technológiách (cocoon, Jena, Sesame) Pokračovanie v práci na návrhu systému Analyzovať, testovať technológie
Kristián Szobi Predĺžené Predĺžené 7.11.2005 9.11.2005 9.11.2005 9.11.2005	 Napísať úvod do architektúry – 1. outline dokumentu v úložisku Spustiť na serveri systém Bugzilla a vytvoriť pre tím (alebo na nejaký nasmerovať) tutoriál o jeho používaní. Napísať o analyzovaných technológiách (JUnit, JBoss, Orbeon) Pokračovanie v práci na návrhu systému Analyzovať, testovať technológie Zistiť možnosti používania Bugzilly, konzultovať s Peťom Lackom, dohodnúť jej používanie
Peter Bartalos 7.11.2005 9.11.2005 9.11.2005	 Napísať o Javadoc, Eclipse Práca na editore ponúk Štúdium sesame

9.11.2005	Kozultovať s Peťom Lackom spustenie Eclipse v softvérovom štúdiu

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 9.11.2005.

Zápis zo stretnutia k tímovému projektu 5/2005

Dátum: 8.11.2005

Prítomní: Michal Barla, Peter Bartalos, Roman Filkorn, Ján Porubský, Peter Sivák, Kristián Szobi, Michal Tvarožek

Miesto: Softvérové štúdio

Zapísal: Peter Sivák

Vyhodnotenie úloh z predchádzajúceho stretnutia

Riešiteľ Termín ukončenia	Opis úlohy	Stav plnenia
Ján Porubský		
7.11.2005	• Napísať o dreamweaver, NVU, JetSpeed, portletoch do analýzy	• Splnená 100%
9.11.2005	• Pokračovať v technologickom návrhu webu a analýze nástrojov	• Splnená 100%
9.11.2005	• Zistiť možnosti využitia cocoonu a portletov, JetSpeed	• Splnená 100%
Michal Barla		
7.11.2005	• Pokračovať v písaní analýzy	• Splnená 50 %
9.11.2005	• Pokračovanie v analýze a špecifikácii	• Splnená 100%
9.11.2005	• Ďalšia práca na ontológii	• Splnená 50 %
Michal Tvarožek		
7.11.2005	• Pokračovať v písaní analýzy, vytvorenie úvodu analýzy a šablóny dokumentu, integrácia dokumentu	• Splnená 50 %
9.11.2005	• Pokračovanie v analýze a špecifikácii	• Splnená 100%
9.11.2005	• Ďalšia práca na ontológii	• Splnená 50 %
Peter Sivák		
7.11.2005	• Analýza nástroja Orbeon + porovnanie s Cocoonom, pokračovanie v analýze Java	• Splnená 100%

	servletov (Tomcat/JBoss) a ontologického úložiska (Sesame/Jena) – verzia v úložisku	
7.11.2005	• Napísať o analyzovaných technológiách (cocoon, Jena, Sesame)	• Splnená 100%
9.11.2005	• Pokračovanie v práci na návrhu systému	• Splnená 50 %
9.11.2005	• Analyzovať, testovať technológie	• Splnená 100%
Kristián Szobi		
7.11.2005	• Napísať úvod do architektúry – 1. outline dokumentu v úložisku	• Splnená 50 %
7.11.2005	• Spustiť na serveri systém Bugzilla a vytvoriť pre tím (alebo na nejaký nasmerovať) tutoriál o jeho používaní.	• Splnená 25 %
7.11.2005	• Napísať o analyzovaných technológiách (JUnit, JBoss, Orbeon)	• Splnená 100%
9.11.2005	• Pokračovanie v práci na návrhu systému	• Splnená 50 %
9.11.2005	• Analyzovať, testovať technológie	• Splnená 100%
9.11.2005	• Zistiť možnosti používania Bugzilly, konzultovať s Peťom Lackom, dohodnúť jej používanie	• Splnená 100%
Peter Bartalos		
7.11.2005	• Napísať o Javadoc, Eclipse	• Splnená 100%
9.11.2005	• Práca na editore ponúk	• Splnená 50 %
9.11.2005	• Štúdium sesame	• Splnená 100%
9.11.2005	• Kozultovať s Peťom Lackom spustenie Eclipse v softvérovom štúdiu	• Splnená 50 %

Náplň stretnutia a závery z diskusie

- Na začiatku stretnutia sa zistil stav plnenia úloh z minulého týždňa.
- Diskutovalo sa o spôsobe vytvárania UML diagramov do dokumentácie.
    - Jednak o samotnom UML, kde Michal B. povedal, že pošle dokument, ktorý má, o vytváraní diagramov UML.
    - A tiež o vhodnom nástroji na vytváranie UML diagramov. Roman F. nám odporučil *Poseidon for UML*, ktorý sme vyskúšali. Zatiaľ sme sa nerozhodli, ktorý nástroj budeme používať.
- Ján P. prezentoval zmeny, ktoré vykonal vo webovej prezentácii tímu (odkaz na systém manažmentu nášho projektu z webovej prezentácie tímu) a v dizajne portálu pracovných príležitostí (zmenšenie loga a prispôsobenie návrhu rozlíšeniu 800x600).
- Ján P. napísal text o prispôsobení webového portálu pre nevidiacich a slabozrakých, ktorý dá ostatným k dispozícii čo najskôr. Z následnej diskusie vyplynulo, že možnosti prispôsobenia portálu nevidiacim a slabozrakým môžeme v dokumentácii uviesť do časti požiadavky, a tiež do časti technologické riešenie.
- Ján P. prerobil prvý návrh portálu tak, aby používal technológiu CSS.
- Roman F. nám povedal, že v Softecu vyvíjajú svoj vlastný nástroj na integrovanie portletov.
- Michal B. nás informoval o výsledkoch stretnutia riešiteľov štátneho projektu.
    - Pôvodne sa na uvedenom stretnutí uvažovalo o tom, že každý nástroj bude na výstup generovať dáta v jazyku založenom na XML, pre ktorý sa v rámci projektu dohodne malý počet používaných elementov.
    - Keďže sa v priebehu stretnutia ukázala potreba komplexnejšej sady formátovacích značiek predbežne sa rozhodlo, že sa budú používať aj HTML značky.
- Roman F. informoval riešiteľov štátneho projektu zo Softecu, že sa chystáme v našom projekte použiť rámec Jetspeed.
- Dohodli sme sa, že si budeme zadávať merateľnejšie úlohy.
- Michal T. nás informoval, že bude spolu s Michalom B. ďalej pracovať na ontológii po 18.11.
- Dohodli sme sa, že okrem ontologického úložiska bude systém používať aj relačnú databázu.
- Michal B. pridal do dokumentácie špecifikáciu systému a analýzu riadenia verzií zdrojových kódov pomocou nástroja Subversion.
- Michal B. nás informoval, že zodpovedným za školský server vyhradený pre štátny projekt je doktorand Jaroslav Jakubík. Dohodli sme sa, že Peter S. sa ho spýta, či je možné vytvoriť kontá na prístup k Subversion a Sesame na tomto serveri aj pre štyroch členov tímu, ktorý nerobia na štátnom projekte.
- Peter B., Peter S. a Kristián S. nás informovali, že sa v pondelok stretli v softvérovom štúdiu a pracovali na návrhu systému. Výstup sa očakáva v podobe textu do dokumentácie a diagramov v stredu 9.11.

- Kristián S. nás informoval, že konzultoval s Petrom Lackom inštaláciu systému BugZilla na server *labss2*. Dohodli sa, že sa stretnú v piatok 11.11. a inštaláciu vykonajú.
- Peter B. nás informoval o pokračovaní práce na editore ponúk. V tejto súvislosti v stredu 2.11. prácu konzultoval s Ing. Grlickým. Na konzultácii sa dohodli, že v tejto časti nebude robiť dynamické generovanie formulárov. Dohodli sme sa, že chceme vidieť aj niečo bližšie o dizajne vytváraného editora (z pohľadu výzoru a ovládania). Ďalej sme sa dohodli, že využijeme skúsenosti používateľov tohto editora pre náš projekt.
- Peter B. nás informoval, že konzultoval s Petrom Lackom inštaláciu J2SE, Eclipse a Sesame na počítačoch v softvérom štúdiu. Stretne sa s ním niekedy v najbližšom týždni.
- Michal B. nás informoval o rozhovore s členmi tímu, ktorý pracujú na projekte WRAPPER. Dohodli sme sa, že im poskytne ontológiu pracovných ponúk, ktorú vytvoril spolu s Michalom T. Táto komunikácia bude prebiehať hlavne s Igorom Bertom.
- Roman F. povedal, že zistí od manažmentu štátneho projektu, či by štyria členovia nášho tímu, ktorý nepracujú na štátnom projekte, nemali podpísať prehlásenie o nešírení informácií a medzivýsledkov zo štátneho projektu tretím stranám.
- Peter B. napísal do dokumentácie text o nástroji javadoc a vývojom prostredí eclipse a najbližšie sa pokúsi načítať do Sesame vytvorenú ontológiu.
- Dohodli sme sa, že nám Roman F. pošle mailom detaily k odovzdávaniu dokumentácie k prvému kontrolnému bodu
    - dokument sa odovzdáva najneskôr v piatok o 14:00.
- Dohodli sme sa, že všetci pošlú mailom Michalovi T. informáciu, ktorú časť dokumentácie robili.
- Dohodli sme sa na rozdelení manažérskych rolí v tíme nasledovne:
    - Vedúci tímu bude Michal T.
    - Manažér vývoja bude Peter S.
    - Manažér kvality bude Kristián S.
    - Manažér plánovania bude Michal B.
    - Manažér podporných činností bude Peter B.
    - Manažér PR bude Ján P.
- Michal T. konzultoval s Romanom F. štruktúru dokumentácie.
- Na konci stretnutia sa určili úlohy na ďalší týždeň.

Plán budúcich úloh

Riešiteľ	Opis úlohy	Termín ukončenia	Priorita
Ján Porubský	• Vytvorí návrh obrazoviek (spolu s Michalom B. a Michalom T.)	16.11.2005	Stredná
	• Vytvorí navigačný model portálu (sekvenciu obrazoviek)	16.11.2005	Stredná
Michal Barla	• Dohodne sa s druhým tímom na termíne odovzdania nášho dokumentu	16.11.2005	Vysoká
	• Pošle metodiku štátneho projektu o prispôsobení nástrojov tak, aby boli prezentovateľné	16.11.2005	Nízka
	• Pošle dokument o vytváraní UML diagramov	16.11.2005	Stredná
	• Napíše do dokumentácie analýzu prezentačnej ontológie Fresnel	16.11.2005	Stredná
	• Vytvorí návrh obrazoviek (spolu s Michalom T. a Jánom P.)	16.11.2005	Stredná
Michal Tvarožek	• Integrácia dokumentácie	18.11.2005	Vysoká
	• Pridá do plánu stĺpec o predpokladanom trvaní úlohy	16.11.2005	Nízka
	• Vykoná úpravy do analýzy a špecifikácie ak budú potrebné	16.11.2005	Stredná
	• Pomôže Jánovi P. a Michalovi B. s návrhom obrazoviek.	16.11.2005	Nízka
Peter Sivák	• Dohodne s Jaroslavom Jakubíkom vytvorenie prístupových práv na server štátneho projektu	16.11.2005	Nízka
	• Vytvorí technologický návrh do dokumentácie	9.11.2005	Vysoká
	• Upraví časť *návrh* v dokumentácii podľa potreby	16.11.2005	Stredná
Kristián Szobi	• Inštalácia systému Bugzilla	16.11.2005	Nízka
	• Vytvorí technologický návrh do dokumentácie	9.11.2005	Vysoká
	• Upraví časť *návrh* v dokumentácii podľa potreby	16.11.2005	Stredná

Peter Bartalos	• Práca na editore pracovných ponúk (načítanie ontológie do Sesame)	16.11.2005	Stredná
	• Napíše do dokumentácie analýzu generovania formulárov na základe informácií v ontologickom úložisku	16.11.2005	Vysoká
	• Pridá stĺpec priorita do šablóny pre zápis zo stretnutia	16.11.2005	Vysoká
	• Bude prezentovať vytváraný editor ponúk	16.11.2005	Stredná
	• Zabezpečenie inštalácie J2SE na počítače v softvérovom štúdiu	16.11.2005	Stredná

Ďalšie stretnutie

Ďalšie stretnutie je naplánované 16.11.2005 o 7:30.

Hachiban Team 八番

Zápis zo stretnutia k tímovému projektu 6/2005

Dátum: 16.11.2005

Prítomní: Michal Barla, Peter Bartalos, Roman Filkorn, Ján Porubský, Peter Sivák, Kristián Szobi, Michal Tvarožek

Miesto: Softvérové štúdio

Zapísal: Ján Porubský

Vyhodnotenie úloh z predchádzajúceho stretnutia

Riešiteľ **Termín ukončenia**	**Opis úlohy**	**Stav plnenia**
Ján Porubský	• Vytvorí návrh obrazoviek (spolu s Michalom B. a Michalom T.)	• Splnená 75%
	• Vytvorí navigačný model portálu (sekvenciu obrazoviek)	• Splnená 75%
Michal Barla	• Dohodne sa s druhým tímom na termíne odovzdania nášho dokumentu	• Splnená 100%
	• Pošle metodiku štátneho projektu o prispôsobení nástrojov tak, aby boli prezentovateľné	• Splnená 100%
	• Pošle dokument o vytváraní UML diagramov	• Splnená 100%
	• Napíše do dokumentácie analýzu prezentačnej ontológie Fresnel	• Splnená 0%
	• Vytvorí návrh obrazoviek (spolu s Michalom T. a Jánom P.)	• Splnená 75%

Michal Tvarožek	• Integrácia dokumentácie • Pridá do plánu stĺpec o predpokladanom trvaní úlohy • Vykoná úpravy do analýzy a špecifikácie ak budú potrebné • Pomôže Jánovi P. a Michalovi B. s návrhom obrazoviek.	• Splnená 100% • Splnená 100% • Splnená 100% • Splnená 100%
Peter Sivák	• Dohodne s Jaroslavom Jakubíkom vytvorenie prístupových práv na server štátneho projektu • Vytvorí technologický návrh do dokumentácie • Upraví časť *návrh* v dokumentácii podľa potreby	• Splnená 50% • Splnená 90% • Splnená 100%
Kristián Szobi	• Inštalácia systému Bugzilla • Vytvorí technologický návrh do dokumentácie • Upraví časť *návrh* v dokumentácii podľa potreby	• Splnená 50% • Splnená 90% • Splnená 100%
Peter Bartalos	• Práca na editore pracovných ponúk (načítanie ontológie do Sesame) • Napíše do dokumentácie analýzu generovania formulárov na základe informácií v ontologickom úložisku • Pridá stĺpec priorita do šablóny pre zápis zo stretnutia • Bude prezentovať vytváraný editor ponúk • Zabezpečenie inštalácie J2SE na počítače v softvérovom štúdiu	• Splnená 75% • Splnená 50% • Splnená 100% • Splnená 100% • Splnená 50%

Náplň stretnutia a závery z diskusie

- V úvodnej časti stretnutia sa zisťoval stav plnenia úloh z minulého týždňa.
- Ján P. prezentoval návrh obrazoviek a navigačný model portálu.
  Počas prezentácie vznikla požiadavka na dotvorenie úvodnej časti so zobrazením nasledovania jednotlivých obrazoviek po sebe.

