



# Tímový projekt

## 3D grafická podpora vyhľadávania znalostí v dokumentoch

Dokumentácia k softvérovému dielu

Tím č. 9 - d9vina

Bc. Ján Chlpek  
Bc. Juraj Jakobovič  
Bc. Ivan Janovic  
Bc. Matej Kompánek  
Bc. Vladimír Polák  
Bc. Marek Takáč

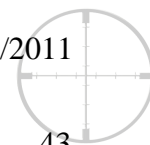
Vedúci pedagóg: Ing. Ivan Polášek, PhD.

d9vina@googlegroups.com  
Ak. rok : 2010/2011



## Obsah

<b>Zoznam tabuliek</b> .....	<b>ii</b>
0.1 Účel a rozsah dokumentu .....	iii
0.2 Zadanie .....	iii
0.3 Ciele a ohraničenia .....	iii
0.4 Prehľad dokumentu .....	iv
0.5 Použité skratky .....	v
<b>1 Analýza problému</b> .....	<b>1</b>
1.1 Analýza projektu Grafická podpora vyhľadávania znalostí v dokumentoch .....	1
1.2 Analýza projektu Vizualizácia softvérových artefaktov v 3D priestore .....	3
1.3 Analýza vizualizačných technológií.....	6
1.3.1 Processing.....	6
1.3.2 Java 3D / JoGL.....	7
1.3.3 C++ / OpenGL / Direct3D.....	7
1.3.4 OSG.....	7
1.3.5 Qt.....	8
<b>2 Špecifikácia riešenia</b> .....	<b>9</b>
2.1 Kontext systému .....	9
2.1.1 Nefunkcionálne požiadavky na systém.....	9
2.2 Špecifikácia funkcií a správanie systému.....	10
2.2.1 Prehľad prípadov použitia .....	10
2.3 Špecifikácia údajov v systéme .....	27
2.3.1 Dátový model systému .....	27
3.1 Návrh architektúry.....	30
3.1.1 Server .....	30
3.1.2 Klient.....	31
3.1.3 Komunikácia server – klient .....	33
<b>4 Prototyp</b> .....	<b>34</b>
4.1 Ciele prototypovania .....	34
4.2 2D klient.....	34
4.2.1 Odstránenie nepotrebných častí .....	34
4.2.2 Konfigurácia.....	35
4.2.3 Logovanie.....	35
4.2.4 Nová implementácia rozhrania KnowledgeBase .....	35
4.2.5 GraphML serializácia grafov .....	36
4.2.6 Server .....	37
4.3 Server .....	38
4.3.1 Zmeny oproti serveru minuloročného tímu.....	38
4.3.2 Vlastnosti serveru .....	38
4.3.3 Konfigurácia serveru .....	39
4.3.4 Komunikácia serveru s 2D klientom.....	39
4.3.5 Komunikácia serveru s databázou Postgre SQL .....	40
4.4 3D klient.....	40
4.4.1 Získanie grafu zo vzdialeného servera .....	40
4.4.2 Upravenie vytvárania návěstí vrcholov a hrán.....	41
4.5 Algoritmy na extrakciu kľúčových slov a získavanie metadát z dokumentov.....	41
4.5.1 Pridaná funkcionálnosť.....	41
4.5.2 Extrakcia kľúčových slov z dokumentov .....	42
4.5.3 Získavanie informácií z metadát .....	42
4.6 Databáza Postgre SQL .....	43



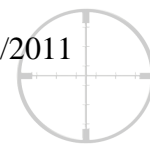
4.6.1 Vytvorenie novej databázy .....	43
4.6.2 Naplnenie databázy .....	43
4.6.4 Konfigurácia.....	44
4.6.5 Mapovacie súbory .....	44
4.6.6 Príklad funkcie v DtoDocument.xml .....	44
4.7 Zhodnotenie výsledkov prototypovania .....	45
<b>Príloha A .....</b>	<b>A-1</b>

## Zoznam obrázkov

Obr. 1. Diagram cieľov produktu .....	iv
Obr. 2. Návrh architektúry systému tímu Starwalkers .....	5
Obr. 3. Diagram prípadov použitia.....	10
Obr. 4. Diagram aktivít pre UC01 – Vyhľadávať dokumenty .....	14
Obr. 5. Diagram aktivít pre UC05 – Editovať dokumenty.....	17
Obr. 6. Diagram aktivít pre UC07 – Vytvoriť väzbu medzi dokumentmi .....	19
Obr. 7. Diagram aktivít pre UC14 – Zobrazit' dokumenty.....	23
Obr. 8. Stavový diagram pre dokument .....	25
Obr. 9. Stavový diagram pre väzbu .....	26
Obr. 10. Dátový model .....	27
Obr. 11. Server .....	30
Obr. 12. Klient.....	32
Obr. 13. Komunikácia klient-server .....	33
Obr. 14. Sekvenčný diagram komunikácie klient-server .....	33
Obr. 15. Nová implementácia rozhrania KnowledgeBase .....	36
Obr. 16. GraphML serializácia grafov .....	37
Obr. 17. Server .....	38
Obr. 18. Screenshot prototypov 2D a 3D klienta .....	46

## Zoznam tabuliek

Tab. 1. Opis UC01 – Vyhľadávať dokumenty .....	13
Tab. 2. Opis UC02 – Pridať dokumenty .....	15
Tab. 3. Opis UC03 – Vymazať virtuálny dokument .....	15
Tab. 4. Opis UC04 – Skryť dokumenty .....	16
Tab. 5. Opis UC05 – Editovať dokument .....	16
Tab. 6. Opis UC06 – Aktualizovať dokument .....	17
Tab. 7. Opis UC07 – Vytvoriť väzbu medzi dokumentmi.....	18
Tab. 8. Opis UC08 – Editovať väzbu medzi dokumentmi.....	19
Tab. 9. Opis UC09 – Odstrániť väzbu medzi dokumentmi .....	20
Tab. 10. Opis UC10 – Vygenerovať dokument .....	20
Tab. 11. Opis UC11 – Vytvoriť prepojovací (virtuálny) dokument .....	21
Tab. 12. Opis UC12 – Prepínať 2D/3D rozhranie.....	21
Tab. 13. Opis UC13 – Pohybovať sa v 3D priestore.....	22
Tab. 14. Opis UC14 – Zobrazit' dokumenty .....	22
Tab. 15. Opis UC15 – Prihlásiť do systému.....	23
Tab. 16. Opis UC16 – Zaregistrovať do systému.....	24
Tab. 17. Väzby medzi dokumentmi .....	43



## Úvod

### **0.1 Účel a rozsah dokumentu**

Dokument predstavuje finálnu verziu dokumentácie k softvérovému dielu vytvoreného na predmete Tímový projekt pre zimný semester. Okrem prehľadu dokumentu v tejto kapitole uvádzame zadanie projektu, jeho ciele a ohraničenia, pričom na konci kapitoly sa nachádza zoznam pojmov problémovej oblasti spolu s používanými skratkami. Ostatné kapitoly predstavujú analýzu a špecifikáciu riešenia, hrubý návrh riešenia a opis prototypu.

### **0.2 Zadanie**

Úlohou tímu bude vytvoriť modul na zobrazenie prepojenia dokumentov podľa informácií v nich v trojrozmernom grafe, ktorý by napomáhal tvorbe nových dokumentov (analytickej a technickej dokumentácie, manuálov, zdrojových kódov a pod.) pomocou zvolených dokumentov.

Hrany medzi jednotlivými dokumentmi budú mať rôzny vzor, farbu, hrúbku a podobne podľa množstva odkazov, spoločných kľúčových slov, autorov, čitateľov/používateľov, typov dokumentov a podobne.

Jednotlivé algoritmy vyhľadávania a prepojení bude možné vypínať, ako aj vyberať v grafe vhodné dokumenty, vyradovať nepotrebné cesty alebo vetvy. Na záver bude možné vybrať zaujímavé dokumenty, ktoré budú zdrojom pre tvorbu obsahu nového dokumentu.

Projekt môže nadväzovať na prácu tímu č. 12 (Dokumenty) a tímu č.20 (3DVizual) z minulého roku.

### **0.3 Ciele a ohraničenia**

Cieľom projektu je splniť požiadavky zadania projektu a vytvoriť funkčný systém pre 3D zobrazenie bázy dokumentov podľa zadaných požiadaviek používateľa, umožniť používateľovi pohyb v priestore a editáciu grafu vytvoreného väzbami medzi dokumentmi. Navyše bude možné vytvoriť nový dokument z vybraných viacerých dokumentov podľa preferencií používateľa.

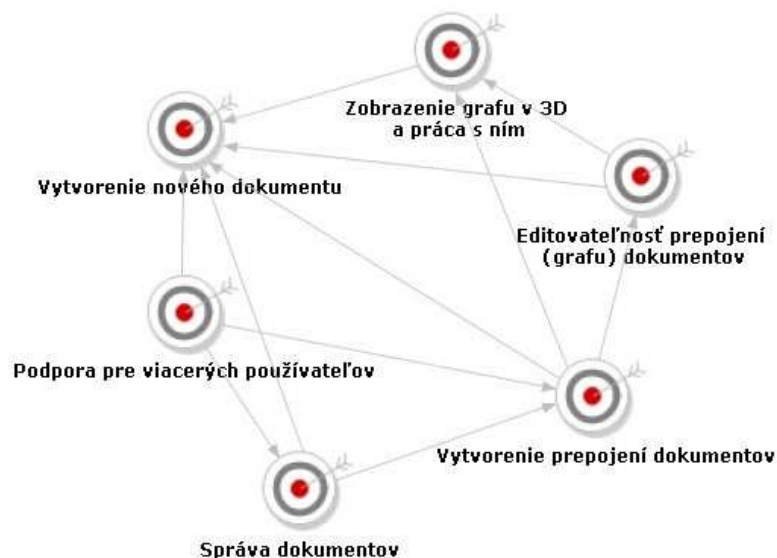


V projekte sa odporúča nadväzovať na práce tímov z minulých rokov. Po analýze týchto projektov sme dospeli k záveru, že nie je možné využiť ich v takej miere, ako sa pôvodne predpokladalo, takže tím sa zmenili aj ciele nášho projektu, ktorý mal byť akýmsi prepojením týchto predchádzajúcich prác spolu s pridaním novej funkcionality.

Keďže sa nemôžeme opierať o rozšírenie funkčnej implementácie, v novom systéme treba zabezpečiť:

- Správu dokumentov – vkladanie, uchovávanie, automatické indexovanie, editovanie
- Vytvorenie prepojení dokumentov
- Editovateľnosť prepojení (grafu) dokumentov
- Zobrazenie grafu v 3D a práca s ním
- Vytvorenie nového dokumentu
- Podporu pre viacerých používateľov

Na obrázku 1 znázorňujeme diagram cieľov nami vytváraného systému v súvislosti s plánovanou funkcionality. Jednotlivé šípky znamenajú závislosti cieľov, resp. nadväznosti dosiahnutia cieľov. To znamená, že aby sme dosiahli cieľ Vytvorenie prepojení dokumentov, je potrebné, aby sme dosiahli cieľ Správa dokumentov.



Obr. 1. Diagram cieľov produktu

## 0.4 Prehľad dokumentu

Úvod – Aktuálna kapitola

Kapitola 1 – Analýza problému



V tejto kapitole uvádzame analýzu tímových projektov, ktoré spracovávali problematiku prekrývajúcu sa s naším zadaním. Okrem toho analyzujeme vizualizačné technológie pre naše potreby.

### **Kapitola 2 - Špecifikácia riešenia**

Táto kapitola obsahuje špecifikovanie kontextu, funkcionality, dát a správania systému.

### **Kapitola 3 – Návrh riešenia**

Táto kapitola predstavuje architektonický návrh nášho riešenia.

### **Kapitola 4 – Prototyp**

V kapitole uvádzame ciele, opis a zhodnotenie prototypu.

## ***0.5 Použité skratky***

2D – dvojdimenzionálny

3D – trojdimenzionálny

API – Application Programming Interface

AWT – Abstract Window Toolkit

DAO – Data Access Objects

FIIT – Fakulta informatiky a informačných technológií

GUI – Graphical User Interface

HTTP – Hypertext Transfer Protocol

J2EE – Java 2 Enterprise Edition

JOGL – Java OpenGL

MIT – Massachusetts Institute of Technology

OSG – Open Scene Graph

RMI – Remote Method Invocation

SE – Standard Edition

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

STU – Slovenská technická univerzita

UC – Use Case

XML – eXtensible Markup Language



# 1 Analýza problému

Problémová oblasť je vopred definovaná a rozdelili sme ju na dve hlavné domény, ktoré predstavujú práce dvoch tímov, ktoré pracovali na projektoch:

- *Grafická podpora vyhľadávania znalostí v dokumentoch*
- *Vizualizácia softvérových artefaktov v 3D priestore*

Prvý projekt bol venovaný práci s dokumentmi, spracovaním dokumentov, vytváraním prepojení a grafov z dokumentov, pričom s grafmi a zobrazeniami sa dalo pracovať. Tento projekt v širšom meradle zodpovedá tomu, čomu venuje i náš tím, s tým rozdielom, že dokumenty boli zobrazené iba v 2D rozhraní, čomu bola prispôsobená celá implementácia.