- Roman F. začal diskusiu ohľadom životopisov jednotlivých používateľov portálu – či by bolo užitočné aby používateľ zverejňoval iba časti životopisu
  Michal T. – zobrazovanie častí životopisov označil za zbytočné
  Zhodli sme sa na tejto možnosti ako na možnom nástroji do budúcnosti

- Michal B. hovoril o štandardoch životopisov vo formáte XML
- Michal B. hovoril o dohode na výmenu dokumentácie s druhým tímom – 18.11.2005, 13:00.
  Výmena dokumentácie bude aj v podobe pdf.

- Michal B. pokračoval kontrolou úloh z minulého týždňa – rozposlal dokument vytváraní UML diagramov, metodiku štátneho projektu o prispôsobení nástrojov. Analýzu prezentačnej ontológie Fresnel nedoplnil do dokumentácie, zostáva naďalej ako úloha, ale s nízkou prioritou
- Michal T. prezentoval stav dokumentácie potrebnej k termínu 18.11.2005. Zhodnotil ju na 75% hotovú. Podčiarkol vznikajúci sklz oproti plánu.
- Michal T. hovoril o zmenách, ktoré nastali vo výslednej dokumentácii a informoval nás o komentároch v dokumentácii, ku ktorým sa má každý vyjadriť.
- Michal T. upozornil na doplnenie použitej literatúry do dokumentácie ak je potrebná.
- Michal B. prezentoval prvú verziu plánu projektu na ďalšie 4 týždne. Vyzval ďalších členov, aby sa pozreli na daný plán a vyjadrili sa k jednotlivým bodom.
- Peter S. sa venoval analýze ontológie a navrhol rozdeliť analýzu na doménovo nezávislú a závislú. Michal B. – dohoda o možnosti takéhoto opisu ontológie.
- Peter S. ďalej informoval o postupe pri vybavovaní prístupu na server štátneho programu. Prístup zatiaľ nebol vybavený. Roman F. prisľúbil pomoc pri tomto probléme.
- Peter S. spolu s Kristiánom S. prezentovali zmeny, ktoré urobili v dokumentácii – notifikácia zmien v úložisku a zmeny v architektúre systému.
- Kristián S. ďalej hovoril o problémoch pri inštalácii bugzilly a problémy pri komunikácii s ľuďmi, ktorých sa inštalácia týka .
- Peter B. zdôraznil potrebu nainštalovať na počítače v učebni Javu – problémy s vybavovaním.
- Michal B. komentoval prezentáciu – hlavný cieľ nie je prispôsobiť generovanie formulárov používateľovi ale ontológii. Dohoda o prerobení analýzy generovania formulárov – Peter B.
- Peter B. predviedol editor pracovných ponúk a doterajší pokrok v editore. Ďalej prezentoval problémy vzniknuté pri tvorbe editora.

- Roman F. zdôraznil požiadavku vytvoriť rýchly editor a nestrácať čas teraz nepodstatnými vecami.
- Michal B. položil otázku Romanovi F. – komu odovzdať z pedagógov dokumentáciu, otázky ohľadom preberacieho protokolu. Roman F. – odovzdať Vlado Grlický, podpísanie protokolu o odovzdaní
- Peter B. prezentácie existujúcej verzie analýzy generovania formulárov - XFORMS. Základná prezentácia XFORMS. Peter S. mal otázku o nutnosti použitia XFORMS a výhodách oproti XHTML. Kristián S. navrhoval riešiť generovanie XFORMS na strane servera.
- Michal B. dohoda s Petrom B. o prerobení analýzy generovania formulárov.
- Diskusia o XFORMS – Roman F. hovoril o možnosti XFORMS a generovaní pomocou XFORMS aj ďalších verzií prezentácií – napríklad na PDA.
- Peter B. zdôvodnil použitie XFORMS , pretože sú platformovo nezávislé.
- Kristián S. prezentoval podporu XFORMS zo strany Cocoonu.
- Michal T. spolu s Romanom F. postupne prechádzali existujúcu verziu dokumentácie. Roman F. pripomenul na potrebu doplnenia dokumentácie o porovnanie navrhovaného portálu s nejakým podobným existujúcim (poprípade dvoma) a vypichnutie zaujímavých vlastností ako možnosť inšpirácie. Tiež prezentácia funkcií lepších voči opisovaným portálom.
- Roman F. pripomienky k dokumentácii v často architektúra systému – zmena obrázku, rozčlenenie textu.
- Diskusia o pasívnom a aktívnom konzumentovi. Roman F. odporučil nájsť pre pasívneho konzumenta iný názov. Michal T. navrhol považovať neprihlásených užívateľov vlastne za užívateľov Guest, pričom by nefungovalo prispôsobovanie tomuto užívateľovi. Ďalej požiadavka od Romana F. presunúť časť návrhu prispôsobení pre nevidiacich do príloh.
- Michal T. postupne s celým tímom prechádzal komentáre v dokumentácii. Vyhodiť z návrhu jetSpeed z dôvodu opakovania sa. Peter S. pridať podnadpisy do architektúry systému a vymeniť názov rúra (dátovod).
- Michal B. a Roman F. diskusia o robení posudkov pre druhý tím. Dohodli sme sa že úlohy na ďalší týždeň postupne vzniknú pri tvorbe posudku – čiže až po tom ako dostaneme dokumentáciu od druhého tímu.
- Peter S. otázky ohľadom prototypu, či má byť prototypovaná iba malá časť viac do hĺbky alebo prototyp zasadení do celého systému. Roman F. objasnil otázky okolo protoypu – prezentácie v dobre vyzerajúcej forme, pracovať už v prostredí daného systému. Prezentácia nejakých ukážkových portletov, vytvorenie základnej stránky.
- Určenie ďalších úloh, pričom väčšina úloh je iba do štvrtka, čiže do dokumentácie, ktorá sa odovzdáva v piatok 18.11.2005. Ďalšie úlohy budú doplnené po tom ako dostaneme dokumentáciu od druhého tímu.

Plán budúcich úloh

Riešiteľ	Opis úlohy	Termín ukončenia	Priorita
Ján Porubský	• Doplní časť navigačný model portálu v dokumentácii • Aktualizácia web stránky tímu	17.11.2005 23.11.2005	Vysoká Stredná
Michal Barla	• Napíše do dokumentácie analýzu prezentačnej ontológie Fresnel • Rozdelenie úloh pre členov tímov na vytvorení posudku • Dopracovanie use case diagramov do dokumentácie	17.11.2005 23.11.2005 17.11.2005	Nízka Stredná Vysoká
Michal Tvarožek	• Integrácia dokumentácie	17.11.2005	Vysoká
Peter Sivák	• Integrácia zmien do časti Architektúra systém-*+66+u v dokumentácii	17.11.2005	Vysoká
Kristián Szobi	• Integrácia zmien do časti Architektúra systému v dokumentácii	17.11.2005	Vysoká
Peter Bartalos	• Práca na editore pracovných ponúk (načítanie ontológie do Sesame) • Prerobí analýzu generovania formulárov (XFORMS, XHTML)	23.11.2005 23.11.2005	Stredná Stredná

Ďalšie stretnutie

Ďalšie stretnutie je naplánované 23.11.2005 o 7:30. Ešte v čase pred ním budú úlohy doplnené.

Zápis zo stretnutia k tímovému projektu 7/2005

Dátum: 23.11.2005

Prítomní: Michal Barla, Peter Bartalos, Roman Filkorn, Ján Porubský, Peter Sivák, Kristián Szobi, Michal Tvarožek

Miesto: Softvérové štúdio

Zapísal: Kristián Szobi

Vyhodnotenie úloh z predchádzajúceho stretnutia

Riešiteľ **Termín ukončenia**	**Opis úlohy**	**Stav plnenia**
Ján Porubský	• Doplní časť navigačný model portálu v dokumentácii • Aktualizácia web stránky tímu	• Splnená 100% • Splnená 100%
Michal Barla	• Napíše do dokumentácie analýzu prezentačnej ontológie Fresnel • Rozdelenie úloh pre členov tímov na vytvorení posudku • Dopracovanie use case diagramov do dokumentácie • Vytváranie plánu na posledné 4 týždne semestra	• Zrušená • Splnená 100% • Splnená 100% • Splnená 100%
Michal Tvarožek	• Integrácia dokumentácie	• Splnená 75%
Peter Sivák	• Integrácia zmien do časti Architektúra systému v dokumentácii • Vytváranie plánu na posledné 4 týždne semestra	• Splnená 100% • Splnená 90%

Kristián Szobi	• Inštalácia systému Bugzilla • Integrácia zmien do časti Architektúra systému v dokumentácii	• Splnená 50% • Splnená 100%
Peter Bartalos	• Práca na editore pracovných ponúk (načítanie ontológie do Sesame) • Prerobí analýzu generovania formulárov (XFORMS, XHTML)	• Splnená 75% • Splnená 100%

Náplň stretnutia a závery z diskusie

- V úvodnej časti stretnutia sa zisťoval stav plnenia úloh z minulého týždňa.
- Michal B. Informuje Romana F., že dokumentácia sa nachádza u p. Bielekovej. Ďalej informoval o rozdelení úloh na posudku.
- Z nasledujúcej diskusie vyplynulo, že študovanie ontológie Fresnel nemá prioritu – využije sa rámec V. Glického. Je tu však riziko, že sa zo v tomto semestri nestihne.
- Michal B. Oznamuje, že sa stretli s Michalom T. kvôli návrhu algortimu generovania formulára. Skúšali aj circle v sesame. Dospeli k názoru, že formulár bude mať veľa combo-boxov pre veľký počet možností.
- Na formulári bude ďalej pracovat Peter B.
- Diskusia o usporiadaní položiek na formulári – podľa čoho položky usporiadavať. Roman F. Navrhuje oddeliť položky logiky (aspoň v rámci namespace-ov).
- Na generovaní formulárov budú ďalej pracovať Peter B., Peter S. a Michal B.
- Ján P. nás oboznámil, že v úložisku je jeho posudok k návrhu a špecifikácii a že urobil aj update stránky (pridal linky, dokumenty, plán)
- Peter B. informoval o naištalovanej Jave na niektorých počítačoch v softvérovom štúdiu.
- Michal B.sa pýta Romana F. ako je to s kontami do suversion. Zistilo sa, že vzniklo nedorozumenie – kontá sa v najbližšej vybavia.
- Kristián S. bude spolupracovať s Jánom P. na overení architektúry.
- Nastala diskusia o práci tímu WRAPPER.
- Michal B. nás oboznamuje s plánom na prototyp. Treba urobiť formulár, ktorý dokáže zapísať niečo do sesame a potom to načítať (za pomoci portletov, cocoona).
- Michal T. navrhol myšlienku použitia OWL Full – asi by to neriešilo problém usporiadania. Treba sa spýtať V. Grlického, či Fresnel usporiadava to, čo sa má zobraziť.

Plán budúcich úloh

Riešiteľ	Opis úlohy	Termín ukončenia	Priorita
Ján Porubský	• Naštuduje si spôsob, ako sa do Jetspeedu vytvarajú šablóny a prevedie grafický návrh portálu do Jetspeedu	30.11.2005	Vysoká
Michal Barla	• Práca na algoritme generovania formulárov - vytvorenie prvej verzie, predpokladá sa spolupráca s Peťom B., Peťom D. a Mišom T	30.11.2005	Vysoká
Michal Tvarožek	• najneskôr do piatku dokončiť posudok • zintegruje prvé príspevky do dokumentácie pre O4	19.11.2005 30.11.2005	Vysoká Nízka
Peter Sivák	• Práca na algoritme generovania formulárov - vytvorenie prvej verzie, predpokladá sa spolupráca s Peťom B. a Mišom T	30.11.2005	Vysoká
Kristián Szobi	• Prepojí JetSpeed, Cocoon a Sesame - ľubovolná jednoduchá funkcionalita (možno až na zahodenie) - overí sa architektúra, získa sa know-how, ako sa pracuje s konkrétnymi nástrojmi	30.11.2005	Vysoká
Peter Bartalos	• bude spolupracovat s Misom B. a Petom S. na návrhu a implementácii algoritmu pre generovanie formularov • bude primárne zodpovedný za ulohu 3 - CRUD. Preberie spolu s Peťom S. čo už je hotové, s Kristiánom ako je to hotové • navrhe abstraktný prístup k ontologickému úložisku cez triedu, ktorá implementuje zadefinovane rozhranie.	30.11.2005 30.11.2005 30.11.2005	Vysoká Vysoká Vysoká

Ďalšie stretnutie

Ďalšie stretnutie je naplánované 30.11.2005 o 7:30.

Zápis zo stretnutia k tímovému projektu 8/2005

Dátum: 30. 11. 2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Tvarožek

Náplň stretnutia a závery z diskusie

- V úvode stretnutia sme vyhodnotili stav riešenia jednotlivých úloh.
  - Prediskutovali sme využitie ontológie Fresnel pri prezentácii ontológie a generovaní formulárov (je to skôr na inštancie, pričom definuje poradie prvkov).
  - Pri inštalácii a sfunkčnení Jetspeedu sa vyskytli problémy, pretože sa nepodarilo vytvoriť vlastný portál. Je potrebné tomu ešte venovať viac úsilia a preštudovať dokumentáciu.
  - Posudok bol dokončený a odovzdaný.
  - Začala práca na tvorbe dokumentácie k prototypu.
  - Na algoritme generovania formulárov sa pracuje, v krátkej dobe bude vytvorený prvý prototyp generátora formulárov pre triedu Benefit z doménovej ontológie.
- Pedagogický vedúci sa vyjadril k dokumentácii k analýze, špecifikácii a hrubému návrhu riešenia:
  - Dokumentácia bola výborne vypracovaná podľa požiadaviek vedúceho. Externému pozorovateľovi mohol chýbať bližší opis niektorých aspektov riešenia.
  - V dokumentácii mierne chýbal hrubý návrh, ktorý by však bude v dokumentácii k prototypu.
  - Architektúru systému treba viac a konkrétnejšie prepracovať.
  - Viac by bolo dobré spomenúť doménovú ontológiu.
  - Treba viac zvýrazniť rozdiely oproti NAZOU.
  - Niektoré zápisy boli príliš podrobné.
  - Treba venovať viac času kontrole kvality dokumentácie (gramatika, preklepy).
- Diskusia otvorených problémov a ďalšieho postupu riešenia:
  - Ako sfunkčniť Jetspeed – doštudovať existujúcu dokumentáciu a skúšať tutorial.
  - Riešenie problému pri generovaní formulárov, keď vlastnosť je definovaná pomocou unionOf.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 7. 12. 2005.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
25. 11. 2005	Tvarožek (samostatne)	Integrácia dokumentácie a dokončenie posudku.	Vysoká	Ukončená
25. 11. 2005	Barla (Sivák)	Vytváranie plánu na posledné 4 týždne semestra.	Stredná	Ukončená
16. 11. 2005	Szobi (samostatne)	Inštalácia systému Bugzilla.	Nízka	Splnená na 50%
14. 12. 2005	Bartalos (samostatne)	Práca na editore pracovných ponúk, práca s ponukami v ontologickom úložisku – vzor CRUD (Create, Retrieve, Update, Delete)	Stredná	Splnená na 90%
30. 11. 2005	Porubský (samostatne)	Konfigurácia Jetspeedu a vytvorenie jednoduchej šablóny pre stránku portálu. Výstup: ukážková stránka portálu, prípadne už aj s nejakou funkcionalitou	Vysoká	Splnená na 25%
30. 11. 2005	Sivák (Barla, Bartalos, Tvarožek)	Návrh a implementácia časti algoritmu generovania formulárov (prvá verzia). Výstup: ukážková aplikácia na vytvorenie jednoduchého formulára	Vysoká	Splnená na 25%
30. 11. 2005	Tvarožek (samostatne)	Tvorba a integrácia dokumentácie pre O4.	Nízka	Ukončená
16. 12. 2005	Szobi (samostatne)	Prepojenie JetSpeed, Cocoon a Sesame. Návrh a implementácia dátovodu pre vytvorenie a/alebo zobrazenie ponuky v Cocoone. Výstup: ukážková aplikácia na zobrazenie jednoduchého formulára cez Cocoon	Vysoká	Splnená na 25%
30. 11. 2005	Bartalos (samostatne)	Návrh abstraktného prístupu k ontologickému úložisku cez triedu, implementujúcu definované rozhranie.	Vysoká	Splnená na 50%

Plán budúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
7. 12. 2005	Porubský (samostatne)	Aktualizácia webu (zápisy, IE, fonty). Výstup: aktuálny web	Vysoká	Nová
7. 12. 2005	Tvarožek (samostatne)	Aktualizácia šablóny na zápis zo stretnutia. Výstup: šablóna (dot)	Stredná	Nová
2. 12. 2005	Tvarožek (všetci)	Tvorba príspevkov do dokumentácie a krátkej správy o postupe riešenia. Výstup: cca. 1 strana správy od každého	Stredná	Nová
19. 12. 2005	Tvarožek (samostatne)	Tvorba a integrácia dokumentácie k prototypu riešenia. Výstup: dokumentácia k riešeniu	Stredná	Nová

Zápis zo stretnutia k tímovému projektu 9/2005

Dátum: 7. 12. 2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Barla

Náplň stretnutia a závery z diskusie

- V úvode stretnutia sme vyhodnotili stav riešenia jednotlivých úloh.
  - Úlohu zaslať do 2.12. krátku správu o riešení a problémoch nesplnili všetci – náprava čo najrýchlejšie.
  - Systém Bugzilla je nainštalovaný a pripravený na použitie.
  - Bol vytvorený projekt – portál na báze Jetspeedu.
    - Používajú sa šablóny vo Velocity.
    - Roman nás varoval, že Velocity je málo expresívne, navrhol JSP.
    - Zhodli sme sa, že funkcionalitu vykonáva primárne Cocoon a preto Velocity nie je rizikom.
  - Bol vytvorený Cocoon portlet, ktorý prepája Jetspeed s Cocoonom pomocou HTTP request a response.
    - Otvorená je otázka Control Flow.
  - Bola vytvorená nová šablóna pre zápisnice zo stretnutí.
- Pedagogický vedúci sa vyjadril k posudku nášho tímu ako aj k posudku na náš projekt:
  - Nami vypracovaný posudok bol v poriadku, bez vážnejších výhrad.
  - Posudok druhého tímu je v určitých častiach v protiklade s ich správou o riešení projektu. Celkovo nie je zlý.
    - v dok. k druhému kontrolnému bodu bude naše vyjadrenie k posudku.
- Diskusia otvorených problémov a ďalšieho postupu riešenia:
  - Technológiu EJB zatiaľ vynechávame – ide zatiaľ nadmieru požadovaného riešenia.
  - V prototype by mal byť jednoduchý formulár s obojsmernou komunikáciou – overenie návrhu architektúry.
    - Bol diskutovaný podrobnejší návrh riešenia vychádzajúci z architektúry.
    - Takýto návrh sa očakáva v dok. k druhému kontrolnému bodu (jej rozsah cca 10 strán).
  - Návrh na rýchle prototypovanie v .NET kvôli prezentovateľnosti sme zamietli, pretože nám neozrejmí riziká.
  - Úloha o vzore CRUD nad ontologickým úložiskom je splnená v rámci JOE
    - je potrebné oddeliť funkcionalitu od grafického rozhrania Swing.
    - Funkcie by mali pracovať s Java Beans triedami.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 14. 12. 2005.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
16. 11. 2005	Szobi (samostatne)	Inštalácia systému Bugzilla.	Nízka	Ukončená
14. 12. 2005	Bartalos (samostatne)	Práca na editore pracovných ponúk, práca s ponukami v ontologickom úložisku – vzor CRUD (Create, Retrieve, Update, Delete)	Stredná	Splnená na 90%
30. 11. 2005	Porubský (samostatne)	Konfigurácia Jetspeedu a vytvorenie jednoduchej šablóny pre stránku portálu. Výstup: ukážková stránka portálu, prípadne už aj s nejakou funkcionalitou	Vysoká	Splnená na 75%
30. 11. 2005	Sivák (Barla, Bartalos, Tvarožek)	Návrh a implementácia časti algoritmu generovania formulárov (prvá verzia). Výstup: ukážková aplikácia na vytvorenie jednoduchého formulára	Vysoká	Splnená na 50%
16. 12. 2005	Szobi (samostatne)	Prepojenie JetSpeed, Cocoon a Sesame. Návrh a implementácia dátovodu pre vytvorenie a/alebo zobrazenie ponuky v Cocoone. Výstup: ukážková aplikácia na zobrazenie jednoduchého formulára cez Cocoon	Vysoká	Splnená na 75%
30. 11. 2005	Bartalos (samostatne)	Návrh abstraktného prístupu k ontologickému úložisku cez triedu, implementujúcu definované rozhranie.	Vysoká	Splnená na 90%
7. 12. 2005	Porubský (samostatne)	Aktualizácia webu (zápisy, IE, fonty). Výstup: aktuálny web	Vysoká	Ukončená
7. 12. 2005	Tvarožek (samostatne)	Aktualizácia šablóny na zápis zo stretnutia. Výstup: šablóna (dot)	Stredná	Ukončená
2. 12. 2005	Tvarožek (všetci)	Tvorba príspevkov do dokumentácie a krátkej správy o postupe riešenia. Výstup: cca. 1 strana správy od každého	Stredná	Splnená na 50%
19. 12. 2005	Tvarožek (samostatne)	Tvorba a integrácia dokumentácie k prototypu riešenia. Výstup: dokumentácia k riešeniu	Stredná	Nezačatá

Plán budúcich úloh

Žiadne nové úlohy na stretnutí zadané neboli, pokračujeme v ukončovaní úloh z minulých stretnutí.

Zápis zo stretnutia k tímovému projektu 10/2005

Dátum: 14. 12. 2005

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Bartalos

Náplň stretnutia a závery z diskusie

- V úvode stretnutia sme vyhodnotili stav riešenia jednotlivých úloh.
  - Jano ukázal stav stránky portálu. Diskutovalo sa o problémoch s funkcionalitou, ktorú má stránka poskytovať (bez dostatočných používateľských práv sú problémy so zatváraním, minimalizáciou okien a pod.).
  - Peto Bartalos prezentoval Java Bean pre pracovnú ponuku. Predviedol jej ukladanie na úložisko. Diskutovalo sa o potrebe funkcií v triede Offer, na získavanie a nastavovanie atribútov podtried triedy Offer (napríklad atribúty triedy Organization). S týmto súvisí aj problém samotných formulárov. Či má existovať jeden veľký (s možnosťou minimalizácie niektorých častí), alebo sú potrebné subformuláre. Prikláňame sa k prvému spôsobu.
- Uvažujeme o použití copletov namiesto portletov. Dôvodom je, že coplety medzi sebou dokážu lahko komunikovať. O copletoch existuje krátky úvod v dokumente od Peťa Siváka.
- Navrhovalo sa nepoužívať JetSpeed z dôvodov neposkytovanej funkcionality pre užívateľov s nižšími právami a kôli tomu, že s cocoonom dokáže komunikovať len cez HTTP volanie. Problém ostal otvorený.
- Hovorilo sa o potrebe dynamického generovania Java Bean pre ponuku (práca na ňom sa presúva na ďalší semester).
- V piatok (9.12.2005) bolo neformálne stretnutie niektorých členov tímu (Barla, Filkorn, Szobi, Tvarožek). Bola vytvorená jednoduchá aplikácia, ktorá využíva formulár pomocou Cocoon Forms a je previazaná na Java Bean triedu. Zavrhli sa XForms. Na základe tejto práce sa následne na stretnutí podarilo previesť celý proces naplnenie formulára, a následné uloženie dát na úložisko. Pospájali sa dokopy cocoon, Java Bean pre ponuku, a jej uloženie. Dohodlo sa, ktoré polia ponuky budú v tejto fáze napĺňané a hned bol vytvorený model potrebný v cocoon.
- Diskutovalo sa o probléme generovanie ID pre ponuku, organizáciu a iné. Problém bude riešený pri ukladaní ponuky na úložisko.
- Nakoniec sa hovorilo o dokumentácii a prezentácii prototypu. Dohodli sme sa, že budeme vyzdvihovať snahu overiť si technológie a odôvodníme zavrhnutie tých, ktoré sme sa rozhodli nepoužiť a nahradiť inými. Dohodlo sa stretnutie na piatok (16.12.2005), sobota je určená na písanie dokumentácie, v nedeľu sa podľa potreby organizuje ďalšie stretnutie.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 16. 12. 2005.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 12. 2005	Bartalos (samostatne)	Práca na editore pracovných ponúk, práca s ponukami v ontologickom úložisku – vzor CRUD (Create, Retrieve, Update, Delete)	Stredná	Ukončená
18. 11. 2005	Porubský (samostatne)	Konfigurácia Jetspeedu a vytvorenie jednoduchej šablóny pre stránku portálu. Výstup: ukážková stránka portálu, prípadne už aj s nejakou funkcionalitou	Vysoká	Splnená na 90%
18. 11. 2005	Sivák (Barla, Bartalos, Tvarožek)	Návrh a implementácia časti algoritmu generovania formulárov (prvá verzia). Výstup: ukážková aplikácia na vytvorenie jednoduchého formulára	Vysoká	Splnená na 50%
18. 12. 2005	Szobi (samostatne)	Prepojenie JetSpeed, Cocoon a Sesame. Návrh a implementácia dátovodu pre vytvorenie a/alebo zobrazenie ponuky v Cocoone. Výstup: ukážková aplikácia na zobrazenie jednoduchého formulára cez Cocoon	Vysoká	Splnená na 90%
2. 12. 2005	Tvarožek (všetci)	Tvorba príspevkov do dokumentácie a krátkej správy o postupe riešenia. Výstup: cca. 1 strana správy od každého	Vysoká	Splnená na 50%
19. 12. 2005	Tvarožek (samostatne)	Tvorba a integrácia dokumentácie k prototypu riešenia. Výstup: dokumentácia k riešeniu	Vysoká	Nezačatá

Plán budúcich úloh

Žiadne nové úlohy na stretnutí zadané neboli, pokračujeme v ukončovaní úloh z minulých stretnutí.

Zápis zo stretnutia k tímovému projektu 1/2006

Dátum: 21.2.2006

Prítomní: Barla, Bartalos, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Sivák

Náplň stretnutia a závery z diskusie

- Na začiatku stretnutia Roman zhodnotil projekt za zimný semester. Hodnotenie tímu bolo 7.
- Roman ďalej zhodnotil dokumentáciu k druhému kontrolnému bodu. Niektoré z pripomienok:
  - nekonzistentnosť diagramu tried – napr. niektoré metódy nemali uvedený návratový typ,
  - treba uviesť súbory, ktoré sme menili pri konfigurácii jednotlivých nástrojov,
  - treba oddeliť časť ontológie, ktorá opisuje ukladané informácie od časti, ktorá opisuje ich štruktúru (A-Box, T-Box)
  - generovanie formulára: treba hlbšie rozpracovať proces transformácie dát z ontologickej reprezentácie pracovnej ponuky na vnútornú reprezentáciu (súbory model, template, bind a zdrojové súbory) – napr. uvažovať poradie ovládacích prvkov.
- Peťo B. v rámci Štátneho projektu zadefinoval nástroj, ktorý by mohol využiť generátor formulárov, ktorý vyvíjame. Jeho cieľom je preskúmať iné ako formulárové rozhrania pre prácu s pracovnými ponukami.
- Dohodli sme sa, že z prípadov použitia, ktoré sme navrli, nebudeme v ďalšom priebehu projektu uvažovať tieto: 5, 6, 7, 8, 10, 11, a zrejme ani 9 a 14. Primárne sa sústredíme na 1, 2, 3, 4, 12, 13.
- Výsledný produkt sa odovzdáva približne v 9. alebo v 10. týždni. Výsledkom bude portál, technická a používateľská dokumentácia.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 28. 2. 2006.

Plán budúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
5. 3. 2006	všetci	Napísať článok do IIT-SRC	Stredná	Nová
28.2.2006	Porubský (samostatne)	Spísať výhody a nevýhody Cocoon Portal Engine a Portal Framework	Vysoká	Nová
28.2.2006	Bartalos (samostatne)	Upraviť zdrojový kód pre prístup k Sesame, tak aby bol prehľadnejší	Stredná	Nová

Zápis zo stretnutia k tímovému projektu 2/2005

Dátum: 28. 2. 2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Szobi

Náplň stretnutia a závery z diskusie

- Diskusia o dôležitosti revízií. Dohodli sme sa, že pri revízii si treba pripraviť prezentáciu kódu na 10 minút.
- Je potreba prerobiť časť nástroja JOE tak, aby sa dal využiť pri generovaní formulárov – cez Java Bean.
- Peter B. urobil prezentáciu kódu pre prácu so Sesame
  - Bude potrebné zovšeobecniť RepositoryAccess (metódy GraphToBean, BeanToGraph)
  - Potreba podpory kaskádneho mazania v ontológii
  - Potreba zavedenia rolí používateľov
  - Celá ponuka bude reprezentovaná jednou veľkou Java Beanou
  - Validácia na úrovni formulárov (na začiatku sú len niektorý fieldy povinné, ale keď sa začne vypĺňať nepovinný field ako napr. adresa, potom sú tam jednotlivé položky označené ako requiered)
  - Nerobiť validáciu pri zápise do Sesame
- Diskusia o formulároch a právomociach užívateľov
  - Vo formulári pri registrácii sa musí zadať aj pozícia vrámci orgranizácie
  - Len administrátor pre organizáciu môže pridávať ponuky
  - OfferedBy bude dostupná len pri agentúrach
  - Rôzne režimy pre používateľov na spravovanie ponúk (podľa právomocí)
  - Vyskytla sa otázka, či je dobré predgenerovať formuláre pre všetky typy používateľov alebo sa to urobi v runtime v nejakej template
- Ján P. nás oboznámil so závermi o použití copletov
  - Portal Engine je jednoduchší ako Jetspeed
  - Prerobenie portálu z Jetpseedu do Copletov bolo úspešné
  - Prihlasovanie je implementované
  - Nefunguje admin konto (user management)
  - Dostatok dokumentácie k Portal Enginu
  - Komunikácia medzi copletmi je možná pomocou eventov
- Romanovi sme hovorili o stretnutí, ktoré sa konalo minulý týždeň

- Identifikovali sme potrebné formuláre
- Vhodné reprezentácie objektových vlastností (Tabs vs Popups)
- Romanovi sa tabky páčia
- Na multiple vlastnosti sa použije repeater
- Nápad o rozbalovaní formulára stlačením tlačítok, avšak nastáva problém s veľkým HTML formulárom
- Zistili sme, že Cocoon používa interne AJAX a JavaScript
- Bude potrebné otestovať ako Cocoon používa JavaScirpt (client vs server)
- Ďalej bude potrebné otestovať, či je možné vytvoriť skladaný formulár a tabkový formulár
- Roman nám povedal, že AccessPortal nemá prioritu, ale keď sa stihne, nebude to na škodu

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 7. 3. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
5. 3. 2006	Všetci (samostatne)	Napísať článok do IIT-SRC	Stredná	25%
28. 2. 2006	Porubský (samostatne)	Spísať výhody a nevýhody Cocoon Portal Engine a Portal Framework	Vysoká	50%