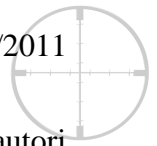
V druhom projekte sa študenti venovali vizualizácii softvéru a jednotlivých jeho artefaktov v 3D priestore pomocou grafových štruktúr, takže aj tento projekt má s naším projektom veľa spoločných prvkov.

Na nasledujúcich riadkoch predstavujeme analýzu oboch projektov a na konci kapitoly sa venujeme analýze vizualizačných technológií využiteľných v našom projekte, ich zhodnoteniu a zhodnoteniu vhodnosti použitia pre náš zámer.

## 1.1 Analýza projektu *Grafická podpora vyhľadávania znalostí v dokumentoch*

V rámci predmetu Tímový projekt I a II na našej fakulte v prechádzajúcom akademickom roku vznikol projekt s názvom *Grafická podpora vyhľadávania znalostí v dokumentoch*, ktorý bol vytvorený tímom č.12 (Šproty). Tento tím taktiež pracoval pod vedením Ing. Ivana Poláška, PhD., pričom nám bola poskytnutá možnosť nadviazať na tento projekt.

Našou hlavnou úlohou pri prípadnom nadväzovaní na tento projekt, by bolo hlavne doplniť funkcionality o zobrazenie vzťahov medzi dokumentmi v 3D priestore. Okrem toho by bolo vhodné vylepšiť indexáciu dokumentov spolu s vygenerovaním nového dokumentu skladajúceho sa z častí viacerých iných dokumentov, medzi ktorými boli zistené alebo definované vzťahy. Avšak ani po viactýždennej snahe sa nám nepodarilo zdrojové kódy, knižnice a ostatné konfiguračné súbory aplikácie tímu č.12 pretransformovať do takého stavu, aby bolo možné projekt spustiť bez chýb. Z toho dôvodu nebolo možné prakticky overiť správnosť funkcionality tohto projektu a preto sme pri tejto analýze vychádzali hlavne z dodanej dokumentácie a zo zdrojových kódov.



Tento projekt bol implementovaný v programovacom jazyku Java, pričom autori použili J2EE frameworky Spring a Hibernate. Okrem toho bola použitá databáza PostgreSQL a viacero Java knižníc, využitých pri vykresľovaní grafov a indexácii dokumentov. Autori podľa nášho názoru veľmi správne zvolili aplikácii architektúru klient-server. Toto rozhodnutie zdôvodňujú tým, že je potrebné oddeliť zobrazovaciu logiku od ostatných častí celého systému, tak aby bolo možné vyvíjať a aj nasadiť viacero nezávislých používateľských rozhraní. Na strane serveru bol použitý aplikačný server JBoss a strana klienta funguje ako applet. Tieto dva oddelené celky spolu komunikujú pomocou Java RMI (Remote Method Invocation).

Autori navrhli server tak, aby na jeho pozadí neustále bežali skryté vlákna, ktoré sa venujú analýze dokumentov a vyhľadávaniu väzieb medzi nimi. Taktiež na tejto strane vytvorili „kontajner algoritmov“, ktorý obsahuje viacero odlišných algoritmov, pričom každý z nich hľadá väzby medzi dokumentmi iným spôsobom (napr.: podľa autorov, kľúčových slov, vzájomných referencií atď.) To má výhodu v tom, že pri nasadzovaní nových algoritmov a novších verzií starých algoritmov by nebolo potrebné zasahovať do ostatných častí systému. Ako už bolo spomenuté klient beží ako applet a jeho úlohou je iba graficky znázorniť výsledky v podobe grafu dokumentov a vzťahov medzi nimi. Systém teda poskytuje tenkého klienta, ktorý je prístupný pomocou webového prehliadača. Klient sa skladá z ovládacieho panela, ktorý obsahuje formuláre pre vstupné dáta a časť pre samotnú vizualizáciu špecifických báz znalostí.

Tento projekt je teda zameraný na grafické zobrazenie väzieb medzi dokumentmi nachádzajúcimi sa v systéme. Aplikácia je však obmedzená iba na renderovanie v 2D priestore, na rozdiel od požiadaviek na náš projekt, ktorý má zvládať nielen vykresľovanie v 2D, ale aj v 3D. Túto funkcionality autori dosiahli pomocou open-source knižnice JGraph, ktorá je kompletne napísaná v jazyku Java a je vyvíjaná už niekoľko rokov, takže je pravdepodobne na vysokej úrovni. Za ďalšiu kľúčovú súčasť aplikácie, ktorá sa týka funkcionality, možno považovať indexáciu dokumentov a hľadanie vzťahov medzi nimi. Indexácia dokumentov je netriviálna úloha, ktorá je stále predmetom výskumu a preto aj existuje viacero dostupných knižníc, ktoré riešia tento problém. Autori najskôr používali Java knižnicu Lucene, no neskôr prešli na knižnicu Compass. Počas analýzy tohto projektu sme však zistili, že tento tím nemal indexáciu dokumentu dostatočne zvládnutú. To hlavne z toho dôvodu, že index bol vytvorený iba z metadát a nie z obsahu samotného dokumentu. Preto dokumenty bez definovaných metadát nie je možné použiť, čo značne znižuje kvalitu tejto aplikácie. Je síce možné okrem automatického definovania väzieb systémom pomocou skladu





algoritmov aj ručne poprepájať dokumenty, ktoré so sebou podľa používateľa súvisia. To však tento problém nerieši, avšak aspoň zlepšuje relevantnosť vzťahov medzi dokumentmi (používateľ totiž často môže poznať korelácie medzi dokumentmi, ktoré by aplikáciou nemuseli byť rozpoznané). Problémom tohto prístupu je však to, že jednou z kľúčových myšlienok projektu je automatizované vyhľadanie vzťahov medzi dokumentmi nielen z metadát, ale aj z ich obsahu.

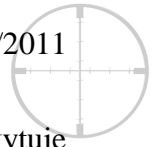
Je evidentné, že táto aplikácia môže pre náš projekt pôsobiť ako istý stavebný kameň. To berieme na zreteľ pri definovaní architektúry našej aplikácie, ale aj pri výbere technológií a knižníc na dosiahnutie identifikovanej funkcionality našej aplikácie. V našej práci avšak nebudeme pokračovať v zdrojových kódach tohto projektu, čo však neznamená, že tento kód nevyužijeme.

## **1.2 Analýza projektu *Vizualizácia softvérových artefaktov v 3D priestore***

V rámci predmetu Tímový projekt vznikol minulý rok pod vedením Ing. Petra Kapca zaujímavý projekt s názvom *Vizualizácia softvérových artefaktov v 3D priestore*. Tento projekt bol vytvorený tímom č. 20 (Starwalkers) a jeho cieľom bola vizualizácia softvéru a jednotlivých jeho artefaktov v 3D priestore pomocou grafových štruktúr. Táto téma sa v značnej miere prekrýva so zameraním nášho projektu a tým pádom je tento projekt vhodným kandidátom, z ktorého by sme mohli vychádzať hlavne v oblasti 3D vizualizácie grafov.

Ako bolo už vyššie spomenuté dôvodom, prečo by nám tento projekt mohol uľahčiť prácu, je najmä fakt, že tomuto tímu sa podarilo úspešne zanalyzovať oblasť zobrazovania grafov v 3D priestore a na základe zistených poznatkov implementovali takýto systém. I keď sa tento projekt okrem vizualizácie grafov venoval aj iným oblastiam, práve oblasti efektívneho vykreslenia a rozmiestnenia grafov v 3D priestore bolo venované najväčšie úsilie. Overiť, či je projekt funkčný sa doteraz podarilo len čiastočne, keďže sa ho nepodarilo oživiť úplne. Problémy sa týkajú najmä buildovania projektu a jednotlivých knižníc potrebných na vykresľovanie grafov. Nezáležiac na tom, či sa rozhodneme prevziať zdrojové kódy tohto tímu alebo si zvolíme možnosť implementovať túto časť projektu nanovo, faktom ostáva, že riešenia, ktoré tento tím priniesol sú dobrým základom pre náš ďalší postup.

Tento systém je napísaný v jazyku C++, pričom na vytvorenie používateľského rozhrania bol využitý framework Qt. Ďalšou z dôležitých súčastí systému je knižnica Open

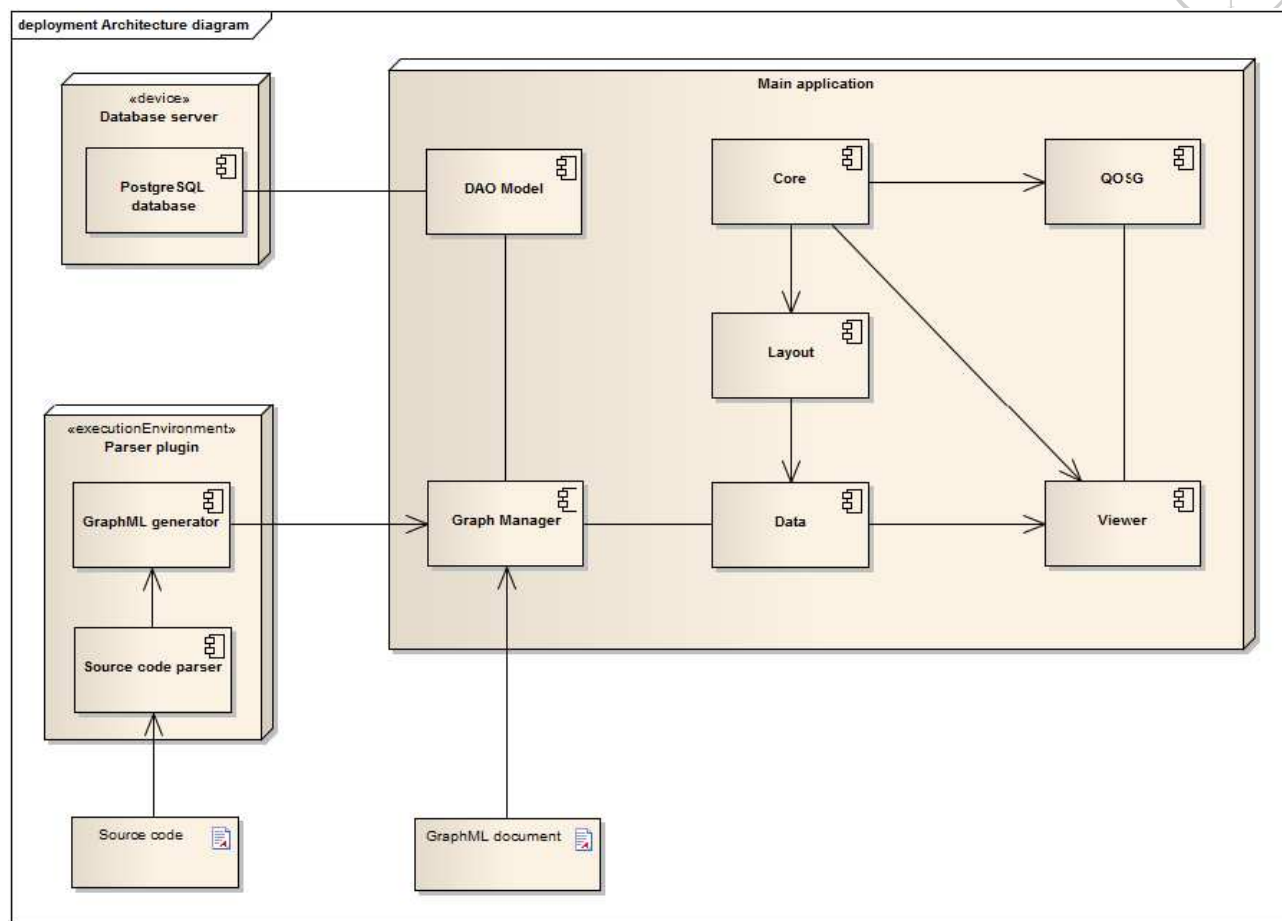


Scene Graph (OSG), ktorá sa používa na samotné vykresľovanie grafu, pričom poskytuje širokú funkčnosť a výkon. Na uchovávanie dát grafu je použitá databáza PostgreSQL. Na komunikáciu medzi hlavnou časťou systému a plugin-om na spracovanie zdrojového kódu bol využitý formát GraphML. Systém sa skladá z dvoch častí a to z hlavnej aplikácie a z parser plugin-u. Keďže tento plugin nie je pre náš projekt relevantný, opíšeme fungovanie len hlavnej časti systému.