Plán budúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
5. 3. 2006	Všetci	Napísať článok do IIT-SRC	Vysoká	25%

Zápis zo stretnutia k tímovému projektu 3/2006

Dátum: 7. 3. 2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: Softvérové štúdio

Zapísal: Porubský

Náplň stretnutia a závery z diskusie

- Ján P. prezentoval výsledky týkajúce sa štúdia portálových riešení v cocoone
  - Ďalší postup sa týka už iba cocoon enginu, pre ktorý sme sa rozhodli
  - Pokračujúce problémy s nástrojmi na správu užívateľov v portálovom frameworku
  - Diskusia o možnom riešení : Michal B. možný smer riešenia prostredníctvom diskusných fór o danej problematike
  - Úroveň konfigurácie projektu pomocou portal engine je v stave ako bol jetspeed
- Peter B. nás obozámil s výsledkami práce na metóde BeanToGraph
  - Potreba metódy, ktorá dostane Beanu a vráti graf
  - Prezentácia potrebných vstupov pre danú metódu
  - Micha T. návrh aby sa spravil reálny príklad na generovanie grafu z Beany
- Roman F. poukázal na nutnosť spraviť nejakú reálnu časť projektu
  - Snaha zsíkať pohľad zhora na celý projekt a nezabiehať veľmi do podrobností jednotlivých častí
- Michal T. sa zaoberal otázkou rozdelenia úloh z minulého stretnutia
  - Dohoda, že presnejšia špecifikácia a rozdelenie sa rozošlú emailom
- Dohoda celého tímu na dokončovaní zápisov zo stretnutia do piatka toho týždňa, kedy sa stretnutie konalo
  - Úloha pre Jána P., ktorý má na to dohliadnuť
- Stretnutie pokračovalo diskusiou
  - Michal B. a Roman F. sa zaoberali názorným rozdelením jednotlivých tabiek pre lepšiu predstavu o problematike
  - Michal T. prešiel k rozdeleniu tímu na dve časti, časť generovania formulárov a časť zaoberjúcu sa portálom
  - Generovanie formulárov – Michal B., Peter B., Peter S.
  - Portál – Michal T., Kristián S., Ján P.
  - Rozdelenie nie je trvalé a počas práce môže dochádzať k zmenám podľa potreby
- Ďalej nasledovala komunikácia a zhrnutie jednotlivých úloh a cieľov v rámci jednotlivých skupín
  - Skupiny sa dohodli na najbližších cieľoch

- Určili sa kritické body práce

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 14. 3. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
5. 3. 2006	Všetci	Napísať článok do IIT-SRC	Vysoká	100%

Plán budúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14.3.2006	Peter B.	Zadefinovať formát XML súboru pre metadáta, potrebné pre BeanToGraph. Sparsovať tieto metadáta (načítať do pamäte (objektu), tak, aby sa neskôr čítali len z nej) Pozmeniť BeanToGraph, aby vedel ukladať novú a upraviť existujúcu ponuku v DB	Vysoká	25%
14.3.2006	Michal B.	Riešiť problém s updatovaním viacerých bean naraz. Tabky.	Vysoká	30%
14.3.2006	Peter S.	Spracovať formulár pracovnej ponuku (XMLka). Prepojiť s metadátovým algoritmom na ukladanie časti ponuky PB.	Vysoká	25%
14.3.2006	Ján P.	Dokončiť konfiguráciu copletov. Grafický návrh copletov a obrazoviek. Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	25%
14.3.2006	Kristián S.	Definovať funkcionalitu jednotlivých obrazoviek. Zistiť možnosti rolí v cocoone.	Vysoká	25%
14.3.2006	Michal T.	Určiť navigačnú štruktúru portálu (logickú). Identifikovať a opísať role z pohľadu systému a z pohľadu používateľov.	Vysoká	25%

Zápis zo stretnutia k tímovému projektu 4/2006

Dátum: 14. 3. 2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Tvarožek

Náplň stretnutia a závery z diskusie

- Vyhodnotenie úloh a postupu projektu
  - Bean2Graph postupuje dobre, vyskytli sa len malé problémy.
  - Treba ešte dorobiť samotný algoritmus na zapísanie beany a metadáta.
- Skladanie bean do jednej veľkej nebude potrebné, ak sa použijú
- Zobrazenie formulárov dalo poriadne zabrať, lebo boli problémy s Cocoonom.
  - Tooltips – nezobrazujú sa správne; je to skôr bonus
  - Internacionalizácia je veľmi dôležitá a treba jej venovať zvýšenú pozornosť.
  - Treba preskúmať a rozbehať Tree widget na reprezentáciu hierarchií
- Na prehľady a zobrazovanie ponúk by bolo možné použiť fazetový prehliadač.
  - Najprv je treba osamostatniť fazetový prehliadač od rámca Softec.
  - Upraviť fazetový prehliadač tak, aby bol použiteľný v našom portály.
- Podarilo sa vytvoriť portál v Cocoone
  - Existujú už základné šablóny a grafický návrh
  - Nastavenie adresárov je treba spraviť v cocoon.xconf
  - Ďalšia práca sa zameria aj na tvorbu/implementáciu obrazoviek podľa návrhu funkcionality a navigačného modelu.
- Cocoon obsahuje autentifikačný framework, ktorý podporuje role.
  - Možnosť vytvárať vlastné role
  - Možnosť nastavovať privilégiá na jednotlivé stránky/rúry a definovať alternatívne rúry.
  - Treba ešte overiť možnosť dedenia rolí.
- Bol vytvorený navigačný model portálu
  - Verejná časť, privátna časť, systémová časť
  - Definícia rolí a ich prístupu k jednotlivým častiam
  - Je potrebné nadviazať na navigačný model a vypracovať štruktúru adresárov a uloženia jednotlivých súborov a vytvoriť príslušné sitemapy.
- Je potrebné premyslieť ako pracovať s organizáciou vo formulároch
  - Je porebné ju odniekadiaľ načítať, ale vo formulári sa nemusí zobrazovať.

- offeredBy, offeredVia, čo kde ako zobrazovať?

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 21. 3. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 3. 2006	Bartalos (samostatne)	Zadefinovať formát XML súboru pre metadíta potrebné pre Bean2Graph a vytvoriť parser na tieto metadáta.	Vysoká	Splnená na 90%
14. 3. 2006	Bartalos (samostatne)	Umožniť uloženie a úpravu ponúk pomocou Bean2Graph (algoritmus, naplnenie XML metadátami).	Vysoká	Splnená na 50%
14. 3. 2006	Barla (samostatne)	Riešiť problém s aktualizovaním viacerých bean súčasne.	Stredná	Zrušená
14. 3. 2006	Barla (samostatne)	Overiť fungovanie tabiek v Cocoone.	Vysoká	Ukončená
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponukt.	Stredná	Splnená na 25%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 50%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigáurácia).	Vysoká	Splnená na 50%
14. 3. 2006	Porubský (samostatne)	Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	Splnená na 75%
14. 3. 2006	Szobi (Tvarožek)	Definovať funkcionalitu jednotlivých obrazoviek (aj offeredBy/Via).	Vysoká	Splnená na 90%
14. 3. 2006	Szobi (samostatne)	Preskúmať možnosti použitia rolí v Cocoone na reprezentáciu rolí v našom systéme.	Stredná	Splnená na 90%
14. 3. 2006	Tvarožek (samostatne)	Vytvoriť navigačnú štruktúru portálu (logickú).	Vysoká	Ukončená
14. 3. 2006	Tvarožek (samostatne)	Identifikovať a opísať role z pohľadu systému a z pohľadu používateľov.	Stredná	Ukončená

Nové úlohy

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
21. 3. 2006	Szobi (samostatne)	Kontrola kvality formulárov a zdrojových kódov pristupujúcich k ontologickému úložisku.	Vysoká	Nová

21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Nová
21. 3. 2006	Szobi (samostatne)	Preložiť portál do angličtiny, výstup priamo do SVN.	Stredná	Nová
21. 3. 2006	Porubský (samostatne)	Problém s nefungujúcim znefunkčnením textboxu pre dátum, ak je zašktrnutý ASAP.	Vysoká	Nová
21. 3. 2006	Tvarožek (Barla?)	Osamostatniť fazetový prehliadač od Softec rámca, aby sa dal použiť v našom portály.	Vysoká	Nová
21. 3. 2006	Tvarožek (Barla?)	Pridať do portálu prehľad ponúk pomocou fazetového prehliadača len s dátami z ontológického úložiska.	Stredná	Nová
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Nová
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Nová
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Nová
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Nová
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Nová
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nová
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nová

Zápis zo stretnutia k tímovému projektu 5/2006

Dátum: 21.3.2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Barla

Náplň stretnutia a závery z diskusie

- Vyhodnotenie úloh a postupu projektu:
- Peter B.:
  - Problém s prístupom k private premenným beany v BeanToGraph
    - Vyriešené – pristupovať sa bude cez getters&setters, pred názov premennej sa doplní prefix
  - BeanToGraph - nie je vymazávanie, dokumentácia
    - Predpokladá sa budúci týždeň
- Ján P.:
  - Nedarilo sa mu plnohodnotne rozchodiť portál
    - Problém bol v zlej konfigurácii *WEB-INF/cocoon.xconf*, kde bola zle nastavená cesta k portal skinom
  - Problém s nesprávnym zobrazením formuláru v niektorých prehliadačoch vyriešil zjednodušením svojich štýlov (aktuálne sa aplikujú iba na textové polia)
- Kristián S.:
  - Problém zobrazovania *OfferedBy* a *OfferedVia*
    - *OfferedVia* sa zobrazovať nemusí nikdy
    - *OfferedBy* sa musí zobraziť, ak ponuku zadáva pracovná agentúra
    - Riešenie pomocou dodatočných metadát, template formulára s podmienkami *(jx:if)* a dodatočná premenná pre JXTemplate generátor – zadaná flowom
  - Problém rolí a správy používateľov
    - Dedenie rolí nejde ➔ v prípade potreby použijeme ich kombinácie
    - Cocoon poskytuje len interface, samotnú správu používateľov si musíme naimplementovať sami (pipelines *addUser, removeUser* a čo je za tým)
  - Skúmal autentizačný framework
    - Správa je v prílohe
  - Manažoval kvalitu
    - Zatiaľ v poriadku
- Michal T.

- Práca na integrácii nástroja Factic do tímového projektu
  - Factic implementuje rozhrania cocoon generatora
  - Funguje v coplete v portáli - zobrazenie nie je dokonalé (layout), treba ho zlepšiť
  - Pridať prácu s ďalším zdrojom dát – relDB

❖ Michal B.

- V tíme sa používali dva rôzne druhy ontologického úložiska (RDF, RDFSchema)
  - Prerobenie časti nástroja Factic, ktorá komunikuje so Sesame na RDFSchema úložisko
- Práca na generovaní formulárov
  - Využitie *RepositoryProxy* z nástroja Factic, doplnenie metód na zisťovanie ďalších vlastností o triedach z ontológie cez *OntologyExaminer*
  - Prvotné generovanie vnútornej reprezentácie formulára a beany pomocou XML DOM – bolo by vhodné pridať ešte jednu objektovú medzivrstvu reprezentujúcu widgety

❖ Peter S.

- Práca na generovaní formulárov
  - Zadefinovanie štruktúry vnútornej reprezentácie formulára a beany v XML
  - Rozchodenie Xalan XSL transformátora – umožňuje viacero súborov
  - Generovanie beany

❖ Ďalšia diskusia:

- Opäť sa preberala internacionalizácia
  - Produkt bude v angličtine, pripravený na slovenčinu – treba vytvoriť anglický slovník pojmov a v portáli sa odkazovať pomocou ID do slovníka
  - Otázna je internacionalizácia obrázkov – treba zistiť čo umožňuje I18n transformer
- Spôsob zobrazovania *Prerequisities*, riešenie multiple range niektorých hrán
  - V modeli musia byť prítomné všetky polia ➔ v beane musia byť všetky polia, niektoré budú prázdne
  - Template s podmienkami – podobne ako checkbox pre *startDateASAP*
  - Tree widget získava vyššiu prioritu – treba si to reálne vyskúšať

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 28.3.2006.

Vyhodnotenie a stav predchádzajúcich úloh

14. 3. 2006	Bartalos (samostatne)	Zadefinovať formát XML súboru pre metadíta potrebné pre Bean2Graph a vytvoriť parser na tieto metadáta.	Vysoká	Splnená na 90%
14. 3. 2006	Bartalos (samostatne)	Umožniť uloženie a úpravu ponúk pomocou Bean2Graph (algoritmus, naplnenie XML metadátami).	Vysoká	Splnená na 90%
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponukt.	Stredná	Splnená na 25%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigáurácia).	Vysoká	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	Splnená na 75%
14. 3. 2006	Szobi (Tvarožek)	Definovať funkcionalitu jednotlivých obrazoviek (aj offeredBy/Via).	Vysoká	Splnená na 90%
14. 3. 2006	Szobi (samostatne)	Preskúmať možnosti použitia rolí v Cocoone na reprezentáciu rolí v našom systéme.	Stredná	Ukončená
21. 3. 2006	Szobi (samostatne)	Kontrola kvality formulárov a zdrojových kódov pristupujúcich k ontologickému úložisku.	Vysoká	Ukončená
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Splnená na 75%
21. 3. 2006	Szobi (samostatne)	Preložiť portál do angličtiny, výstup priamo do SVN.	Stredná	Splnená na 25%
21. 3. 2006	Porubský (samostatne)	Problém s nefungujúcim znefunkčnením textboxu pre dátum, ak je zaškrtnutý ASAP.	Vysoká	Ukončená
21. 3. 2006	Tvarožek (Barla?)	Osamostatniť fazetový prehliadač od Softec rámca, aby sa dal použiť v našom portály.	Vysoká	Ukončená
21. 3. 2006	Tvarožek (Barla?)	Pridať do portálu prehľad ponúk pomocou fazetového prehliadača len s dátami z ontológického úložiska.	Stredná	Ukončená
21. 3. 2006	Sivák	Generovanie beany a mapovacích	Vysoká	Splnená na 50%

	(samostatne)	metadát pre Bean2Graph.		
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Nezačatá
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 50%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá

Nové úlohy

- Úsilie sa sústreďuje hlavne na dokončenie úloh zadefinovaných na minulých stretnutiach

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
	Barla		Nízka	Nová
	Bartalos		Nízka	Nová
	Porubský		Nízka	Nová
	Sivák		Nízka	Nová
	Szobi		Nízka	Nová
	Tvarožek		Nízka	Nová

Príloha – Správa k autentizačnému frameworku (Kristián Szobi)

Aplikačný framework je dostatočne flexibilný, takže ho stačí len správne nakonfigurovať. Dokáže chrániť rovnakým spôsobom niekoľko dokumentov (dokument = request na dátovod, súbor, atď) tak, aby mal užívateľ po autorizácii prístup ku všetkým. Rovnako podporuje aj viacero spôsobov autorizácie – každý dokument alebo skupina dokumentov môže byť chránení zvlášť. Legitimita užívateľov sa overuje pomocou tzv. *autentizačného resource*, ktorým býva najčastejšie relačná databáza, XML databáza, LDAP alebo obyčajný XML súbor.

Autentizačný framework obsahuje komponenty, ktoré sa dajú využiť v sitemape. Sú to predovšetkým akcie, ktoré riadia samotné spracovanie http požiadavky v dátovode. Autorizačné skupiny sú vyjadrené pomocou handlerov, ktoré sú priradené k jednotlivým dokumentom. Spravovanie konfigurácií jednotlivých handlerov má na starosti Autentizačný manažér.

O jednotlivých užívateľoch je v session vytvorený objekt authentification. Tento obsahuje okrem základných údajov(id, rola) aj položku data, do ktorej si môžeme ukladať doplňujúce údaje. Framework nám neumožňuje priraďovanie viacero rolí jednému úžívateľovi.

Ak užívateľ nemá prístup k požadovanému dokumentu, môže sa zavolať náhradný dátovod.

Zápis zo stretnutia k tímovému projektu 6/2006

Dátum: 28.3.2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Bartalos

Náplň stretnutia a závery z diskusie

- Na začiatku stretnutia Michal T.:
  - upozornil na niektoré nedostatky v oblasti manažmentu (aktualizácia web stránky).
  - navrhol zápisy robiť do štvrtka, nakoniec bolo dohodnuté robiť ich do stredy.
  - poveril Peťa S., aby spracoval informácie, ktoré sme mu posielali o tom na akých úlohách by sme mali robiť.
  - Bolo dohodnuté, že každý týždeň bude každý vytvárať zoznam úloh, na ktorých by mal pracovať a pošle ich Peťovi S. Na zapísanie týchto úloh sa použije formulár vytvorený Michalom T.
- Diskusia o požadovanom obsahu dokumentácie. Z diskusie vyplynulo, že je vhodné zamerať sa hlavne na portál samotný, než na jeho dokumentáciu.
- Vyhodnotenie úloh a postupu projektu:
- Peter B.:
  - BeanToGraph:
    - Dorobiť vymazávanie a otestovať transformáciu.
  - GraphToBean:
    - Dorobiť trasformáciu multiple atribútov (názov Bean do XML).
- Ján P.:
  - Spolupráca portálu s Factikom.
  - Pozrieť si transformácie a štýly.
  - Pracovať na internacionalizácii portálu.
- Kristián S.:
  - Vytvoril XML s údajmi o používateľoch. Diskutovalo sa o možnosti ukladania týchto dát do databázy. Autorizácia používateľov ostane v XML.
  - Odskúšal možnosti pridávania nových používateľov a ich manažovania. Je potrebné dotiahnuť to do konca - napojiť na GUI portálu.
- Michal T.
  - Prispôsoboval nástroj Factic portálu.
- Michal B.

- Rozmýšlal nad spôsobom uchovávania metadát.
  - Rozhodol sa pre ontológiu, ktorú aj navrhol a začal vytvárať (vlastný kód, nie protege), používa OWL full.
  - Je potrebné zmeniť predikát medzi inštanciou ponuky a metadátami, tak aby metadáta rozširovali ponuku (subject je metainformácia, object je ponuka).

- Peter S.
  - Práca na generovaní bean
    - Dorobiť multiple atribúty – list.
    - Kompilovať kód.
  - Odskúšal tree widget
    - Je to použiteľné.
    - Ešte nevieme bindovať.

- Ďalšia diskusia:
  - Priebeh registrácie sa organizácie:
    - Organizácia si vyplní formulár pre registráciu. Uvedomí sa administrátor (úvaha o spôsobe: mail, ...), ktorý môže následne aktivovať jej konto.
  - Článok IIT-SRC
    - Roman sa vyjadril k chybám v článku a poukázal na potreby dopracovania.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 5.4.2006.

Vyhodnotenie a stav predchádzajúcich úloh

14. 3. 2006	Bartalos (samostatne)	Zadefinovať formát XML súboru pre metadíta potrebné pre Bean2Graph a vytvoriť parser na tieto metadáta.	Vysoká	Ukončená
14. 3. 2006	Bartalos (samostatne)	Umožniť uloženie a úpravu ponúk pomocou Bean2Graph (algoritmus, naplnenie XML metadátami).	Vysoká	Splnená na 90%
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponukt.	Stredná	Splnená na 25%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigáurácia).	Vysoká	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	Splnená na 75%
14. 3. 2006	Szobi (Tvarožek)	Definovať funkcionalitu jednotlivých obrazoviek (aj offeredBy/Via).	Vysoká	Ukončená
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Splnená na 75%
21. 3. 2006	Szobi (samostatne)	Preložiť portál do angličtiny, výstup priamo do SVN.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Nezačatá
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá

Nové úlohy

- Úsilie sa sústreďuje hlavne na dokončenie úloh zadefinovaných na minulých stretnutiach a prerobenie článku na IIT-SRC.

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
	Barla		Nízka	Nová
5. 4. 2006	Bartalos	Transformácia GraphToBean	Vysoká	Splnená na 75%
	Porubský		Nízka	Nová
	Sivák		Nízka	Nová
	Szobi		Nízka	Nová
	Tvarožek		Nízka	Nová

Zápis zo stretnutia k tímovému projektu 7/2005

Dátum: 4. 4. 2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Filkorn

Miesto: D209

Zapísal: Sivák

Náplň stretnutia a závery z diskusie

- Vyhodnotenie úloh a postupu projektu:
- Peter B.:
  - Bean-to-graph a graph-to-bean sú dokončené a čiastočne otestované.
  - Treba testovať na vygenerovaných beanach, keď budú dostupné.
  - Problém s multiple range sa dá riešiť definovaním rodičovskej triedy alebo pridávaním prefixov, ci postfixov do názvov členských premenných. Rozhodli sme sa pre druhú možnosť, lebo očakávame problémy pri zisťovaní typu elementov v zozname.
- Kristián S.:
  - Rieši prepojenie registračného formulára s triedou, ktorá vykonáva manažment používateľov.
  - Napr. ConfigurationGeneration.AddRole() je private.
- Michal B.:
  - Zmenil v algoritme, ktorý generuje formulár, zisťovanie vlastností (zisťovali sa z inštancií, čo spôsobovalo, že sa nenašli všetky)
  - Factic nefunguje v portáli, ale funguje mimo neho. Zrejme problém s mapovaním URL adries.
  - Riešiť zisťovanie typu ovládacieho prvku (widget) z ontológie.
- Ján P.:
  - Pracovať na štýloch a internacionalizácii.
- Peter S.:
  - Pracuje na generovaní bean.
  - Treba riešiť multiple range a vygenerovať metadáta pre bean-to-graph.
  - Databáza hsql sa dá ľahko použiť.
  - Spísal zoznam úloh.
- Ďalšia diskusia:
  - Ak sa nepodarí vyriešiť autorizáciu v Cocoone, budeme zvažovať návrat k Jetspeedu.
  - Riešiť možnosť zadefinovať šablónu formulára pre zadávanie pracovnej ponuky pre určitú skupinu ľudí – niektoré ovládacie prvky sa skryjú a priradí sa im implicitná hodnota.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 11. 4. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 3. 2006	Bartalos (samostatne)	Umožniť uloženie a úpravu ponúk pomocou Bean2Graph (algoritmus, naplnenie XML metadátami).	Vysoká	Ukončená
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponukt.	Stredná	Splnená na 25%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigáurácia).	Vysoká	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	Splnená na 75%
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Splnená na 75%
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Splnená na 50%
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá

Plán budúcich úloh

- Úsilie sa sústreďuje najmä na dokončenie úloh zadefinovaných na minulých stretnutiach a odoslanie článku na IIT-SRC 2006.

Zápis zo stretnutia k tímovému projektu 8/2006

Dátum: 11. 4. 2006

Prítomní: Barla, Bartalos, Porubský, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Szobi

Náplň stretnutia a závery z diskusie

- Vyhodnotenie úloh a postupu projektu:
- Peter B.:
  - Použitie vygenerovaných beanov od Peťa B.
  - Testoval transformácie GraphToBean a BeanToGraph. Výsledky vyzerajú dobre.
  - Problém s uploadovaním beany pri použití vocabulary
- Kristián S.:
  - Rieši user management. Podarilo sa volanie vlastných dátovodov pomocou Cocoon API
  - Problém s UserBean implementáciou v Cocoone – nemá implementované settery
- Michal B.:
  - Spravil model metadát
  - Pracoval s Kristiánom S. na user managemente
- Ján P.:
  - Zmenil štýly vo Facticu, aj generovanie HTML v XSLT šablóne
  - Spojil štýly do page.css
  - Po refreshi stránky prihlásených užívateľom sa už Factic nezobrazí
- Peter S.:
  - Vygeneroval beany.
  - Spísal zoznam úloh.
- Ďalšia diskusia:
  - Diskusia o metadátach v ontológií. Ako komplexne by mali zachytávať rozloženie grafický komponentov.
  - Atribúty metadát z ontológie bude potrebné generovať do XML
  - Budúci týždeň – prezentácia na PW
  - Do 17.4. treba poslať abstrakt na konferenciu

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 18. 4. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponukt.	Stredná	Splnená na 25%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigáurácia).	Vysoká	Splnená na 90%
14. 3. 2006	Porubský (samostatne)	Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	Zrušená
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Splnená na 75%
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Splnená na 75%
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá

Plán budúcich úloh

- Úsilie sa sústreďuje najmä na dokončenie úloh zadefinovaných na minulých stretnutiach.

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
18. 4. 2006	Szobi (Barla)	Vytvoriť formulár (model, template, bean a binding) pre žiadosť o registráciu, ktorý pošle údaje mailom na adresu, ktorá je v konfiguračnom xml súbore	Vysoká	Nová
18. 4. 2006	Szobi (samostatne)	Zdigitalizovať poznámky k článku	Vysoká	Nová
18. 4. 2006	Szobi (Barla)	Spraviť triedu, ktorá používa hsql databázu (príp. xml súbor) na ukladanie informácií o používateľoch a rolách	Nízka	Nezačatá
18. 4. 2006	Tvarožek (Barla)	Zobrazenie ponúk ktoré zadala len daná organizácia, tlačidlo Edit na načítanie z BeanToGraph	Vysoká	Nová
18. 4. 2006	Tvarožek (Barla)	Factic napojiť na HSQL databázu (adaptér na neotologické dáta), vytvoriť tlačidlo edit do facticu: po stlačení sa spustí flow na načítanie obsahu ponuky z graph-to-bean a zobrazí sa formulár, zobrazenie neotologických dát	Vysoká	Nová
18. 4. 2006	Tvarožek (Szobi, Porubský)	Vytvoriť sitemap podľa navrhnutého navigačného modelu, vytvoriť prázdne coplety podľa navrhnutého navigačného modelu	Vysoká	Nová
18. 4. 2006	Sivák (samostatne)	Bindovanie tree widgetu	Stredná	Nová
18. 4. 2006	Sivák (Barla)	Vytvoriť triedu, ktorá z xml súboru načíta strom a poskytne ho tree widgetu	Stredná	Nová
18. 4. 2006	Porubský (Szobi)	Vytvoriť slovník pre anglický jazyk, namiesto konkrétnych textov v copletoch vkladať iba odkazy do slovníka	Stredná	Nová

Zápis zo stretnutia k tímovému projektu 9/2006

Dátum: 18. 4. 2006

Prítomní: Barla, Bartalos, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Barla

Náplň stretnutia a závery z diskusie

- Vyhodnotenie úloh a postupu projektu:
- Peter B.:
  - Pokračuje v úpravách vygenerovaných bean a metadát
    - Zopár nekonzistentných názvov medzi xml a beanou
    - Niektoré objektové vlastnosti nie sú aktuálne reprezentované samostatnou beanou
- Kristián S.:
  - Pokračoval v práci na user managemente (pridávanie používateľov)
    - Vytvoril vlastnú implementáciu UserBean, ktorá má potrebné set metódy – nutný predpoklad pre úspešný binding
    - Napriek tomu binding z repeateru do beany nefungoval → vyriešené na stretnutí, objekt do ktorého sa riadok z repeateru binduje musí mať atribút ID a príslušné get a set metódy
  - Bude sa venovať úpravám formulárov user managementu – výber používateľa, ktorého chceme editovať
  - Problém s I18n
    - Zatiaľ sa to správa značne nedeterministicky
- Peter S.:
  - Spravil triedu, využívanú tree widgetom, ktorá strom nemá definovaný priamo v kóde ale v XML súbore
    - XML súbor je zatiaľ písaný ručne, nie je generovaný
- Ján P.:
  - Snažil sa riešiť problémy internacionalizácie, zatiaľ neúspešne
    - Na výstupe sú kľúče do slovníka a nie ich odpovedajúce preklady
- Michal T.:
  - Do Facticu pridal zoraďovanie podľa spoločnosti
    - Keďže predpokladáme, že sa používateľovi zobrazia len jeho ponuky, resp. ponuky spoločnosti, bude zrejme potrebné zoraďovať podľa iných vlastností (dátum, kto pridal a pod.)
  - Príprava Facticu na zobrazovanie neontologických dát

- Dátum pridania, stav ponuky, kto ju pridal

➢ Pridanie edit tlačidla do Facticu
- Zavolá URL s parametrami namespace a id
- Predpokladajú sa ďalšie parametre, ktoré zachytia stav Facticu, aby sa doň dalo z formulára vrátiť

❖ Michal B.:

➢ Prerobil model metadát podľa posledných požiadaviek
- Práca na generovaní je v tejto fáze projektu pozastavená, predpokladá sa dokončenie v posledných dvoch týždňoch semestra

➢ Vytvorenie flow-u a úpravy sitemapy, ktoré zachytia stlačenie edit vo Facticu a preberú parametre – URI ponuky, ktorá sa má zobraziť vo formulári

➢ Spolu s Michalom T. vyriešili nekorektné správanie Facticu v portáli – chýbajúce obrázky, zlé smerovanie hyperlinks

❖ Ďalšia diskusia:

➢ Diskusia o prezentáciách, ktoré tím čakajú
- PeWe, 20.4. 10:10 – prezentovať budú Peter S. a Michal B.
- IIT.SRC, 26.4 – prezentovať bude predbežne Kristián

➢ HSQL databáza by mala byť ľahko vymeniteľná za inú

➢ O týždeň sa odovzdáva produkt
- Zintenzívnime stretnutia a hlásenia postupu prác

➢ To Do:
- Formulár pracovnej ponuky: prepojiť formulár a g2b, b2g – uzavrieť živ. cyklus (binding, flow), napĺňať aj neontologické dáta
- Spraviť formulár pre registráciu organizácie
- Upraviť formuláre pre user management
- Upraviť menu – niektorá funkcionalita dostupná cez tools má byť vyvedená priamo do menu
- Upraviť štýly portálu – to je to, čo nás v konečnom dôsledku „predáva“
- Upraviť Factic tak, aby poskytoval len prehľady ponúk organizácie/jednotlivca (zrejme v závislosti od rolí) – potrebné údaje Factic získa z UserBeany dostupnej cez cocoonAPI, pri tomto sa predpokladá využitie Kristiánovho know-how o user managemente.

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 20. 4. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponuky.	Stredná	Splnená na 25%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigurácia).	Vysoká	Splnená na 90%
14. 3. 2006	Porubský (samostatne)	Nájsť možnosti riešenia problému s nástrojom na správu používateľov.	Vysoká	Zrušená
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Splnená na 75%
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Splnená na 75%
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá

Plán budúcich úloh

- Úsilie sa sústreďuje najmä na dokončenie úloh zadefinovaných na minulých stretnutiach.