Hlavná časť systému pozostáva z viacerých modulov, pričom každý má svoje špecifické funkcie. Dátová časť systému pozostáva z troch kľúčových modulov, pričom modul dátovej štruktúry grafy (Data) je najpodstatnejší, keďže je využívaný pri vizualizácii a layoutovaní. V tomto module sú reprezentované grafy prostredníctvom objektov. Zdrojom pre vytvorenie objektového modelu grafu je buď súbor, v ktorom je reprezentovaný graf vo formáte GraphML, alebo databáza PostgreSQL, s ktorou sa komunikuje prostredníctvom modulu mapovania objektov (DAO Model). Databáza teda slúži na uchovávanie jednotlivých grafov medzi jednotlivými behmi systému. Pri spustení systému modul manažér grafov (Manager) získa dostupné grafy z databázy a inicializuje z nich objektové modely.

Ďalším dôležitým modulom je modul rozmiestnenia uzlov v 3D priestore (Layout), ktorý pracuje s algoritmom Fruchterman-Reingold, ktorý upravil študentom FIIT STU Bc. Jakub Ukrop v rámci svojej bakalárskej práce. Tento algoritmus je vykonávaný v samostatnom vlákne, pričom toto vykonávanie prebieha v nekonečnej slučke. Rozhranie však umožňuje tento proces prerušiť a taktiež znova spustiť.

Nemenej dôležitým modulom je modul 3D vizualizácie (Viewer), ktorý zabezpečuje vizualizáciu grafu a taktiež interakciu používateľa s týmto grafom. Na tieto činnosti využíva knižnicu Open Scene Graph a tiež knižnicou Qt, ktorá zabezpečuje 2D prvky používateľského rozhrania. Jednotlivé moduly a ich prepojenia v rámci architektúry sú zobrazené na obrázku č.2.



Obr. 2. Návrh architektúry systému tímu Starwalkers

Znovupoužitie kódov tímu Starwalkers by nám značne uľahčilo prácu pri našom projekte, keďže ich výsledky v rámci vizualizácie boli na veľmi dobrej úrovni. Dosiahli to najmä tým, že ich grafický engine prešiel niekoľkými optimalizáciami, ktoré napomohli jeho výkonu najmä pri zobrazovaní rozsiahlejších grafov. Znovu implementovať tieto algoritmy by mohlo byť dosť náročné a to i časovo. Ďalším pozitívom je, že zdrojom pre vytvorenie nového grafu môže byť GraphML dokument. Toto vytvára priestor pre prepojenie systému pre vyhľadávanie znalostí s 3D vizualizáciou grafu. Negatíva by mohli vyplynúť najmä z preberania zdrojových kódov po iných autoroch a fakt, že doteraz sa nám tento projekt nepodarilo úplne spojzdať, tento názor len potvrdzuje.



### **1.3 Analýza vizualizačných technológií**

Pri implementácii vizualizácie grafov existuje viacero možností, čo sa týka vizualizačných knižníc a technológií, ako aj nástrojov pre tvorbu používateľského rozhrania. V ďalšej časti sú bližšie popísané a zhodnotené ich prínosy a nevýhody.

#### **1.3.1 Processing**

Processing je open source programovací jazyk a vývojárske prostredie na implementáciu problémov vizuálneho dizajnu. Tento projekt bol iniciovaný pred 10 rokmi a doteraz je vedený univerzitou MIT.

Jadro tejto knižnice je implementované v jazyku Java, vďaka čomu sú projekty vytvorené v tomto prostredí multiplatformové. Takisto každý kód napísaný v jazyku Processing je validný Java program.

Knižnica umožňuje vytváranie 2D aj 3D grafiky, pričom pracuje na úrovni vysokej abstrakcie, vďaka čomu je vývoj týchto aplikácií niekoľkonásobne rýchlejší oproti klasickým 3D nástrojom.

Komponenty vytvorené v tejto knižnici je jednoduché preniesť do klasických Java projektov, pretože každý takto vytvorený vizuálny komponent je použiteľný s GUI nástrojmi Java SE ako je Swing alebo AWT. Na realizáciu tohto prepojenia je potrebné využitie základných knižníc Processing. Zároveň teda treba dodať, že táto technológia nie je viazaná na svoje vývojárske prostredie.

Processing poskytuje dva spôsoby vykresľovania grafiky. Prvou z nich je možnosť natívneho Java vykresľovania. Výhodou tejto metódy je fakt, že takto napísaný program bude bežať na každej implementácii virtuálneho stroja Javy bez nutnosti inštalovania dodatočných knižníc. Takisto nevyžaduje prítomnosť grafickej karty. Nevýhodou je značné spomalenie procesu vykresľovania, čo sa týka zložitosti vykresľovanej scény alebo počtu objektov. Druhou možnosťou je vykresľovanie 3D scény prostredníctvom technológie JOGL (Java OpenGL), ktorá prináša značné urýchlenie vykresľovania. Nevýhodou je nutnosť dodatočných knižníc (JOGL + OpenGL). Oba spôsoby poskytujú štandardné schopnosti grafických rozhraní ako je napríklad vyhladzovanie alebo nasvietenie scény.

Hlavnou výhodou tejto technológie je jednoduchosť nasadenia, keďže implementované vizualizácie je možné priamo použiť v klasických Java rozhraniach alebo zobrazovať v internetových prehliadačoch prostredníctvom technológie Java Appletov bez nutnosti inštalácie.



Je nutné dodať, že knižnica nedokáže konkurovať natívnym implementáciám v nižších jazykoch (C++), takže nie je vhodná pre zobrazovanie a prácu s veľmi komplikovanými scénami, ktoré vyžadujú rýchlu odozvu pre pohodlnú prácu s rozhraním.

### 1.3.2 Java 3D / JoGL

Tieto dve knižnice predstavujú možnosti platformy Java na vykresľovanie 3D scény. Obe pracujú na základe natívnych volaní a využitia hardvérovej podpory vykresľovania, vďaka čomu sú efektívnejšie oproti predchádzajúcej knižnici.

V súvislosti s technológiou Java 3D je potrebné spomenúť bakalársku prácu študenta FIIT Bc. Jakuba Ukropa, ktorá venuje problematike vykresľovania rozsiahlych 3D grafov. Riešenie bolo implementované na platforme Java s využitím technológie Java 3D. Implementácia bola dostatočne efektívna na zobrazovanie grafov s počtom uzlov rádovo v tisícoch.

Výhodou knižnice Java 3D je poskytnutý objektový model, ktorý môže byť výhodou oproti klasickým procedurálnym rozhraniam grafických knižníc.

### 1.3.3 C++ / OpenGL / Direct3D

Knižnice OpenGL a Direct3D predstavujú rozhranie pre programovanie aplikácii, ktoré vytvárajú 2D a 3D zobrazenie s využitím hardvérovej podpory grafických kariet. Obe rozhrania pracujú v jazyku C / C++, pričom existujú nadstavby pre vyššie programovacie jazyky (ako spomínaný JOGL).

OpenGL predstavuje multiplatformovú špecifikáciu, ktorá zahŕňa aj open-source implementácie, zatiaľ čo Direct3D je súčasťou proprietárnych knižníc DirectX spoločnosti Microsoft, ktoré sú určené pre operačný systém Windows.

Obe tieto knižnice predstavujú najefektívnejší spôsob zobrazovania 3D scény. Napriek tomu zložitou implementácie presahujú možnosti tohto projektu.

### 1.3.4 OSG

Open Scene Graph (OSG) je open-source, multiplatformové programové rozhranie na vývoj grafických aplikácií. Je určené na implementáciu problémov vizuálnej simulácie, virtuálnej reality, modelovania a pod. Pracuje na základe rozhrania OpenGL a ponúka objektový model,



ktorý uľahčuje prácu s 3D zobrazením. Tento nástroj je podporovaný väčšinou operačných systémov.

Výhodou použitia knižnice OSG je už spomínaný objektový model, ktorý pracuje ako nadstavba nad OpenGL. Okrem toho pracuje s dátovou štruktúrou grafu scény, ktorý reprezentuje vizualizované objekty. Vďaka tomu odbreňuje programátora od nutnosti práce priamo s primitívnymi útvarmi 3D zobrazenia, následkom čoho je uľahčenie a zrýchlenie vývoja. Okrem toho podporuje optimalizáciu 3D zobrazenia na nízkej úrovni, čo sa týka vzdialených objektov, úrovne detailu alebo prekrývania jednotlivých objektov.

Táto knižnica je najbežnejšie používaným nástrojom pri vizualizácii rozsiahlych grafov. V rámci projektov FIIT môžeme spomenúť projekty tímu Starwalkers alebo bakalársku prácu Bc. Petra Kajana.

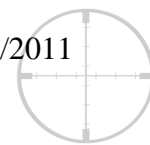
### 1.3.5 Qt

Qt je open-source knižnica na tvorbu okien a GUI pracujúca v jazyku C / C++. Je vyvíjaná spoločnosťou Nokia. Je vysoko oceňovaná hlavne odborníkmi na problematiku HCI (human computer interaction), keďže ponúka množstvo prvkov pre tvorbu grafického rozhrania. Okrem toho poskytuje programové rozhranie aj pre tvorbu klasických konzolových programov.

Hlavnou výhodou tejto knižnice sú jej bohaté možnosti pri tvorbe okien, efektívna objektová štruktúra a jej multiplatformovosť (okrem klasických operačných systémov existujú aj implementácie pre mobilné platformy ako Symbian alebo Windows Mobile).

Qt vo svojej distribúcii okrem vývojárskej sady obsahuje aj samostatné prostredie, ktoré umožňuje jednoduchý vývoj okien. Okrem toho poskytuje možnosť integrácie s prostredím Visual Studio 2008.

V oblasti tvorby GUI má táto knižnica minimálnu konkurenciu. Spomenúť môžeme knižnicu MFC, ktorá je však výrazne zastaraná s menšou možnosťou prispôsobenia a vysokou náročnosťou implementácie. Aj vďaka tomu je Qt ideálnou možnosťou tohto projektu pre tvorbu štandardných GUI prvkov.



## 2 Špecifikácia riešenia

V tejto kapitole uvádzame špecifikáciu riešenia, zahŕňajúcu kontext nášho systému a jeho súvislosť s problematikou, funkcie systému a prípady použitia, údaje v systéme s dátovým modelom a správanie systému vyjadrené niekoľkými diagramami.

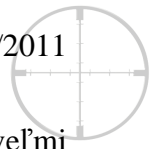
### 2.1 Kontext systému

Po analýze práce tímov č. 12 (Grafická podpora vyhľadávania znalostí v dokumentoch) a tímu č.20 (Vizualizácia softvérových artefaktov v 3D priestore) sme sa rozhodli pokračovať v ich práci a využiť maximum z toho, čo sa im podarilo dosiahnuť. Aj napriek nášmu rozhodnutiu začať implementáciu od začiatku, algoritmy, ktoré tieto dva tímy implementovali vieme využiť a prípadne migrovať do nami zvolených programovacích jazykov. Rovnako pri návrhu systému a špecifikovaní požiadaviek sme využili analýzy týchto dvoch tímov, čím sme ušetrili čas, ktorý sme mohli venovať oživovaniu ich projektov.

Nami navrhovaný systém bude umožňovať 3D zobrazenie vzťahov medzi dokumentmi. Vzťahy medzi dokumentmi budú určené na základe metadát, ale aj obsahu dokumentov. Používateľ si bude môcť samostatne, nezávisle na vygenerovaných väzbách vytvoriť vlastné väzby. Každá väzba bude mať isté vlastnosti, ktoré budú vizuálne odlišné (farba, hrúbka, a i.). Informácie o dokumentoch sa budú ukladať v báze znalostí na serverovej časti systému. Na strane klienta, bude prebiehať spracovanie informácií aj samotné zobrazovanie. Zobrazovanie a samotné jadro systému navrhujeme čo najviac oddelené, pre prípadné zmeny v zobrazení, napríklad v prípade nutnosti optimalizácie rýchlosti zobrazovania.

#### 2.1.1 Nefunkcionálne požiadavky na systém

Od systému sa očakáva, že po určitej dobe a vylepšeniach (napríklad ďalšími tímami) bude vhodný na využívanie v praxi. Mal by byť teda konkurencieschopný a cieľom je, že by sa v budúcnosti mohol stať súčasťou produktu Gratex Knowledge Office a napomáhal tvorbe nových dokumentov (analytickej a technickej dokumentácie, manuálov, zdrojových kódov a pod.) pomocou získaných znalostí. Toto predpokladá fakt, že síce v rámci projektu budeme pracovať rádovo s desiatkami dokumentov, ale v budúcnosti je možné, že systém bude



pracovať s tisíckami dokumentov, ktorých 3D zobrazenia a práca s nimi môže byť veľmi náročná a preto treba zvážiť technológie, ktoré budeme používať aj smerom do budúcnosti.