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
18. 4. 2006	Szobi (Barla)	Vytvoriť formulár (model, template, bean a binding) pre žiadosť o registráciu, ktorý pošle údaje mailom na adresu, ktorá je v konfiguračnom xml súbore	Vysoká	Nezačatá
18. 4. 2006	Szobi (samostatne)	Zdigitalizovať poznámky k článku	Vysoká	Zrušená
18. 4. 2006	Szobi (Barla)	Spraviť triedu, ktorá používa hsql databázu (príp. xml súbor) na ukladanie informácií o používateľoch a rolách	Nízka	Nezačatá
18. 4. 2006	Tvarožek (Barla)	Zobrazenie ponúk ktoré zadala len daná organizácia, tlačidlo Edit na načítanie z BeanToGraph	Vysoká	Nová
18. 4. 2006	Tvarožek (Barla)	Factic napojiť na HSQL databázu (adaptér na neotologické dáta), vytvoriť tlačidlo edit do facticu: po stlačení sa spustí flow na načítanie obsahu ponuky z graph-to-bean a zobrazí sa formulár, zobrazenie neotologických dát	Vysoká	Nezačatá
18. 4. 2006	Tvarožek (Szobi, Porubský)	Vytvoriť sitemap podľa navrhnutého navigačného modelu, vytvoriť prázdne coplety podľa navrhnutého navigačného modelu	Vysoká	Nová
18. 4. 2006	Sivák (samostatne)	Bindovanie tree widgetu	Stredná	Splnená na 90%
18. 4. 2006	Sivák (Barla)	Vytvoriť triedu, ktorá z xml súboru načíta strom a poskytne ho tree widgetu	Stredná	Ukončená
18. 4. 2006	Porubský (Szobi)	Vytvoriť slovník pre anglický jazyk, namiesto konkrétnych textov v copletoch vkladať iba odkazy do slovníka	Stredná	Nová

Zápis zo stretnutia k tímovému projektu 10/2006

Dátum: 25. 4. 2006

Prítomní: Barla, Bartalos, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Tvarožek

Náplň stretnutia a závery z diskusie

- V piatok sa podaril binding z formulára (PB+MB)
- Peter Bartalos
  - Upratovanie kódu, refaktoring, properties súbor pre repository
- Kristián Szobi
  - Coplet management už funguje (piatok)
  - User management
    - Ide AddUser, EditUser, DeleteUser
    - Hierarchia rolí pre používateľov
    - Password je ****
    - Každý používateľ môže mať enabled/disabled konto
- Michal Barla
  - CachingURI coplets
    - Bolo treba vedieť presmerovať, čo sa podarilo - invalid cache a meniť temporary-uri
  - Generalizácia hardkódovaných ciest
  - Flow a prepínanie copletov už celkom ide (Factic-Form a späť)
- Peter Sivák
  - TreeWidget
  - HSQL adaptér pre ukladanie dát o ponukách
- Ján Porubský (neprítomný)
  - Zmena rozhrania (layoutu) portálu
  - Aktualizované menu
  - Doplnené obrázky
- Michal Tvarožek
  - Internacionalizácia cez i18n
  - HSQL adaptér integrovaný do Facticu
  - Nove štýly XSL, CSS pre Factic
  - Integrácia s Facticu s OfferForm

- Dokumentácia
  - Predbežná štruktúra technickej dokumentácie
  - Draft používateľskej dokumentácie

❖ ToDo:

- Pripraviť mail o testovaní pre NAZOU
- 1.máj ISD článok
- Používateľskú dokumentáciu dokončiť do 2.-3. mája
  - Minimalizovať úsilie, ako pre trošku inteligentných
  - 5-10 strán ~ obrázky
  - Dať ju aj do časti Help v portáli
  - Zrušiť powerusera
  - Zamerať sa tak, aby sa dalo robiť testovanie systému
- Ukladanie nastavení skins
- Skúsiť RSS ak je v Cocoone nejaký sample (FIIT, Pravda, …)
- Štvrtok bude prvá verzia článku pre Romana Filkorna
- Prejdeme naspäť na Cocoon 2.1.8, aby sme mali všetci rovnakú verziu

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 26. 4. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Splnená na 75%
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponuky.	Stredná	Splnená na 90%
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Splnená na 75%
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigurácia).	Vysoká	Splnená na 90%
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Splnená na 90%
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Splnená na 75%
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá
18. 4. 2006	Bartalos (Barla)	Vytvoriť formulár (model, template, bean a binding) pre žiadosť o registráciu, ktorý pošle údaje mailom na adresu, ktorá je v konfiguračnom xml súbore	Vysoká	Splnená na 50%
18. 4. 2006	Sivák (Barla)	Spraviť triedu, ktorá používa hsql databázu (príp. xml súbor) na ukladanie informácií o používateľoch a rolách	Nízka	Ukončená
18. 4. 2006	Tvarožek (Barla)	Zobrazenie ponúk ktoré zadala len daná organizácia, tlačidlo Edit na načítanie z BeanToGraph	Vysoká	Splnená na 75%

18. 4. 2006	Tvarožek (Barla)	Factic napojiť na HSQL databázu (adaptér na neontologické dáta), vytvoriť tlačidlo edit do facticu: po stlačení sa spustí flow na načítanie obsahu ponuky z graph-to-bean a zobrazí sa formulár, zobrazenie neontologických dát	Vysoká	Splnená na 90%
18. 4. 2006	Tvarožek (Szobi, Porubský)	Vytvoriť sitemap podľa navrhnutého navigačného modelu, vytvoriť prázdne coplety podľa navrhnutého navigačného modelu	Vysoká	Splnená na 75%
18. 4. 2006	Sivák (samostatne)	Bindovanie tree widgetu	Stredná	Splnená na 90%
18. 4. 2006	Tvarožek (Szobi)	Vytvoriť slovník pre anglický jazyk, namiesto konkrétnych textov v copletoch vkladať iba odkazy do slovníka	Stredná	Splnená na 90%

Plán budúcich úloh

- Úsilie sa sústreďuje najmä na dokončenie úloh zadefinovaných na minulých stretnutiach.

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
3. 5. 2006	Tvarožek (samostatne)	Pripraviť používateľskú dokumentáciu.	Stredná	Nová
1. 5. 2006	Barla (všetci)	Článok na ISD.	Vysoká	Nová

Zápis zo stretnutia k tímovému projektu 11/2006

Dátum: 2. 5. 2006

Prítomní: Barla, Bartalos, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Sivák

Náplň stretnutia a závery z diskusie

- Odprezentovali sme článok na konferencii IIT-SRC (KS)
- V piatok sme nainštalovali portál pre pedagog. vedúceho
- Poslali sme článok na konferenciu ISD (MB, MT, PB)
- V portálovej časti treba ešte dokončiť registráciu
- Používateľskú dokumentáciu treba odovzdať na konci projektu
- Kristián Szobi
  - Pracoval na manažmente používateľov
  - Treba vyriešiť right management
  - V dokumentácii bude treba spísať skúsenosti s manažmentom používateľov tak, aby sa dali použiť v štátnom projekte
- ToDo:
  - Prerequisites do formulára pracovnej ponuky
  - Treba predviesť portál zástupcovi tímu č.10
  - Treba vyriešiť odosielanie URL rodičovského uzla označeného uzla v tree widgete
- Ďalej sa budeme sústreďovať na generovanie formulára
  - Treba spraviť metódu, ktorá pre zadané URI OWL entity vráti typ widgetu
  - Implementovať metamodel
  - Upraviť generovanie java bean a mapovacích pravidiel tak, aby neboli potrebné ručné zmeny
  - Upraviť ontológiu (napr. pridať vlastnosti FullyDefinedBySubclasses)

Ďalšie stretnutie

Ďalšie stretnutie je naplánované na 9. 5. 2006.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
14. 3. 2006	Sivák (samostatne)	Spracovať formulár pracovnej ponuky (XMLká).	Vysoká	Ukončená
14. 3. 2006	Sivák (Bartalos)	Prepojiť formulár pracovnej ponuky s metadátovým algoritmom na ukladanie ponuky.	Stredná	Ukončená
14. 3. 2006	Porubský (samostatne)	Dokončiť konfiguráciu copletov.	Stredná	Ukončená
14. 3. 2006	Porubský (samostatne)	Grafický návrh copletov a obrazoviek (CSS, obrázky, konfigurácia).	Vysoká	Ukončená
21. 3. 2006	Szobi (samostatne)	Návrh štruktúry adresárov portály podľa navigačného modelu, skúsiť navrhnúť sitemapy a stručne opísať v dokumente.	Stredná	Zrušená
9. 5. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Identifikovať a navrhnúť spôsob zápisu metadát pre generovanie formulára.	Stredná	Ukončená
16. 5. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
16. 5. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 25%
9. 5. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Splnená na 90%
22. 5. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
16. 5. 2006	Tvarožek (samostatne)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Nezačatá
18. 4. 2006	Bartalos (Barla)	Vytvoriť formulár (model, template, bean a binding) pre žiadosť o registráciu, ktorý pošle údaje mailom na adresu, ktorá je v konfiguračnom xml súbore	Vysoká	Zrušená
18. 4. 2006	Sivák (Barla)	Spraviť triedu, ktorá používa hsql databázu (príp. xml súbor) na ukladanie informácií o používateľoch a rolách	Nízka	Ukončená
9. 5. 2006	Tvarožek (Barla)	Zobrazenie ponúk ktoré zadala len daná organizácia, tlačidlo Edit na načítanie z BeanToGraph	Vysoká	Splnená na 75%

18. 4. 2006	Tvarožek (Barla)	Factic napojiť na HSQL databázu (adaptér na neontologické dáta), vytvoriť tlačidlo edit do facticu: po stlačení sa spustí flow na načítanie obsahu ponuky z graph-to-bean a zobrazí sa formulár, zobrazenie neontologických dát	Vysoká	Ukončená
18. 4. 2006	Tvarožek (Szobi, Porubský)	Vytvoriť sitemap podľa navrhnutého navigačného modelu, vytvoriť prázdne coplety podľa navrhnutého navigačného modelu	Vysoká	Ukončená
18. 4. 2006	Sivák (samostatne)	Bindovanie tree widgetu	Stredná	Ukončená
18. 4. 2006	Tvarožek (Szobi)	Vytvoriť slovník pre anglický jazyk, namiesto konkrétnych textov v copletoch vkladať iba odkazy do slovníka	Stredná	Ukončená
22. 5. 2006	Tvarožek (samostatne)	Pripraviť používateľskú dokumentáciu.	Stredná	Splnená na 25%
1. 5. 2006	Barla (všetci)	Článok na ISD.	Vysoká	Ukončená

Plán budúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
4. 5. 2006	Tvarožek (samostatne)	Vytvoriť a poslať kostru dokumentácie	Stredná	Nová
4. 5. 2006	Sivák (Szobi)	Pridať Prereguisites do formulára	Stredná	Nová
9. 5. 2006	Sivák (samostatne)	Tree widget do registrácie nového použ.	Stredná	Nová
9. 5. 2006	Barla (samostatne)	Implementovať metamodel	Stredná	Nová

Zápis zo stretnutia k tímovému projektu 12/2006

Dátum: 25. 4. 2006

Prítomní: Barla, Bartalos, Sivák, Szobi, Tvarožek, Filkorn

Miesto: D209

Zapísal: Bartalos

Náplň stretnutia a závery z diskusie

- Peter Bartalos
  - Práca na prerekvizitách – beana + mapovanie
- Michal Barla + Michal Tvarožek
  - Algoritmus na zisťovanie typu widgetu. Vytvorené tri triedy pre jednotlivé typy + 1 abstraktná.
- Peter Sivák
  - Práca na webe tímu
  - Generovanie model a template
- Diskutovalo sa o rôznych veciach súvisiacich s generovaním formulárov a prácou potrebnou na dokončenie projektu.

Ďalšie stretnutie

Ďalšie stretnutie nie je naplánované.

Vyhodnotenie a stav predchádzajúcich úloh

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
21. 3. 2006	Sivák (samostatne)	Generovanie beany a mapovacích metadát pre Bean2Graph.	Vysoká	Splnená na 90%
21. 3. 2006	Barla (samostatne)	Návrh a implementácia triedy pre načítanie obsahu ovládacieho prvku.	Vysoká	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Serializácia objektov do XML a generovanie modelu, templatu a bindingu z vnútornej reprezentácie.	Stredná	Splnená na 75%
21. 3. 2006	Sivák (Barla)	Kompilácia vygenerovanej beany.	Stredná	Splnená na 90%
21. 3. 2006	Sivák (Barla)	Integrácia častí generovania formulárov.	Stredná	Nezačatá
21. 3. 2006	Sivák (Barla)	Rozpoznávanie ovládacích prvkov v ontológii.	Stredná	Ukončená
18. 4. 2006	Tvarožek (Barla)	Zobrazenie ponúk ktoré zadala len daná organizácia, tlačidlo Edit na načítanie z BeanToGraph	Vysoká	Splnená na 75%

Plán budúcich úloh

- Úsilie sa sústreďuje najmä na dokončenie úloh zadefinovaných na minulých stretnutiach.

Predpokladané ukončenie	Riešitelia	Opis úlohy	Priorita	Stav riešenia
3. 5. 2006	Tvarožek (samostatne)	Pripraviť používateľskú dokumentáciu.	Stredná	Nová

6. Posudky a vyjadrenia k posudkom

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

ANALÝZA TEXTOV PRACOVNÝCH PONÚK Z PROSTREDIA WEBU

POSUDOK PROTOTYPU A DOKUMENTÁCIE TÍMU Č. 10

Tím č. 8: Hachiban
Vedúci tímu: tvarozek@gmx.net
Mailový alias: tp@atrip.sk
Predmet: Tvorba softvérového systému v tíme
Pedag. vedúci: Ing. Roman Filkorn
Ak. rok: 2005/2006

Členovia tímu:
Bc. Michal Barla
Bc. Peter Bartalos
Bc. Ján Porubský
Bc. Peter Sivák
Bc. Kristián Szobi
Bc. Michal Tvarožek

Úvod

Predkladaný dokument obsahuje posudok pilotnej aplikácie a súvisiacej projektovej dokumentácie projektu, ktorý rieši tím č. 10 v rámci predmetu *Tvorba softvérového systému v tíme* v akademickom roku 2005/2006. Posudok sa vzťahuje na výstupy vytvorené medzi prvým a druhým kontrolným bodom. Tím sa v tejto etape riešenia projektu rozhodol vytvoriť a zdokumentovať pilotnú aplikáciu, ktorá pozostáva z troch obaľovačov portálov pracovných ponúk a z následnej integrácie a prezentácie získaných dát. Obaľovače boli vytvorené v prostredí Lixto Visual Wrapper.

Posudzovaná dokumentácia sa skladá z dvoch častí: z technickej dokumentácie a z používateľskej príručky. Členenie technickej dokumentácie vychádza z architektúry pilotnej aplikácie. V jednotlivých podkapitolách sú postupne opísané všetky tri úrovne spracovania dát. Používateľská príručka poskytuje návod na inštaláciu prostredia a spustenie aplikácie, ktorej výstupom sú statické stránky HTML. Tie sú určené na prehľadné zobrazenie získaných pracovných ponúk.

Tento posudok nasleduje štruktúru posudzovanej dokumentácie. Prvá kapitola je zameraná na technickú dokumentáciu, druhá kapitola opisuje používateľskú príručku pilotnej aplikácie. Celkové zhodnotenie sa nachádza v tretej kapitole.

1. Technická dokumentácia

Technická dokumentácia k pilotnej aplikácií, ktorú sme posudzovali v tejto etape riešenia projektu sa nachádza v kapitole *5. Pilotná aplikácia* dokumentácie. Je rozdelená na tri časti: úroveň obaľovačov, úroveň integrácie a úroveň prezentácie. Zvolené členenie hodnotíme ako vhodné. V kapitole by však bolo dobré v úvodnej časti zosumarizovať, ktoré z navrhnutých častí architektúry autori nakoniec prototypovali.

1.1. Úroveň obaľovačov

V tejto časti sa nachádza opis troch obaľovačov implementovaných v prostredí Lixto Visual Wrapper. Každý obaľovač bol vytvorený pre jeden portál pracovných ponúk. Cieľom tejto úrovne prototypu bolo (podľa kap. 4.6 z prvého kontrolného bodu) oboznámenie sa autorov s problémami, ktoré sa vyskytujú v procese získavania dát a s ich riešeniami. Dôležitým prínosom tejto úrovne je získanie testovacej bázy dát, ktorá sa neskôr bude dať využiť pri porovnávaní výsledkov.

V technickej dokumentácii sa pre každý obaľovač uvádza hierarchická štruktúra vzorov pre každý obaľovaný portál, vzťah vzorov k výstupnému XML dokumentu a stručné zhodnotenie obaľovača. Pri opise jedného z obaľovačov sa nachádzajú aj ukážky webovej stránky, na ktorej bol obaľovač testovaný. Podobné ukážky by mohli obsahovať aj opisy ostatných obaľovačov. Čitateľ by tak získal lepšiu predstavu o tom, prečo bola zvolená práve uvedená hierarchia vzorov.

Pri návrhu pilotnej aplikácie riešitelia analyzovali viaceré portály pracovných ponúk a vybrali z nich tri, ktoré sú odstupňované podľa náročnosti získavania dát. Z pôvodného zoznamu sa však obaľovač implementoval iba pre jeden portál *(LinkedIn)*. Pri nahradení portálu *DayJob.com* za *Career Builder* nebolo uvedené žiadne vysvetlenie zmeny. Pri zmene portálu *Executives on the Web* za *EuroJobs* bola ako dôvod uvedená prílišná zložitosť získavania dát a ich nevhodná štruktúra. Na mieste je otázka, či bol dodržaný jeden z cieľov prototypu – oboznámenie sa s problémami a pokus o ich riešenie. Zo zhodnotenia jednotlivých obaľovačov sme nadobudli dojem, že implementácia všetkých troch bola

v podstate jednoduchá. Preto by sme odporúčali riešiteľom aby spoločne vytvorili ešte jeden obaľovač, kde by si vyjasnili problémy s autentifikáciou, JavaScriptom a cookies.

V zhodnoteniach jednotlivých obaľovačov nám chýbalo najmä vyhodnotenie získaných skúseností v súvislosti s návrhom systému, čo vidíme ako vážny nedostatok dokumentácie. Problémom môže byť aj skutočnosť, že v dvoch implementovaných obaľovačoch bolo nutné použiť dodatočnú vlastnosť nástroja Lixto Visual Wrapper *RegExp mangling*. V dokumentácii nie je uvedené, ako sa k tomuto problému stavia návrh systému – či je problém riešený iným spôsobom alebo sa predpokladá dopracovanie návrhu.

1.2. Úroveň integrácie

Táto časť opisuje spôsob integrácie výsledkov získaných počas procesu obaľovania. Jej cieľom bola extrakcia zadefinovaných dátových položiek z výstupov obaľovania rôznych portálov a ich integrácia do jedného XML dokumentu s jednotne zadefinovanými elementmi.

Autori uvádzajú opis elementov pracovnej ponuky v XML súbore spolu s informáciou, či sú pre pracovnú ponuku povinné. Ďalej autori opisujú proces spracovania údajov, ktorý spočíval v preformátovaní dátumov na jednotný formát, v premapovaní priemyselných odvetví podľa zvolenej kategorizácie a v odfiltrovaní prázdnych ponúk.

Pri mapovaní autori uvádzajú formát súboru XML, na základe ktorého sa mapovanie vykonáva. Bolo by názornejšie, keby uvedený príklad obsahoval ukážku reálneho súboru. Z príkladu nie je jasné, či autori počítajú s prípadom, keď jedna pracovná ponuka patrí do viacerých priemyselných odvetví. Štruktúry súborov XML bude pravdepodobne potrebné ešte prepracovať, najmä v súvislosti s využitím výsledkov projektu v rámci programu NAZOU..

Autori uvádzajú, že niektoré obaľovače získavajú aj prázdne ponuky, pričom toto správanie bližšie neanalyzujú. Opisujú iba opatrenie, ktoré zabezpečí, aby sa tieto ponuky ďalej nespracovávali. Bolo by vhodné zistiť, či ide o chybu obaľovača alebo sa prázdne ponuky nachádzajú priamo na obaľovanom portáli.

Prototypovanie tejto časti aplikácie hodnotíme ako užitočné, pretože sa autori stretli s problémami, ktoré nastali pri integrovaní výsledkov obaľovania rôznych portálov. Výsledkom prototypovania je zistenie, že použitie samostatných transformácií XSLT nie je najvhodnejšie riešenie a je potrebné pridať do návrhu systému samostatný programový modul, ktorý podporí prípravu na integráciu.

Spôsob integrácie všetkých údajov naraz pri väčšom množstve údajov nie je dostatočne efektívny, čo si uvedomili aj riešitelia projektu, keď do návrhu pilotnej aplikácie zahrnuli aj vrstvu „ďalšie spracovanie údajov“. Táto vrstva však nakoniec bez vysvetlenia nebola v prototype implementovaná. V prípade, že potrebujeme k existujúcim dátam pridať dáta získané z ďalšieho portálu, potrebujeme spracovať znova aj už spracované dáta. Aktuálne sú všetky tri portály spracované pomocou šablóny zapísanej v jednom súbore. Za vhodnejšie riešenie pokladáme vytvorenie jednej šablóny, ktorá by zabezpečovala všeobecnejšiu transformáciu a následné vytvorenie šablón pre každý konkrétny obaľovač, pre transformáciu portálovo závislých údajov. Toto riešenie súčasne umožní aj nasadenie inkrementálnej integrácie.

1.3. Úroveň prezentácie

V tretej podkapitole autori opisujú prezentačný nástroj, ktorý navrhli a implementovali s cieľom prehľadne zobraziť zoznam získaných pracovných ponúk. Keďže výstupy obaľovačov sú zvyčajne spracúvané ďalšími aplikáciami, nie je ich grafická prezentácia v tomto projekte dôležitá. Následne tento nástroj nie je určený na prezentovanie dát koncovým používateľom, ale slúži skôr na odhaľovanie chýb vzniknutých pri vývoji jednotlivých častí

systému. Napriek tomu si myslíme, že vhodná prezentácia získaných dát je pre koncových používateľov užitočná, a to najmä pri ladení riadiacich programov. Autori sa preto mohli viac venovať takému návrhu architektúry nástroja, ktorým by sa zvýšila jeho tvárnosť.

Tvorba serverového nástroja na generovanie stránok HTML v reálnom čase sa ukázala ako neefektívna, kvôli dlhej dobe odozvy. Po tomto zistení autori zvažovali dve alternatívy: preniesť získané dáta do SQL databázy a využiť jej indexovacie mechanizmy na zrýchlenie generovania stránok alebo *off-line* generovanie stránok. Autori zvolili druhé z riešení bez bližšieho zdôvodnenia, pravdepodobne z dôvodu, že implementácia riešenia pomocou SQL servera by im trvala dlhšie a v tejto etape nebola nevyhnutná.

Jadrom prezentačného nástroja je XSLT šablóna opisujúca transformáciu XML dokumentu, ktorý je výsledkom integračnej časti aplikácie, na množinu stránok HTML.

2. Používateľská príručka

Používateľská príručka obsahuje informácie potrebné na spustenie každej časti pilotnej aplikácie. Oceňujeme, že autori do príručky zahrnuli aj návod na inštaláciu prostredia Lixto Visual Wrapper. Samotné spustenie obaľovača by však mohlo byť opísané trochu podrobnejšie. Z dokumentácie nie je úplne jasné, ktorý súbor obsahuje definíciu obaľovača a aké parametre vyžaduje program *wrap* na svoje spustenie. Ostatné časti príručky boli spracované prehľadne, veľmi oceňujeme použité obrázky.

Obrázky odporúčame dať aj do častí, kde sa opisuje grafické používateľské prostredie, či už ide o program Lixto Visual Wrapper alebo o stránky tvoriace výstup pilotnej aplikácie, kde by sa obrázok obzvlášť hodil.

3. Zhodnotenie

Celkovo hodnotíme prototypovanie dobre, aj keď boli vytvárané najmä prototypy na zahodenie. Autori nepochybne pri tvorbe pilotnej aplikácie vynaložili veľa úsilia, ktoré však v ďalšej fáze zúročia iba v podobe získaných skúseností. Keďže však autori doteraz nemali skúsenosti s problematikou obaľovačov bolo práve vyskúšanie si celého životného cyklu obaľovača významným a prínosným krokom k pochopeniu problematiky.

Celkovo autori mohli využiť zdroje aj efektívnejšie – nepridelíť na tvorbu obaľovačov až troch členov tímu. Ušetrené zdroje mohli využiť pri prototypovaní niektorých ďalších častí navrhovaného systému. Myslíme si, že v návrhu sú moduly, ktorých čiastočné prototypovanie by tím posunulo ďalej.

Z hľadiska vytýčených cieľov prototypu nám v dokumentácii chýba lepšie zhodnotenie nadobudnutých skúseností a vyjadrenie sa smerom k pôvodnému návrhu. Riešitelia nezrealizovali „ťažký“ obaľovač tak, ako to mali naplánované. Chýba vyhodnotenie tohto faktu – znamená to, že výsledný systém nebude určený pre obaľovanie stránok s *Cookies* a použitou technológiou *JavaScript* alebo sa tomuto problému hodlajú riešitelia ešte venovať?

Autori sa mohli tiež venovať vhodnosti, resp. nevhodnosti používateľského rozhrania použitého nástroja na tvorbu obaľovačov z hľadiska potrieb používateľov a vyvodiť závery pre ich aplikáciu v podobne rád a odporúčaní pre tvorbu vlastného používateľského rozhrania.

Formálna stránka dokumentu je na dobrej úrovni, našli sme však niekoľko nedostatkov, súvisiacich s integráciou dokumentu. Napríklad v častiach o implementácii jednotlivých obaľovačov je vidieť snaha o spoločný formát opisu obaľovača, ale niektoré rozdiely sú aj tak badateľné. Obrázky spracovávanej stránky boli iba pri jednom obaľovači.

Celkovo hodnotíme vykonanú prácu ako veľmi dobrú, najmä keď zvážime, že riešitelia pracovali v pre nich novej oblasti.

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

VYJADRENIE K POSUDKU TÍMU Č. 10

(K DOKUMENTÁCII PO PRVOM KONTROLNOM BODE)

Tím č. 8: Hachiban
Vedúci tímu: tvarozek@gmx.net
Mailový alias: tp@atrip.sk
Predmet: Tvorba softvérového systému v tíme
Pedag. vedúci: Ing. Roman Filkorn
Ak. rok: 2005/2006

Členovia tímu:
Bc. Michal Barla
Bc. Peter Bartalos
Bc. Ján Porubský
Bc. Peter Sivák
Bc. Kristián Szobi
Bc. Michal Tvarožek

Úvod

Predkladaný dokument obsahuje vyjadrenie k posudku projektovej dokumentácie k softvérovému systému vytvorenému v rámci predmetu Tvorba softvérového systému v tíme v akademickom roku 2005/2006. Vyjadrenie sa vzťahuje na posudok odovzdaný tímom č. 10 na našu dokumentáciu odovzdanú v prvom kontrolnom bode. Dokument sa skladá zo dvoch kapitol zodpovedajúcich štruktúre posudku.

1. Formálna stránka

V posudku sa konštatuje, že dokumentácia je prehľadne štruktúrovaná a celkovo vhodne naformátovaná. Posudok oprávnene upozorňuje na viaceré nedostatky vo forme preklepov, ktoré bohužiaľ aj napriek kontrole vo výslednom texte zostali. Úmyselne sme nepoužili 1,5 násobné riadkovanie, aby sme nepredĺžili už teraz značne dlhý dokument. Ku kombinácii slovenských a anglických pojmov nás prinútila skutočnosť, že viaceré z nich nemajú vhodné slovenské ekvivalenty. Číslovanie strán v časti riadenie nebolo použité úmyselne kvôli nízkemu počtu strán, ale priznávame, že by zrejme bolo vhodné ho aj napriek tomu použiť.

2. Obsahová stránka

Posudok konštatuje delenie dokumentu na časť pre riadenie a časť pre riešenie. Autori posudku nám vytýkali neprítomnosť slovníka pojmov. V našom prípade sa nám však nezdal potrebný, viac sme sa zamerali na vysvetlenie používaných skratiek.

Súhlasíme s tým, že by bolo vhodné uviesť na začiatku dokumentácie krátku charakteristiku programu NAZOU tak, aby aj nezainteresovaný čitateľ mal prehľad o širšom kontexte riešenia. Pretože sme však všetci mali možnosť získať aspoň základné informácie o tomto projekte, nezdala sa nám táto časť nevyhnutná. Posudok tvrdí, že v časti 1.3 sú tri scenáre vytrhnuté z kontextu a prevzaté z programu NAZOU, čo však nie je pravda. Všetky tri z nich sme vytvorili samostatne a nezávisle od programu NAZOU. Môže sa síce zdať, že sú prebraté, ale vyhľadanie, vloženie a modifikácia ponuky sú tak základné, že snáď každého napadnú ako prvé a preto napadli rovnako nás ako aj tvorcov scenárov v programe NAZOU.

Analýza problémovej oblasti podľa nás má zahŕňať aj uvedenú analýzu spôsobu zápisu konceptov z aplikačnej domény do ontológie, ktorá ešte nehovorí nič o tom ako bude samotný zápis údajov fyzicky riešený.

Ako posudok nepriamo priznáva, výber nástrojov zohľadňuje rozhodnutia vykonané v čase návrhu. Pre kvalifikované rozhodnutie je však potrebné vopred sa oboznámiť z dostupnými možnosťami, aby sme pri návrhu mohli tieto rozhodnutia vykonať. Práve z tohto dôvodu podľa nás kapitola Analýza riešenia má obsahovať analýzu dostupných nástrojov a technológií, ktoré využijeme pri implementácií riešenia, pričom analyzované technológie a nástroje boli vybraté ako reprezentanti daných oblastí v nadväznosti na ohraničenia programu NAZOU.

Posudok tvrdí, že prípady použitia boli prebraté z programu NAZOU bez uvedenia kontextu. Opäť si dovolíme nesúhlasiť. Pri tvorbe prípadov použitia sme vychádzali z programu NAZOU, avšak viaceré sme doplnili, resp. rozšírili. Navyše sme ku každému z nich pridali identifikátory a priority. Opäť nemožno predpokladať, že na rovnaký portál vymyslíme odlišné prípady použitia.

Architektúra v hrubom návrhu nie je prebratá s chybami, ale prebratá a úmyselne pozmenená. Slúži aj k tomu, aby čitateľovi ozrejmila kontext celého systému a súvislosti

medzi portálom pracovných príležitostí a programom NAZOU. Práve absenciu opisu kontextu systému nám posudok viacerých miestach vyčíta, avšak tam kde je uvedený nám vyčíta, že je prebratý. Súhlasíme s tým, že by bolo vhodné vyznačiť tú časť, ktorá priamo súvisí s portálom.

Analýza nástrojov podľa nás nepatrí do časti hrubého návrhu, kam patrí len jej vyhodnotenie a uvedenie nástrojov, ktoré chceme použiť pri tvorbe portálu. Súhlasíme s tým, že by bolo vhodné sa zaoberať aj integračným testovaním s nástrojmi programu NAZOU, avšak tieto sami nie sú v súčasnosti plne implementované a následne integrovateľné. Navyše hlavné prepojenie portálu je riešené pomocou ontologického úložiska, ktoré je rovnaké ako v programe NAZOU. Integrácia portálu do programu NAZOU nie je primárnym cieľom tímového projektu, aj keď sa snažíme sledovaním požiadaviek štátneho programu jeho integrátorom uľahčiť budúcu prácu.