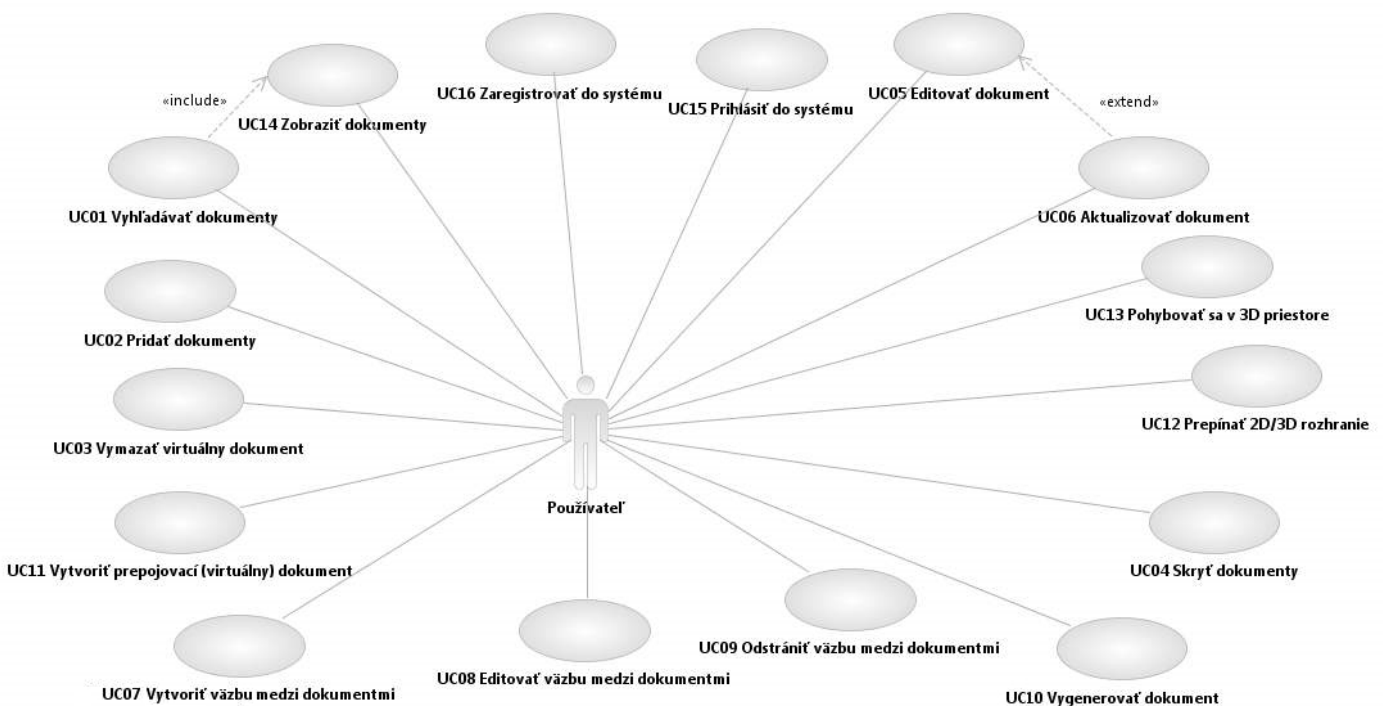
## 2.2 Špecifikácia funkcií a správania systému

Systém pre naše potreby vytvárame pre jeden typ používateľov, ktorí budú mať rovnaké práva. Je dôležité, aby bola vytvorená podpora pre viacerých používateľov, aby sme mohli riešiť záležitosti s históriou prezerania dokumentov. Týmto rozumieme odporúčanie podobných dokumentov na základe toho, aké dokumenty prezeral nejaký konkrétny používateľ v minulosti.

V tejto kapitole uvedieme prehľad prípadov použitia s popisom a následne pre každý prípad použitia uvedieme tabuľku s opisom a scenárom. Pre zaujímavé prípady použitia uvádzame aj diagramy aktivít. Okrem toho pre kľúčové entity uvádzame stavové diagramy.

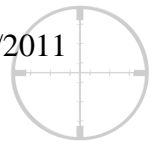
### 2.2.1 Prehľad prípadov použitia

Na obrázku č. 3 sú v diagrame prípadov použitia znázornené identifikované prípady použitia pre systém. Pre každý prípad použitia v jeho podrobnom opise uvádzame aj prioritu, teda dôležitosť pre splnenie tých hlavných vytýčených cieľov pre úspešnosť nášho projektu.



Obr. 3. Diagram prípadov použitia.





Prípady použitia pre navrhovaný systém sú teda nasledovné:

#### **UC01 – Vyhľadávať dokumenty**

Systém poskytne používateľovi v grafickom rozhraní vyplniť formuláre a podľa zadaných informácií vyhľadá príslušné dokumenty. V tabuľke č. 1 je prípad použitia UC01 podrobne opísaný. Obrázok č. 4 znázorňuje diagram aktivít pre daný prípad použitia.

#### **UC02 – Pridať dokumenty**

Systém poskytne používateľovi možnosť pridať jeden alebo viac dokumentov. V tabuľke č. 2 je prípad použitia UC02 podrobne opísaný.

#### **UC03 – Vymazať virtuálny dokument**

Systém umožní používateľovi vymazať existujúci virtuálny dokument zo systému. V tabuľke č. 3 je prípad použitia UC03 podrobne opísaný.

#### **UC04 – Skryť dokumenty**

Systém umožní používateľovi v grafe dokumentov skryť jeden alebo viacero uzlov (t.j. dokumentov). V tabuľke č. 4 je prípad použitia UC04 podrobne opísaný.

#### **UC05 – Editovať dokument**

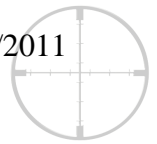
Používateľ pomocou systému môže upraviť metadáta dokumentu, ako napríklad názov dokumentu, meno autora, kľúčové slová atď., alebo priamo jeho obsah. V tabuľke č. 5 je prípad použitia UC05 podrobne opísaný. Obrázok č. 6 znázorňuje diagram aktivít pre daný prípad použitia.

#### **UC06 – Aktualizovať dokument**

Používateľ môže nahrať do systému novú verziu dokumentu. Systém novú verziu indexuje, starú verziu odstráni a uchová novú verziu namiesto pôvodného dokumentu. V tabuľke č. 6 je prípad použitia UC06 podrobne opísaný.

#### **UC07 – Vytvoriť väzby medzi dokumentmi**

Pomocou systému používateľ vytvorí prepojenie medzi dokumentmi, ktoré spolu súvisia, ale v systéme zatiaľ prepojené nie sú. V tabuľke č. 7 je prípad použitia UC07 podrobne opísaný. Obrázok č. 6 znázorňuje diagram aktivít pre daný prípad použitia.

**UC08 – Editovať väzbu medzi dokumentmi**

Používateľ môže pomocou systému upraviť atribúty niektorej väzby na základe nových zistení, prípadne iných okolností. V tabuľke č. 8 je prípad použitia UC08 podrobne opísaný.

**UC09 – Odstrániť väzbu medzi dokumentmi**

Ak používateľ zistí, že niektorá väzba nevyjadruje skutočnú súvislosť medzi dokumentmi, pomocou systému ju môže odstrániť. Takáto situácia môže nastať napríklad preto, že systém niektoré väzby medzi dokumentmi vytvoril automaticky, ale tieto neboli vytvorené správne. V tabuľke č. 9 je prípad použitia UC09 podrobne opísaný.

**UC10 – Vygenerovať dokument**

Systém vygeneruje nový dokument na základe zvolených dokumentov. V tabuľke č. 10 je prípad použitia UC10 podrobne opísaný.

**UC11 – Vytvoriť prepojavací (virtuálny) dokument**

Systém vytvorí dokument, ktorý slúži na umelé prepojenie jednotlivých dokumentov. Tento dokument neobsahuje samotné znalosti, ale slúži na združenie dokumentov obsahujúcich znalosti o určitej doméne. V tabuľke č. 11 je prípad použitia UC11 podrobne opísaný.

**UC12 – Prepínať 2D/3D rozhranie**

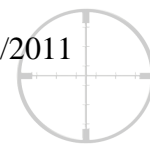
Systém používateľovi umožní prepínanie medzi zobrazením 2D a 3D. V tabuľke č. 12 je prípad použitia UC12 podrobne opísaný.

**UC13 – Pohybovať sa v 3D priestore**

Systém používateľovi umožní pohybovanie grafom v 3D priestore pozostávajúce z približovania a otáčania. V tabuľke č. 13 je prípad použitia UC13 podrobne opísaný.

**UC14 – Zobrazit' dokumenty**

Systém umožní používateľovi zvoliť dokument, ktorý chce zobrazit' v grafe spolu s jeho vzťahmi k ostatným dokumentom a zobrazí ho. V tabuľke č. 14 je prípad použitia UC14 podrobne opísaný. Obrázok č. 7 znázorňuje diagram aktivít pre daný prípad použitia.



### UC15 – Prihlásiť do systému

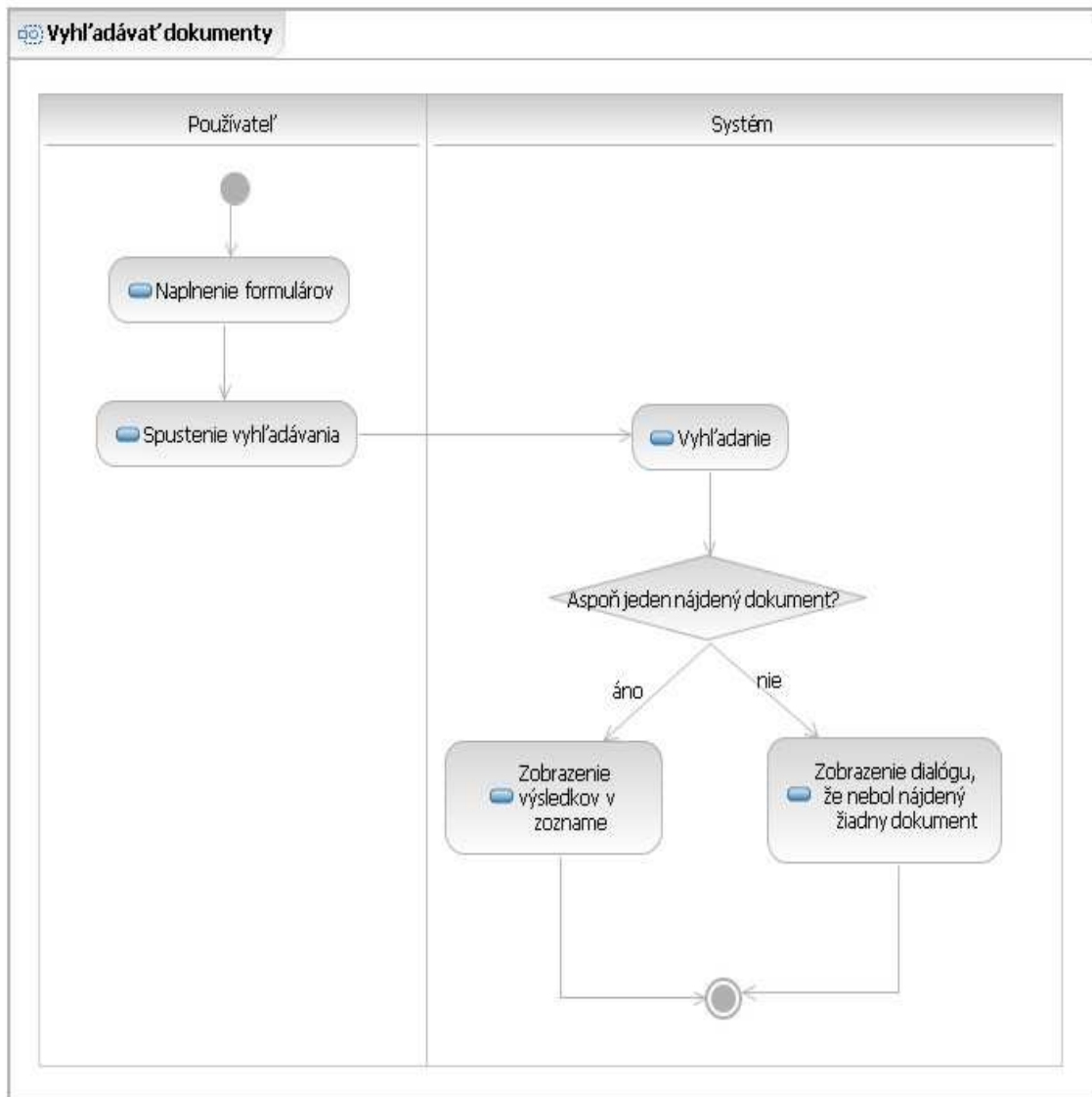
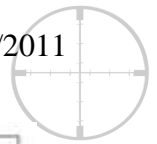
Používateľ, aby mohol plnohodnotne využívať funkcionality programu, sa pomocou systému môže prihlásiť do systému. V tabuľke č. 15 je prípad použitia UC15 podrobne opísaný.