3. Zhodnotenie

Posudok obsahuje viaceré objektívne i podnetné skutočnosti s ktorými súhlasíme. Musíme však rovnako skonštatovať, že obsahuje aj dosť vyjadrení s ktorými nesúhlasíme ako napr. umiestnenie kapitoly Analýza nástrojov a tvrdenia, že sme mnohé veci prebrali „s chybami“ z programu NAZOU. Nazdávame sa, že nám posudok poskytol dôležitý alternatívny pohľad na riešený projekt.

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

ANALÝZA TEXTOV PRACOVNÝCH PONÚK Z PROSTREDIA WEBU

POSUDOK DOKUMENTÁCIE TÍMU Č. 10

Tím č. 8: Hachiban
Vedúci tímu: tvarozek@gmx.net
Mailový alias: tp@atrip.sk
Predmet: Tvorba softvérového systému v tíme
Pedag. vedúci: Ing. Roman Filkorn
Ak. rok: 2005/2006

Členovia tímu:
Bc. Michal Barla
Bc. Peter Bartalos
Bc. Ján Porubský
Bc. Peter Sivák
Bc. Kristián Szobi
Bc. Michal Tvarožek

Úvod

Predkladaný dokument obsahuje posudok projektovej dokumentácie k softvérovému systému vytvorenému v rámci predmetu Tvorba softvérového systému v tíme v akademickom roku 2005/2006. Posudok sa vzťahuje na dokumentáciu odovzdanú tímom č. 10 v prvom kontrolnom bode. Zadaním bolo vytvoriť predprogramovaný obaľovač a demonštrovať jeho funkčnosť na príklade domény pracovných ponúk.

Dokument sa skladá zo štyroch kapitol. Prvá kapitola opisuje pohľad na dokumentáciu ako celok. Druhá kapitola sa zameriava na prvú časť dokumentácie – opis inžinierskeho diela, tretia kapitola je opisuje časť projektovej dokumentácie venovanú riadeniu. Štvrtá kapitola obsahuje zhodnotenie predloženej dokumentácie.

1. Celkový pohľad na dokument

Predložená projektová dokumentácia je formálne rozdelená na časť opisujúcu prácu na softvérovom systéme a na časť opisujúcu riadenie projektu. Prvá časť, opisujúca softvérový systém, obsahuje analýzu existujúcich nástrojov na tvorbu obaľovačov a špecifikáciu systému s opisom kontextu a funkcionality systému ako aj s opisom spracúvaných údajov.

Hrubý návrh riešenia obsahuje opis architektúry systému, opis vnútornej štruktúry obaľovača a opis navrhnutého jazyka na opis obaľovača. Dokument sa ďalej venuje nástrojom na implementáciu a vývoj obaľovača a opisu pilotnej aplikácie, ktorú tím bude vyvíjať v najbližšej dobe. Autori opisujú tri možné alternatívy ďalšieho postupu v podobe rôznych prototypov obaľovača a uvádzajú dôvody, ktoré ich viedli k výberu jednej z nich.

Druhá časť dokumentu opisuje riadenie projektu. Obsahuje ponuku vypracovanú na tému Portál pracovných príležitostí, zápisy zo stretnutí tímu, plán projektu na najbližšie obdobie a informácie o organizácii tímu a rozdelení práce medzi jednotlivých riešiteľov.

2. Časť I – Softvérový systém

Časť softvérový systém obsahuje analýzu problémovej oblasti, špecifikáciu požiadaviek a hrubý návrh riešenia. Autori analyzujú proces získavania údajov z webových stránok pomocou obaľovačov. Nástroje RoboSuite 5.5 SR2 a Lixto Visual Wrapper, ako aj dva pomocné nástroje WebVCR a PiggyBank sú veľmi podrobne opísané. Bolo by však vhodné uviesť aj krátke zhrnutie najdôležitejších vlastností jednotlivých nástrojov spolu s ich výhodami a nevýhodami. Celkovo analýza nástrojov obsahuje množstvo informácií, ktoré sa budú dať veľmi dobre využiť v ďalšom riešení projektu. Chýbala nám však analýza samotnej problémovej oblasti automatizovaného získavania informácií z webu, v ktorej by autori identifikovali existujúce problémy a nové prístupy k tvorbe obaľovačov.

Špecifikácia požiadaviek je rozdelená na viacero častí, pričom okrem základných funkcií systému opisuje aj používateľov a kontext systému, údaje spracúvané systémom a technológie potrebné k jeho realizácii. Jednotlivé časti špecifikácie sú vypracované na rôznej úrovni podrobnosti. Opis funkcionality systému je prehľadne zapísaný vo forme tabuľky, pričom pri jednotlivých funkciách sú uvedené aj ich priority. Prípady použitia by bolo vhodné opísať podrobnejšie a formálnejšie a uviesť ich na začiatku kapitoly. Za zvláštnu pokladáme špecifikáciu požiadaviek na používateľa systému.

Autori miešajú špecifikáciu požiadaviek s analýzou problémovej oblasti a nástrojov. Opis cieľového prostredia, opis technológií a prostriedkov realizácie a všeobecný opis údajov spracúvaných obaľovačom by bolo vhodné uviesť skôr v analýze problémovej oblasti, resp.

analýze nástrojov. V texte sa autori na viacerých miestach odvolávajú na „ciele a ohraničenia projektu“, ale tieto nie sú nikde vysvetlené.

Hrubý návrh ako celok pôsobí veľmi dobrým dojmom, má primeraný rozsah a hlavné časti rozpracúva na dobrej úrovni podrobnosti. Podkapitoly sú logicky usporiadané, niektoré sú však rozpracované oveľa podrobnejšie ako iné.

Návrh začína prehľadným obrázkom a opisom architektúry systému, ktorá vychádza zo vzoru Dátovody a filtre, aj keď to autori v texte neuvádzajú. Zaujímavou črtou navrhnutej architektúry je možnosť paralelného spracovania údajov v navrhnutých komponentoch. Vzhľadom na dôležitosť architektúry by však mala byť táto časť podrobnejšie rozpracovaná.

Autori dobre opisujú vnútornú štruktúru obaľovača a návrh jazyka na zápis obaľovača. V definícii jazyka sa mieša opis jazyka s postupnosťou krokov, ktoré obaľovač vykoná pri interpretácii jednotlivých príkazov jazyka, čo znižuje celkovú prehľadnosť. Jedným z kritérií na navrhovaný jazyk je procedurálny prístup, ale navrhnutý jazyk neumožňuje zápis procedúr.

Návrh obsahuje aj spôsob zápisu obaľovača vo forme XML súboru s definovanou štruktúrou a príklad zápisu obaľovača v navrhnutom jazyku, čo hodnotíme pozitívne. Po dokončení návrhu jazyka by bolo vhodné vytvoriť pre jeho zápis schému vo forme DTD alebo pomocou XSchema. Príklad programu obaľovača je uvedený len vo forme postupnosti krokov obaľovača pri spracovaní programu. Bolo by lepšie program uviesť v navrhnutom formáte zápisu vo forme XML súboru a celý príklad dať do prílohy.

V podkapitole o implementačných prostriedkoch autori podrobne opisujú rôzne existujúce vývojové prostredia a nástroje. Táto časť by mala byť uvedená v analýze, pričom v návrhu by boli uvedené už len vybrané nástroje so stručným zdôvodnením ich výberu. Veľmi podrobne sa tiež autori venujú pilotnej aplikácii – prototypu, ktorý budú vyvíjať v najbližšom období. Veľmi dobre rozoberajú viaceré možnosti postupu, avšak samotný text je príliš obsiahly, slabo štruktúrovaný a miestami pôsobí príliš neformálne, resp. netechnicky. Implementačným prostriedkom a prototypu je venovaný príliš veľký priestor na úkor samotného hrubého návrhu systému, ktorý následne pôsobí ako menej prepracovaný.

Námety a postrehy:

- Technológia AJAX a iné podobné technológie sa začínajú častejšie používať v praxi. Zvážte vplyv týchto technológií na funkcionalitu obaľovača.
- Bolo by dobré opísať vnútornú reprezentáciu autentizačných údajov z HTTP hlavičky a „cookies“. Prípadne aj hlbšie analyzovať tieto technológie.
- Ak na vnútornú reprezentáciu programu využívate objekty DOM, bolo by vhodné zvážiť aj iný spôsob reprezentácie, resp. analyzovať možné spôsoby vnútornej reprezentácie. Účelovo definované triedy pre objekty vnútornej reprezentácie by určite poskytli väčšie možnosti pre vyjadrenie atribútov.
- Ak niektoré obmedzenia, ciele a rozhodnutia súvisia so štátnym programom NAZOU, je vhodné to uviesť a priamo sa naň odvolať.

Pripomienky k dokumentácii:

- Analýzu je dobré lepšie štruktúrovať a viac vyzdvihnúť dôležité vlastnosti.
- Diagram prípadov použitia v analýze Lixto Visual Wrapper a ich opis by mal byť uvedený na začiatku kapitoly, pred opisom štrukturálnych prvkov obaľovača.
- Chýbajú odkazy na literatúru (t.j. nie na každú literatúru existuje odkaz).
- Domény webu a internetu sú dosť odlišné, pričom Internet nie je to isté ako internet.
- Nie je uvedené, čo je MainDocument, aj keď na viacerých miestach sú naň odkazy.
- Keď oddeľujete odseky medzerou a nie odsadením, nie je dobré používať 1,5 riadkovanie.
- Je vhodnejšie používať výraz „opis“ namiesto výrazu „popis“
- Nepoužívať hovorové výrazy, namiesto „s 3“ používať „s troma“.

- Použiť zaužívané preklady (well-formed – správne naformátovaný) a nemiešať angličtinu a slovenčinu („nie je attribute zadaný“).

3. Časť II – Riadenie

Časť riadenie obsahuje dokumentáciu k riadeniu projektu, vypracovanú v požadovanom rozsahu a kvalite. Plán projektu na nasledujúce obdobie je dostatočne podrobný. V dokumentácii sa však nenachádza žiadny plán na letný semester (ani rámcový). Autori uvádzajú rozdelenie úloh v tíme, ale podľa nás z neho nie je dostatočne jasné, kto čo robí. Nie je celkom jasné, aký význam má v tejto fáze riešenia projektu uvádzať zodpovednosť implementátor, resp. testovanie. Uvedené funkcie z hľadiska riadenia projektu nie sú vysvetlené a ani nezodpovedajú funkciám uvedeným v rámci stretnutia k riadeniu tímu.

Zápisy zo stretnutí tímu obsahujú požadované informácie a sú veľmi pekne spracované. Často sú však až príliš podrobné, obsahujú zbytočne veľa detailov a pôsobia ako „poznámky z prednášky“. Definovanie úloh na nasledujúce obdobie je príliš nekonkrétne, bez časových ohraničení. Úlohy sú ťažko merateľné (ak vôbec), nie sú jasne určené zodpovednosti členov tímu za vypracovanie jednotlivých úloh. Zápisy neobsahujú vyhodnotenie predchádzajúcich úloh, resp. je vyhodnotenie uvedené len v nepostačujúcom rozsahu.

Pripomienky k dokumentácii:

- Dokumentácia uvádza, že vypracovaná ponuka sa nachádza v kapitole 1, ale v skutočnosti je v prílohe A.
- Prečo plán projektu uvádza päť stupňov priority, keď je použitý len jeden?
- Predmet má názov Tvorba softvérového *systému* v tíme.
- Prečo je použité iné formátovanie ako v časti softvérový systém?
- Gramatické chyby a preklepy.

4. Zhodnotenie

Celkovo je projektová dokumentácia viac ako dobre vypracovaná. Rozsah a úroveň podrobnosti sú veľmi dobré, dokument sa venuje väčšine dôležitých aspektov vytváraného projektu v primeranom rozsahu. Analýza existujúcich nástrojov na tvorbu obaľovačov je miestami až príliš podrobná a ťažšie pochopiteľná. V analýze sa autori príliš zamerali na konkrétne nástroje a bližšie sa nevenovali analýze samotnej problémovej oblasti, identifikácii existujúcich problémov a nových prístupov k tvorbe obaľovačov. Tiež by bolo dobré v úvode dokumentu uviesť stručný celkový prehľad systému, jeho cieľov a účelu a hlavných funkcií s niekoľkými typickými scenármi použitia.

Špecifikácia požiadaviek obsahuje opis kontextu systému, opis prípadov použitia a scenárov použitia, ale po obsahovej stránke by mala byť prepracovanejšia a najmä viac konkrétna, formálna a procesne zameraná. Na viacerých miestach špecifikácia pôsobila skôr ako analýza, resp. opis problémovej oblasti obaľovačov.

Jednotlivé časti hrubého návrhu riešenia sú rozpracované podrobnejšie, iné menej podrobne. Uvedený návrh architektúry systému a štruktúry obaľovača je možno až príliš všeobecný a bolo by dobré ho dopracovať. Definícia jazyka na opis obaľovača je dostatočne podrobná, aj keď značne neprehľadná, najmä kvôli zmiešaniu opisu jazyka s opisom funkcionality obaľovača.

V hrubom návrhu autori podrobne opisujú nástroje a technológie na implementáciu a vývoj obaľovača a uvádzajú dôvody, prečo si ich vybrali. Táto podkapitola je však v zásade analýzou týchto nástrojov a ich najdôležitejších vlastností a teda nepatrí do návrhu systému a

mala by byť v inej kapitole. V časti návrhu by malo byť len zhodnotenie vykonanej analýzy a zoznam vybratých nástrojov.

Časť dokumentu opisujúca riadenie projektu obsahuje všetky potrebné časti. Plán projektu je podrobne vypracovaný, ale neobsahuje žiadny rámcový plán na letný semester. Zápisy zo stretnutí majú jednotnú podobu, avšak často sú zbytočne detailné, pričom úlohy sú často zadané príliš nepresne bez konkrétnych časových ohraničení.

Po formálnej stránke je dokument vypracovaný na viac ako dobrej úrovni. Formátovanie dokumentu je veľmi dobré a dostatočne čitateľné, obrázky a tabuľky sú prehľadné. Častým problémom dokumentu je však neprehľadnosť spôsobená nedostatočnou štruktúrovanosťou textu. Dokument obsahuje veľa dlhých viet, s komplikovanými vetnými konštrukciami, ktoré pôsobia mätúco a je ich potrebné čítať viackrát. Problémy so štylistikou a formálnosťou textu sa premietli aj do zbytočného miešania slovenských a anglických slov a do nemalého počtu gramatických chýb a preklepov. Na mnohých miestach sa prejavuje nekonzistencia v štýle písania a detailoch formátovania.

Náš celkový dojem z posudzovanej dokumentácie je aj napriek uvedeným nedostatkom pozitívny. Z dokumentácie je cítiť nemalé množstvo práce na podrobnej analýze problematiky obaľovačov, nástrojov na ich tvorbu, ako aj na návrhu celého systému.

7. Manažment verzií, konfigurácií a zmien

Na manažment verzií sme použili nástroj Subversion a klienta Tortoise SVN. Štruktúra úložiska je nasledovná (dodržiava odporúčania autorov Subversion):

- Tags – označené revízie
- Branches – alternatívne vývojové vetvy
- Trunk – hlavný smer vývoja
- Doc – dokumentácia

8. Preberacie protokoly