### UC16 – Zaregistrovať do systému

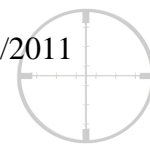
Používateľ sa pomocou systému môže zaregistrovať do systému. V tabuľke č. 16 je prípad použitia UC16 podrobne opísaný.

**Tab. 1.** Opis UC01 – Vyhľadávať dokumenty

<b>Identifikátor</b>	UC01	
<b>Názov</b>	Vyhľadávať dokumenty	
<b>Opis</b>	Používateľ v grafickom rozhraní vyplní formuláre a nechá systém podľa týchto informácií vyhľadať príslušné dokumenty.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému.	
<b>Výstup. podm.</b>	Používateľ môže vidieť zoznam dokumentov, ktoré spĺňajú kritéria, ktoré navolil.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ vyplní formuláre pre vyhľadanie dokumentov (autor, názov, kľúčové slová atď.).
	2	Používateľ dá vyhľadať dokumenty.
	3	Systém zobrazí zoznam dokumentov, ktoré vyhovujú vyhľadávacím informáciám.
	4	Prípad použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	3.a	Systém nenájde žiadne vyhovujúce dokumenty a zobrazí upozornenie pre používateľa.
	4.a	Prípad použitia končí.
<b>Rozširujúce body</b>		



**Obr. 4.** Diagram aktivít pre UC01 – Vyhľadávať dokumenty

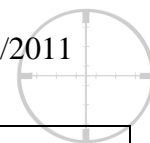


Tab. 2. Opis UC02 – Pridať dokumenty

<b>Identifikátor</b>	UC02	
<b>Názov</b>	Pridať dokumenty	
<b>Opis</b>	Používateľ zvolí dokument/y, ktoré chce pridať do systému.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému.	
<b>Výstup. podm.</b>	Zvolené dokumenty sú pridané v systéme.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí že chce pridať nové dokumenty do systému.
	2	Používateľ zvolí dokumenty a potvrdí ich vloženie do systému.
	3	Systém vloží dokumenty do databázy.
	4	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	3.a	Systém zistil, že aspoň jeden súbor nie je v podporovanom formáte a zobrazí správu pre používateľa .
	4.a	Pokračuje sa krokom 1 primárneho behu.
<b>Rozširujúce body</b>		

Tab. 3. Opis UC03 – Vymazať virtuálny dokument

<b>Identifikátor</b>	UC03	
<b>Názov</b>	Vymazať virtuálny dokument	
<b>Opis</b>	Používateľ chce vymazať existujúci virtuálny dokument zo systému.	
<b>Priorita</b>	Stredná	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Dokument, ktorý má byť vymazaný je v systéme a je zobrazený.	
<b>Výstup. podm.</b>	Systém vymaže zvolený virtuálny dokument.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí zobrazený virtuálny dokument, ktorý chce vymazať.
	2	Používateľ zvolí možnosť vymazania tohto dokumentu.
	3	Systém vymaže tento uzol spolu so všetkými väzbami, ktoré do tohto uzla smerujú alebo vychádzajú.
	4	Systém vymaže tento virtuálny dokument.
	5	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1.a	Používateľ zvolí viacero virtuálnych dokumentov, ktoré chce vymazať.
	2.a	Používateľ zvolí možnosť vymazania všetkých označených dokumentov.
	3.a	Systém vymaže všetky tieto uzly spolu so všetkými väzbami, ktoré do týchto uzlov smerujú alebo z nich vychádzajú.



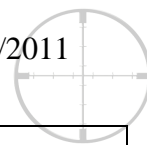
	4.a	System vymaže tento virtuálny dokument.
	5.a	Prípád použitia končí.
<b>Rozširujúce body</b>		

Tab. 4. Opis UC04 – Skryť dokumenty

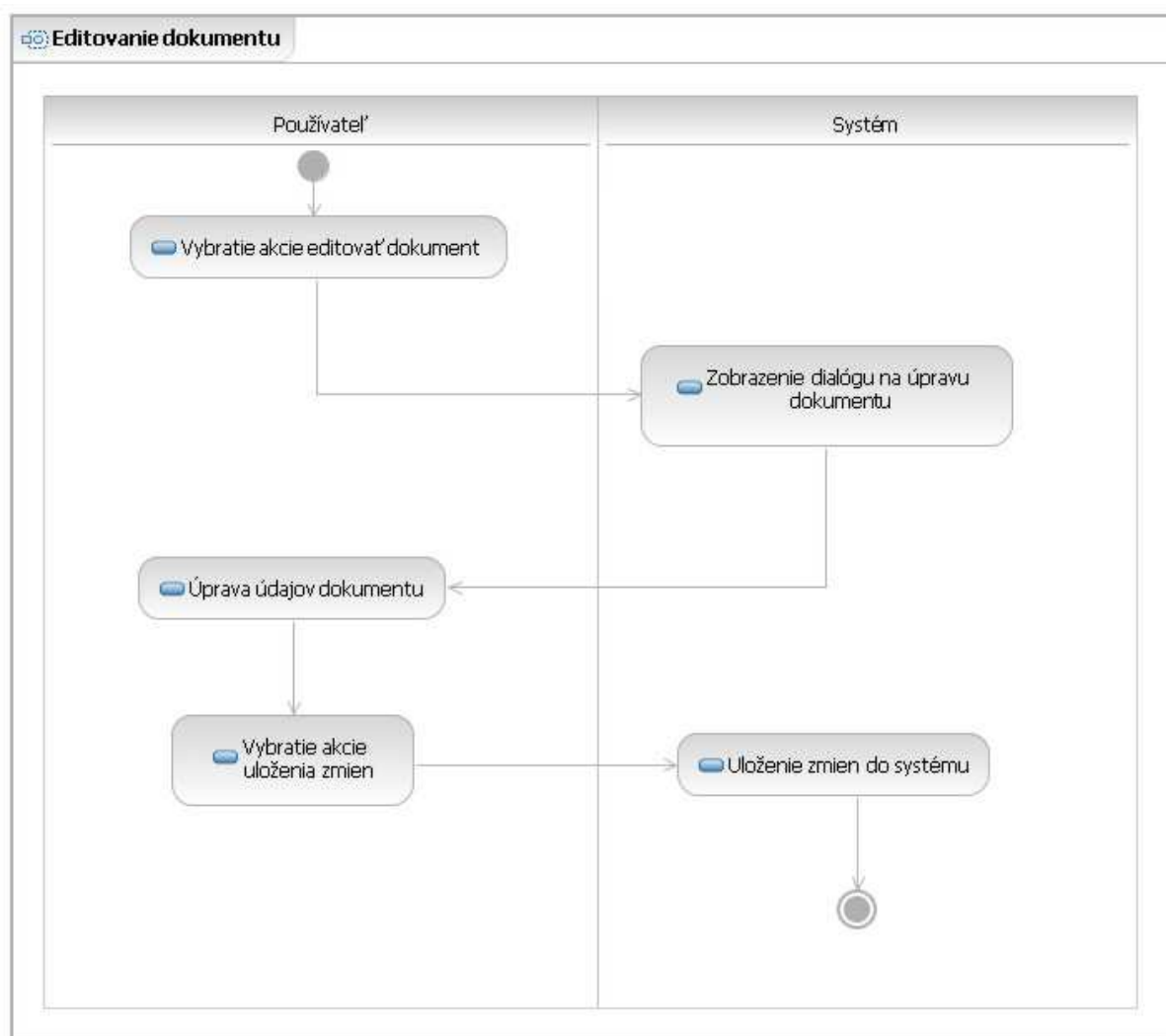
<b>Identifikátor</b>	UC04	
<b>Názov</b>	Skryť dokumenty	
<b>Opis</b>	Používateľ chce v grafe dokumentov skryť jeden alebo viacero uzlov (t.j. dokumentov).	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Dokument, ktorý má byť skrytý je v systéme a je zobrazený.	
<b>Výstup. podm.</b>	Označené dokumenty spolu s väzbami, ktoré z týchto uzlov vychádzali alebo vchádzali nie sú zobrazené. Ak ostali dokumenty, ktoré nemajú žiadne väzby, tiež sú skryté.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí zobrazený dokument, ktorý chce skryť.
	2	Používateľ zvolí možnosť skrytia tohto dokumentu.
	3	System nezobrazí tento uzol spolu so všetkými väzbami, ktoré do tohto uzla smerujú alebo vychádzajú. Ak ostali dokumenty, ktoré nemajú žiadne väzby, tiež ich systém nezobrazí.
	4	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1.a	Používateľ zvolí viacero dokumentov, ktoré chce skryť .
	2.a	Používateľ zvolí možnosť skrytia všetkých označených dokumentov.
	3.a	System nezobrazí tieto uzly spolu so všetkými väzbami, ktoré do týchto uzlov smerujú alebo z nich vychádzajú. Ak ostali dokumenty, ktoré nemajú žiadne väzby, tiež ich systém nezobrazí.
	4.a	Prípád použitia končí.
<b>Rozširujúce body</b>		

Tab. 5. Opis UC05 – Editovať dokument

<b>Identifikátor</b>	UC05	
<b>Názov</b>	Editovať dokument	
<b>Opis</b>	Používateľ upraví metadáta dokumentu, ako napríklad názov dokumentu, meno autora, kľúčové slová atď., alebo priamo jeho obsah.	
<b>Priorita</b>	Stredná	
<b>Vstup. podm.</b>	Používateľ je prihlásený v systéme. Dokument, ktorý chce používateľ upraviť je viditeľný v aktuálnom zobrazení grafu.	
<b>Výstup. podm.</b>	Dokument je upravený a uložený v systéme.	



<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí akciu editovať dokument.
	2	System zobrazí dialóg na úpravu dokumentu.
	3	Používateľ upraví údaje popisujúce dokument (jeho názov, meno autora, atď.), prípadne obsah dokumentu.
	4	Používateľ v dialógu zvolí akciu uložiť.
	5	System uloží zmeny, ktoré zadal používateľ.
	6	Prípad použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
<b>Rozširujúce body</b>	3.	Možné rozšírenie o nahratie novej verzie dokumentu.



**Obr. 5.** Diagram aktivít pre UC05 – Editovať dokumenty

**Tab. 6.** Opis UC06 – Aktualizovať dokument

<b>Identifikátor</b>	UC06
<b>Názov</b>	Aktualizovať dokument

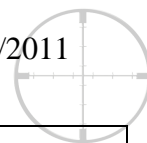


<b>Opis</b>	Používateľ nahrá do systému novú verziu dokumentu. Systém novú verziu zaindexuje namiesto pôvodného dokumentu a starú verziu odstráni.	
<b>Priorita</b>	Stredná	
<b>Vstup. podm.</b>	Tento prípad použitia rozširuje UC05 Editovať dokument v bode 3.	
<b>Výstup. podm.</b>	Súbor dokumentu je nahradený novšou verziou a je korektne zindexovaný v systéme.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí akciu pre nahratie novšej verzie dokumentu.
	2	Systém používateľovi ponúkne dialóg na výber súboru.
	3	Používateľ vyberie a označí súbor určený na nahratie do systému.
	4	Systém zaindexuje nový súbor a starý odstráni. Novým súborom taktiež prepíše jeho neaktuálnu verziu v databáze.
	5	Systém zaktualizuje väzby pre novú verziu dokumentu.
	6	Prípad použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
<b>Rozširujúce body</b>		

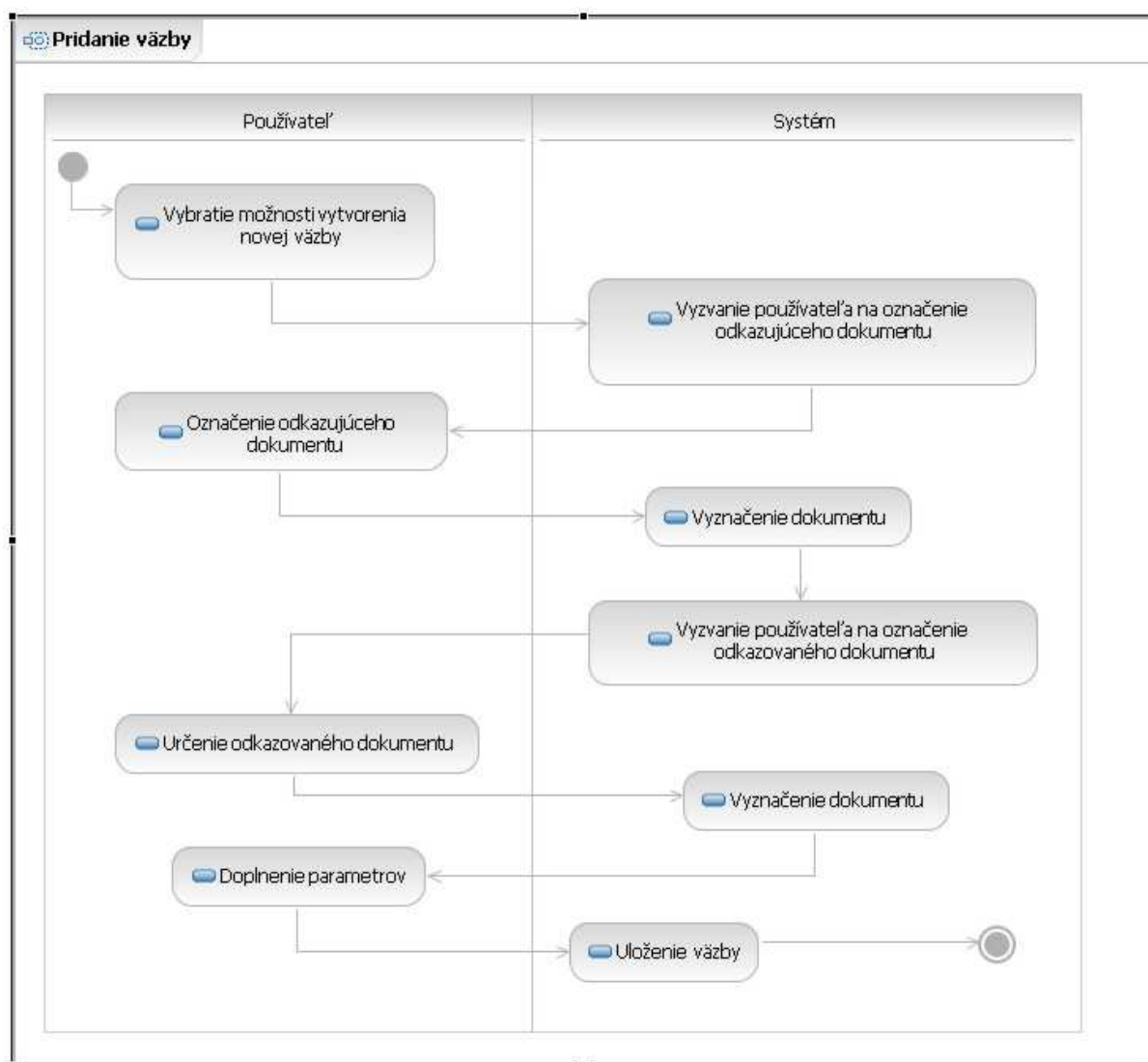
Tab. 7. Opis UC07 – Vytvoriť väzbu medzi dokumentmi

<b>Identifikátor</b>	UC07	
<b>Názov</b>	Vytvoriť väzbu medzi dokumentmi	
<b>Opis</b>	Používateľ vytvorí prepojenie medzi dokumentmi, ktoré spolu súvisia, ale v systéme zatiaľ prepojené nie sú.	
<b>Priorita</b>	Stredná až vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený v systéme. Obidva dokumenty, medzi ktorými má byť vytvorená väzba sú zobrazené.	
<b>Výstup. podm.</b>	Je vytvorená a uložená nová väzba medzi dokumentmi.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí možnosť vytvoriť novú väzbu medzi dokumentmi.
	2	Systém používateľa vyzve, aby vyznačil odkazujúci dokument.
	3	Používateľ vyberie odkazujúci dokument spomedzi zobrazených dokumentov.
	4	Systém na ploche vyznačí odkazujúci dokument a vyzve používateľa na označenie odkazovaného dokumentu.
	5	Používateľ vyberie odkazovaný dokument spomedzi zobrazených dokumentov.
	6	Systém na ploche vyznačí odkazovaný dokument a vyzve používateľa na doplnenie parametrov väzby (typ, sila, obojsmernosť, atď.).
	7	Používateľ zadá ostatné parametre väzby.
	8	Systém novú väzbu vytvorí a uloží.
	9	Prípad použitia končí.
<b>Alternatívna</b>	<b>Krok</b>	<b>Činnosť</b>





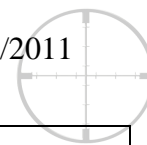
<b>postupnosť</b>		
<b>Rozširujúce body</b>		



Obr. 6. Diagram aktivít pre UC07 – Vytvoriť väzbu medzi dokumentmi

Tab. 8. Opis UC08 – Editovať väzbu medzi dokumentmi

<b>Identifikátor</b>	UC08	
<b>Názov</b>	Editovať väzbu medzi dokumentmi	
<b>Opis</b>	Používateľ upraví atribúty niektorej väzby na základe nových zistení, prípadne iných okolností.	
<b>Priorita</b>	Stredná až vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Väzba, ktorú chce používateľ upraviť je zobrazená.	
<b>Výstup. podm.</b>	Atribúty väzby sú zmenené a tieto zmeny sú uložené v systéme.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ označí väzbu, ktorej atribúty si želá upraviť.



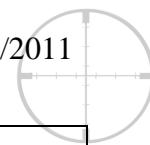
	2	Používateľ vyvolá akciu upraviť väzbu.
	3	System zobrazí dialóg na úpravu väzby.
	4	Používateľ upraví atribúty väzby a zvolí uloženie vykonaných zmien.
	5	System zmenenú väzbu uloží.
	6	Prípad použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
<b>Rozširujúce body</b>		

Tab. 9. Opis UC09 – Odstrániť väzbu medzi dokumentmi

<b>Identifikátor</b>	UC09	
<b>Názov</b>	Odstrániť väzbu medzi dokumentmi	
<b>Opis</b>	Používateľ zistí, že niektorá väzba nevyjadruje skutočnú súvislosť medzi dokumentmi a tak ju odstráni. Takáto situácia môže nastať napríklad preto, že systém niektoré väzby medzi dokumentmi vytvoril automaticky, ale tieto neboli vytvorené správne.	
<b>Priorita</b>	Stredná až vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Väzba, ktorú chce používateľ vymazať je zobrazená.	
<b>Výstup. podm.</b>	Väzba je odstránená zo systému a nie je zobrazená.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ označí väzbu, určenú na vymazanie.
	2	Používateľ vyvolá akciu odstrániť väzbu.
	3	System vyzve používateľa na potvrdenie svojej voľby.
	4	Používateľ voľbu potvrdí.
	5	System vyznačenú väzbu odstráni.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	4.a	Používateľ voľbu odmietne.
	5.a	Prípad použitia končí.
<b>Rozširujúce body</b>		

Tab. 10. Opis UC10 – Vygenerovať dokument

<b>Identifikátor</b>	UC 10
<b>Názov</b>	Vygenerovať dokument
<b>Opis</b>	Vygenerovanie nového dokumentu na základe zvolených dokumentov.
<b>Priorita</b>	Vysoká
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Musí byť zvolený aspoň jeden dokument.
<b>Výstup. podm.</b>	Je vytvorený dokument na základe zvolených dokumentov.
<b>Aktér</b>	Používateľ



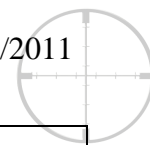
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí vytvorenie nového dokumentu.
	2	Systém ponúkne časti jednotlivých dokumentov pre ďalšiu selekciu.
	3	Používateľ vyberie časti dokumentov.
	4	Systém vygeneruje nový dokument.
	5	Systém zobrazí novo vygenerovaný dokument.
	6	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
<b>Rozširujúce body</b>		

Tab. 11. Opis UC11 – Vytvoríť prepojavací (virtuálny) dokument

<b>Identifikátor</b>	UC 11	
<b>Názov</b>	Vytvoríť prepojavací (virtuálny) dokument	
<b>Opis</b>	Vytvorenie dokumentu, ktorý slúži na umelé prepojenie jednotlivých dokumentov.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Musia byť načítané a zobrazené aspoň dva dokumenty.	
<b>Výstup. podm.</b>	Je vytvorený virtuálny dokument a je prístupný používateľovi spolu s ostatnými dokumentmi.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí vytvorenie virtuálneho dokumentu.
	2	Používateľ zvolí z existujúcich dokumentov časti, ktoré sa využijú.
	3	Systém vytvorí virtuálny dokument a pridá ho k existujúcim.
	4	Systém zobrazí virtuálny dokument.
	5	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	2.a	Používateľ vytvorí väzby od zobrazených dokumentov k novému virtuálnemu dokumentu
	3.a	Pokračuje sa bodom 3.
<b>Rozširujúce body</b>		

Tab. 12. Opis UC12 – Prepínať 2D/3D rozhranie

<b>Identifikátor</b>	UC 12	
<b>Názov</b>	Prepínať 2D/3D rozhranie	
<b>Opis</b>	Prepínanie medzi zobrazením 2D a 3D.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Musí existovať zobrazovaný graf.	
<b>Výstup. podm.</b>	Zobrazenie je prepnuté do zvoleného režimu.	
<b>Aktér</b>	Používateľ	



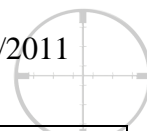
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí prepnutie do 3D.
	2	Systém zobrazí graf v 3D.
	3	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1.a	Používateľ zvolí prepnutie do 2D.
	2.a	Systém zobrazí graf v 2D.
	3.a	Prípád použitia končí.
<b>Rozširujúce body</b>		

Tab. 13. Opis UC13 – Pohybovať sa v 3D priestore

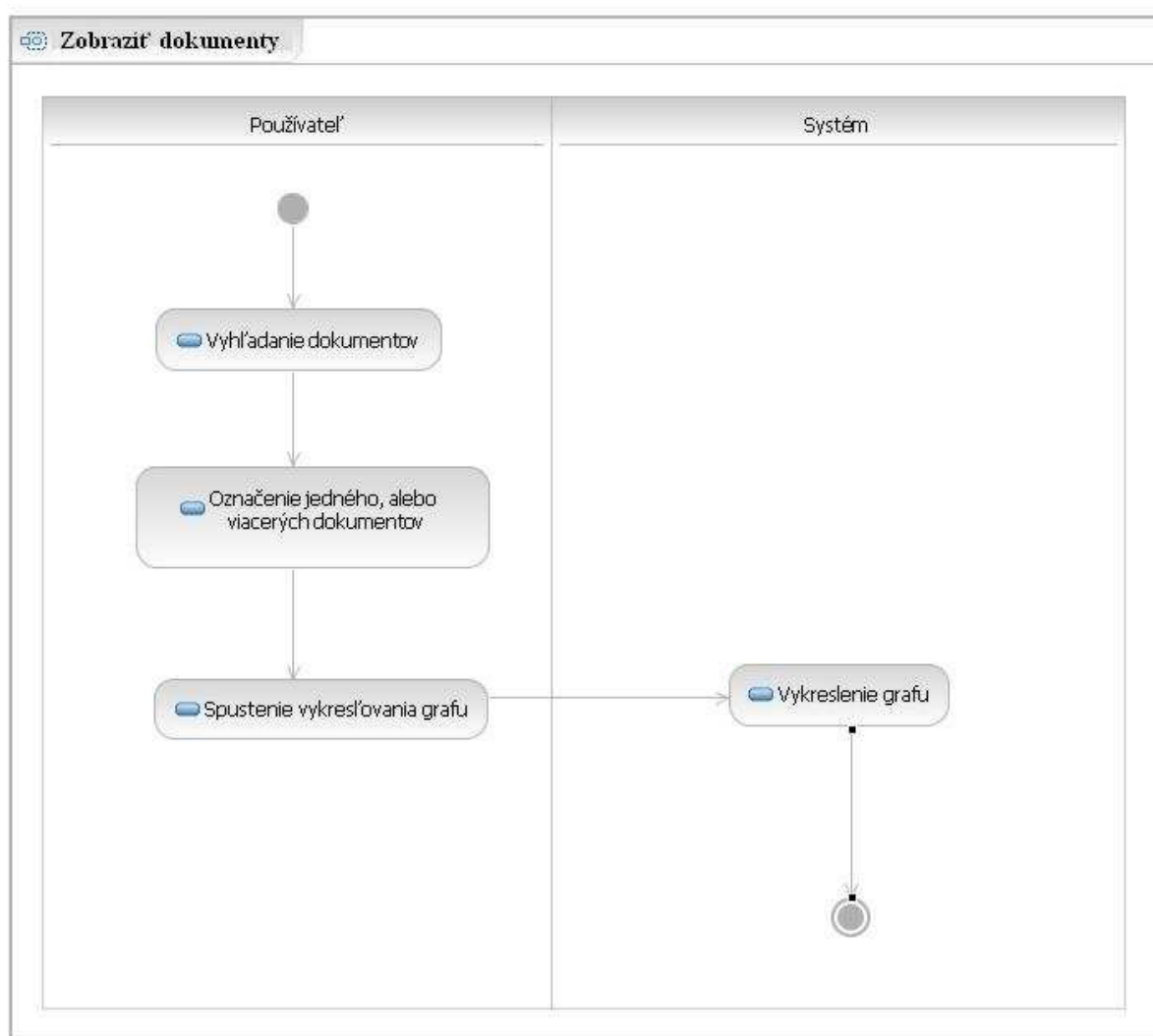
<b>Identifikátor</b>	UC 13	
<b>Názov</b>	Pohybovať sa v 3D priestore	
<b>Opis</b>	Pohybovanie grafom v 3D priestore pozostávajúce z približovania a otáčania.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Zobrazenie musí byť prepnuté do 3D zobrazenia. Musí existovať vytvorený graf.	
<b>Výstup. podm.</b>	Používateľovi je dovolené sa pohybovať v 3D priestore.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ použije na otáčanie grafu myš alebo klávesy.
	2	Systém zobrazí požadované otočenie.
	3	Prípád použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1.a	Používateľ použije na priblíženie alebo oddialenie grafu koliesko na myši alebo klávesy.
	2.a	Systém zobrazí požadované priblíženie.
	3.a	Prípád použitia končí.
<b>Rozširujúce body</b>		

Tab. 14. Opis UC14 – Zobrazit' dokumenty

<b>Identifikátor</b>	UC14	
<b>Názov</b>	Zobrazit' dokumenty	
<b>Opis</b>	Používateľ zvolí dokument, ktorý chce zobrazit' v grafe spolu s jeho vzťahmi k ostatným dokumentom.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je prihlásený do systému. Na zozname vyhľadanych dokumentov sa nachádza aspoň jedna položka.	
<b>Výstup. podm.</b>	Systém zobrazí vybrané dokumenty aj s príslušnými vzťahmi.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ vyhľadá dokumenty pomocou primárneho toku UC01.



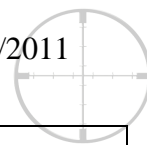
	2	Používateľ si zvolí jeden dokument ako centrálny a dá ho zobrazit'.
	3	System zobrazí v grafe centrálny dokument spolu so všetkými dokumentmi, ktoré sú priamo spojené väzbou s centrálnym dokumentom.
	4	Prípad použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	2.a	Používateľ si zvolí možnosť zobrazit' viaceré (prípadne všetky vyhl'adané) dokumenty.
	3.a	System zobrazí plochu, na ktorej budú všetky zvolené dokumenty považované za centrálny a následne sa postupuje tak isto ako v kroku 3 primárnej postupnosti.
<b>Rozširujúce body</b>		



Obr. 7. Diagram aktivít pre UC14 – Zobrazit' dokumenty

Tab. 15. Opis UC15 – Prihlásiť do systému

<b>Identifikátor</b>	UC15
<b>Názov</b>	Prihlásiť do systému
<b>Opis</b>	Používateľ sa chce prihlásiť do systému, aby mohol plnohodnotne využívať



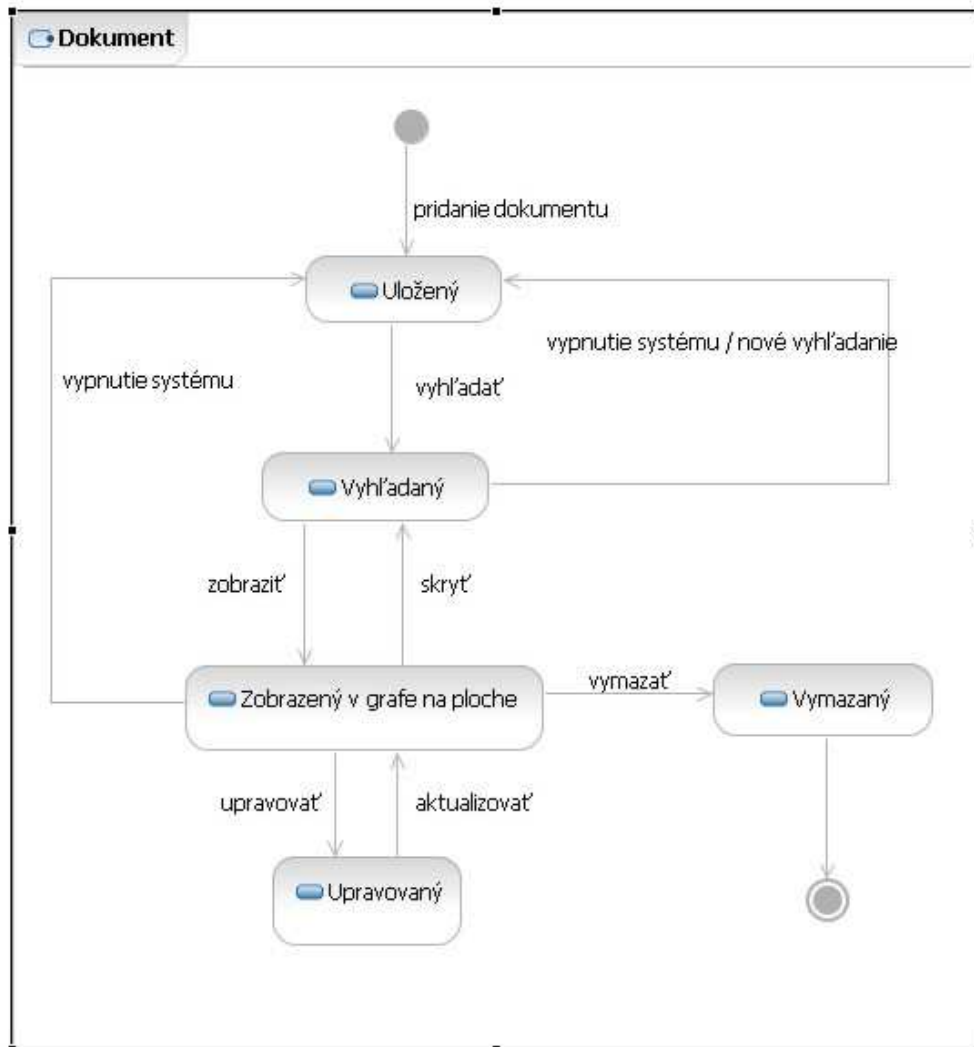
	funkcionalitu programu.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ je zaregistrovaný v systéme.	
<b>Výstup. podm.</b>	Používateľ je prihlásený v systéme	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí možnosť, prihlásiť do systému.
	2	Používateľ napíše meno a heslo a potvrdí prihlásenie.
	3	System overí, že zadané údaje sú správne.
	4	System prihlási používateľa do systému
	5	Prípad použitia končí.
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	3.a	System zistil, že údaje nie sú správne a upovedomí o tom používateľa.
	4.a	Pokračuje sa krokom 2 primárneho toku.
<b>Rozširujúce body</b>		

Tab. 16. Opis UC16 – Zaregistrovať do systému

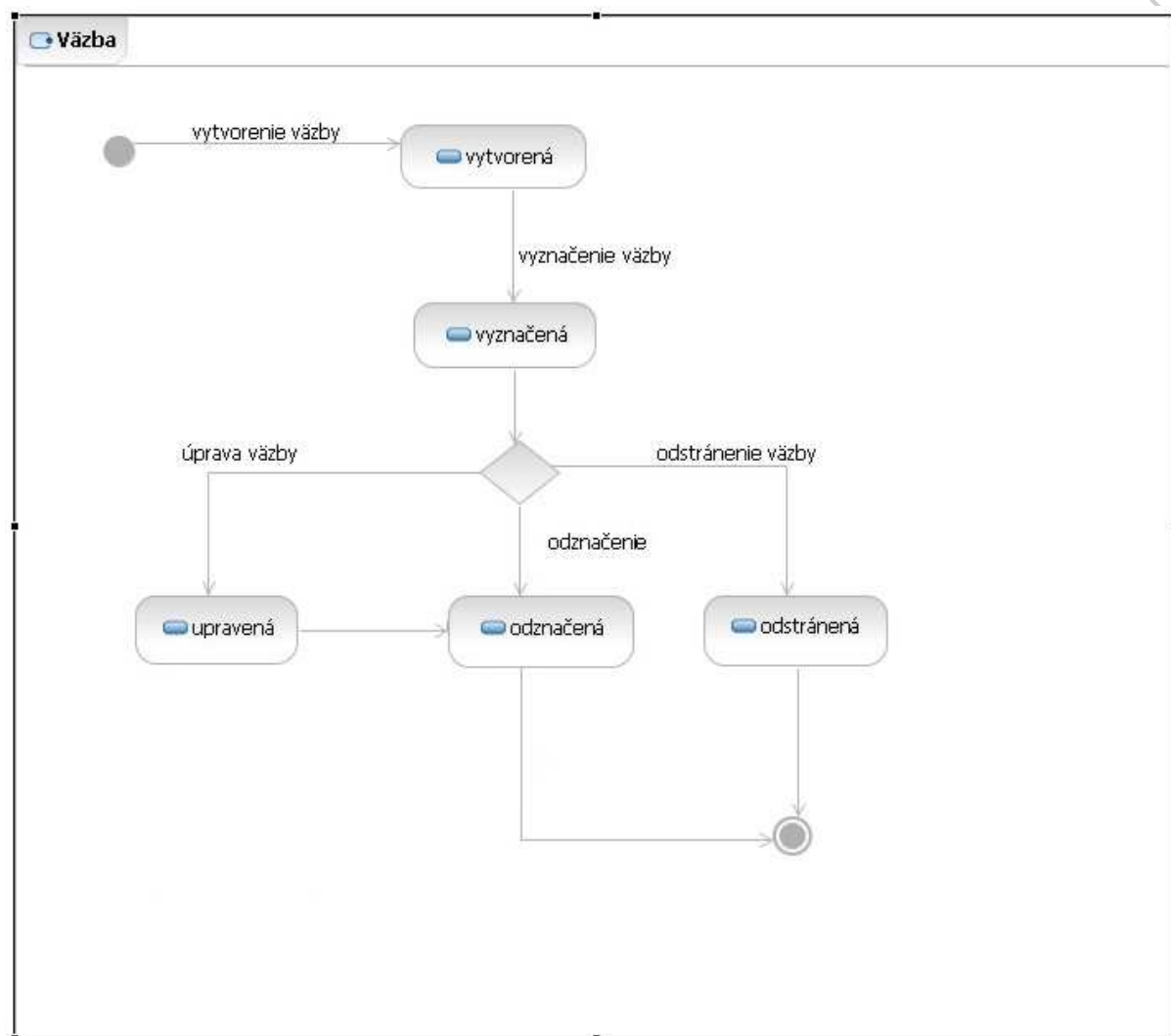
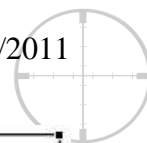
<b>Identifikátor</b>	UC16	
<b>Názov</b>	Zaregistrovať do systému	
<b>Opis</b>	Používateľ sa chce zaregistrovať do systému.	
<b>Priorita</b>	Vysoká	
<b>Vstup. podm.</b>	Používateľ nie je zaregistrovaný v systéme.	
<b>Výstup. podm.</b>	Používateľ je zaregistrovaný v systéme.	
<b>Aktér</b>	Používateľ	
<b>Základná postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	1	Používateľ zvolí možnosť zaregistrovať do systému.
	2	System zobrazí registračný formulár.
	3	Používateľ vyplní zobrazené formuláre.
	4	System overí, či sú tieto údaje správne.
	5	System odošle kontrolný e-mail na adresu, ktorú zadal používateľ a upovedomí ho o tom.
	6	Používateľ navštívi odkaz, ktorý dostal v tele e-mailu a tým sa stane platne zaregistrovaným.
7	Prípad použitia končí.	
<b>Alternatívna postupnosť</b>	<b>Krok</b>	<b>Činnosť</b>
	4.a	System zistil, že vyplnené údaje nie sú správne (nesedí štruktúra e-mailu, už existuje používateľ s daným menom atď.) a upovedomí o tom používateľa.
	5.a	Pokračuje sa krokom 2 primárneho toku.
<b>Rozširujúce body</b>		



Pre entity Dokument a Väzba sme vytvorili aj stavové diagramy, aby sme znázornili ďalšie kľúčové záležitosti týkajúce sa špecifikácie správania systému. Diagramy predstavujú obrázky č. 8 a 9. Entity a dátový model sú popísané v nasledujúcej kapitole.

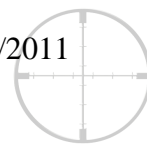


Obr. 8. Stavový diagram pre dokument



Obr. 9. Stavový diagram pre väzbu

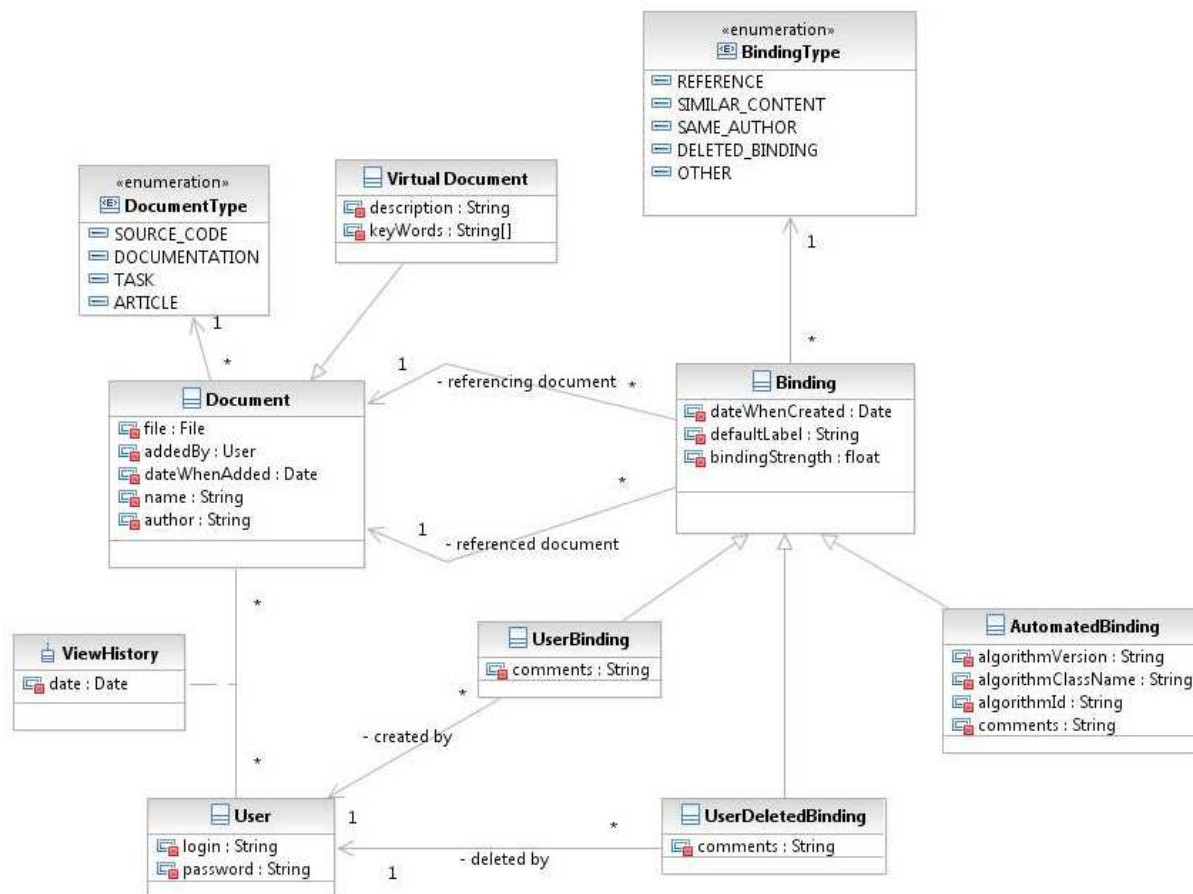




## 2.3 Špecifikácia údajov v systéme

V tejto kapitole sa nachádza dátový model systému spolu s opisom jednotlivých tried, ich atribútov, ohraničení a iných podmienok. Na konci kapitoly uvádzame odlišnosti špecifikácie údajov nášho systému a systémov, z ktorých vychádzame.

### 2.3.1 Dátový model systému



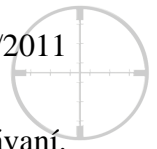
Obr. 10. Dátový model

Dátový model zobrazený na obrázku č. 10 bol vytvorený na základe návrhu minuloročného tímu č. 20. Nasleduje popis jednotlivých entít prevzatý z dokumentácie.

**Document** – reprezentuje dokument v báze znalostí. Eviduje podrobnosti kto a kedy ho vložil, ako aj niektoré metadáta, ktoré môžu pomôcť pri vyhľadávaní väzieb (autor, atď.). File je súbor, ktorý je týmto dokumentom reprezentovaný.

Povinné atribúty: name

**VirtualDocument** – ide o taký dokument, ktorý neobsahuje samotné znalosti, ale slúži na združenie dokumentov obsahujúcich znalosti o určitej doméne. Kľúčové slová slúžia na to,



aby aj takýto dokument bol zaindexovaný a dal sa vyhľadať pri fulltextovom vyhľadávaní. Taktiež slúžia kľúčové slová a popis ako informácia pre jeho čitateľov.

**Binding**- reprezentuje väzbu medzi dokumentmi. Sila väzby je číslo medzi 0-1, ktoré určuje, aká je silná previazanosť dokumentov. Čím je číslo väčšie, tým je väzba silnejšia. Typ väzby určuje, o aký druh previazanosti sa jedná.

Povinné atribúty: bindingStrength, bindingType, referencingDocument, referencedDocument

**BindingType** – určuje aký reálny význam má väzba. Reálnym významom myslíme spôsob, ako sú sémanticky dokumenty spojené.

- REFERENCE – jeden dokument sa priamo odkazuje druhý referenciou, alebo referencovaný dokument detailnejšie popisuje niektoré koncepty, aspekty, fakty z odkazujúceho sa dokumentu.
- SIMILIAR\_CONTENT – dokumenty majú podobný obsah, venujú sa podobnej téme.
- SAME\_AUTHOR – dokumenty majú toho istého autora.
- DELETED\_BINDING – väzba, ktorá bola používateľom manuálne zmazaná. Väzba takéhoto typu je tu len pre algoritmy vyhľadávajúce väzby, aby medzi týmito dokumentmi ďalšie väzby nevytvárali.
- OTHER – väzba iného nešpecifikovaného významu.

Väzby sú ďalej dedičnosťou rozčlenené na nasledovné triedy (BindingClass):

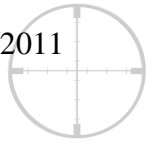
**UserBinding** – väzba vytvorená niektorým používateľom. Eviduje sa okrem iného aj používateľ, ktorý väzbu vytvoril a jeho komentár k väzbe. Nesmie byť typ väzby DELETED\_BINDING.

**UserDeletedBinding** – väzba, ktorá bola používateľom zmazaná. Ide o informáciu pre algoritmy vyhľadávajúce väzby, že medzi týmito dokumentmi väzby tvoriť nemajú. Eviduje sa okrem iného aj používateľ, ktorý väzbu vytvoril a jeho komentár k väzbe. Väzba musí mať typ väzby DELETED\_BINDING.

**AutomaticBinding** – všetky väzby, ktoré boli vytvorené automaticky algoritmom hľadajúcim väzby medzi dokumentmi. Eviduje sa ktorý algoritmus väzbu vytvoril (id, kvalifikované meno triedy, verzia). Atribút comments eviduje poznámky algoritmu, z ktorých má byť jasné na akom základ väzbu vytvoril. Typ väzby nesmie byť DELETED\_BINDING.

Povinné atribúty: algorithmId, algorithmClassName, algorithmVersion

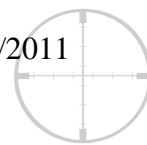
**User** – reprezentuje používateľa systému.



Povinné atribúty: login, password

Nad existujúcim dátovým modelom navrhujeme nasledovné zmeny:

- Vytvorenie entity **DocumentType**, ktorá by reprezentovala typ vloženého dokumentu. Potreba definovania typu dokumentu vznikla na základe úlohy generovania nových dokumentov. Vďaka typu dokumentu je možné definovať rôzne spôsoby spájania týchto dokumentov. Takisto na základe typu dokumentu bude možné ich rôzne zobrazenie v grafe.
- Vytvorenie asociačnej triedy **ViewHistory**, ktorá reprezentuje históriu prehliadania jednotlivých dokumentov používateľmi.

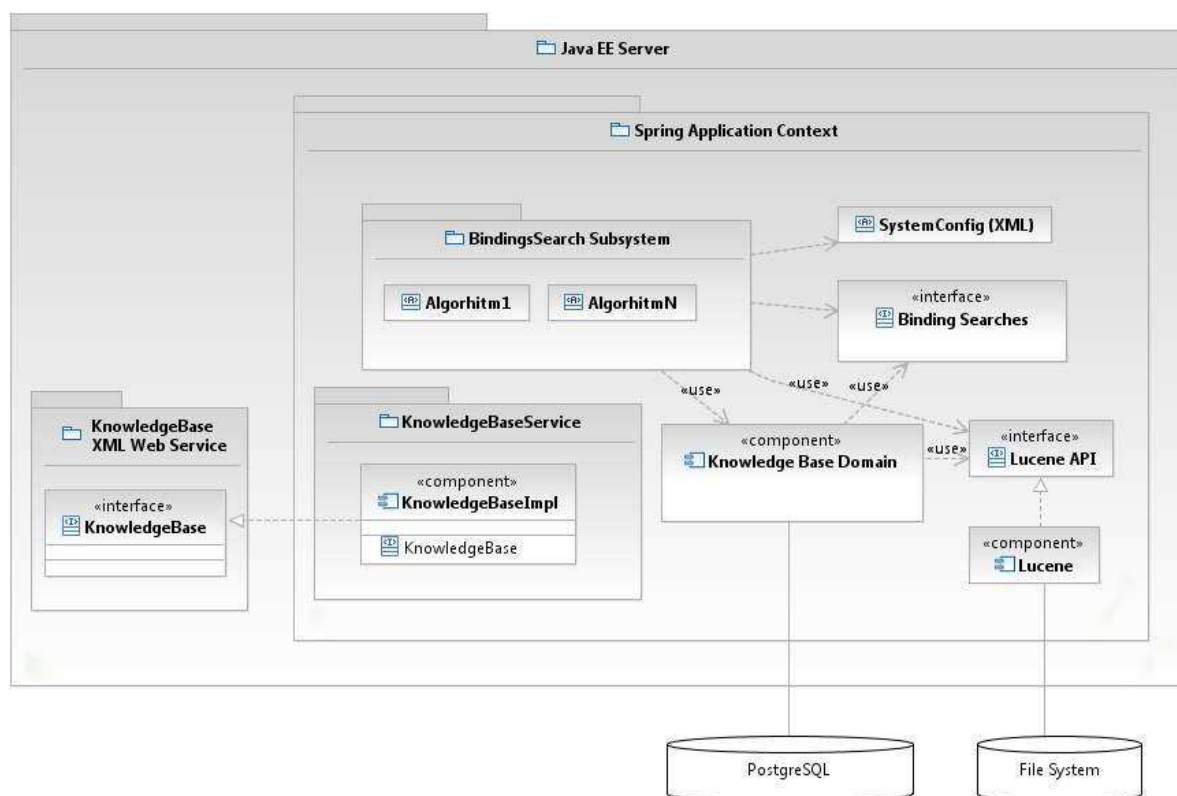


## 3 Návrh riešenia

### 3.1 Návrh architektúry

Základným rozdelením architektúry vytváraného systému je rozdelenie na klientskú a serverovú časť. Obe tieto časti nadväzujú na prácu minuloročných tímov a ich návrhov. V závere kapitoly znázorňujeme komunikáciu klientskej a serverovej časti.

#### 3.1.1 Server



Obr. 11. Server

Serverová časť aplikácie znázornená na obrázku č. 11 z veľkej časti preberá štruktúru z minuloročného projektu. Ako hlavný kontajner je na server strane použitý Java aplikačný server. V aplikačnom serveri bude systém bežať ako Spring aplikácia. To znamená, že necháme technológiu Spring, aby nakonfigurovala väzby v rámci nášho systému. BindingSearchSubsystem je konfigurovaný cez samostatné xml, kde sú definované jeho algoritmy. Tento XML súbor bude tiež vo formáte frameworku Spring. Po jeho zavedení do aplikačného kontextu sa vytvoria vlákna, reprezentujúce jednotlivé algoritmy vyhľadávania väzieb. Pri zavádzaní nového algoritmu je potrebné naprogramovať triedu tohto algoritmu,



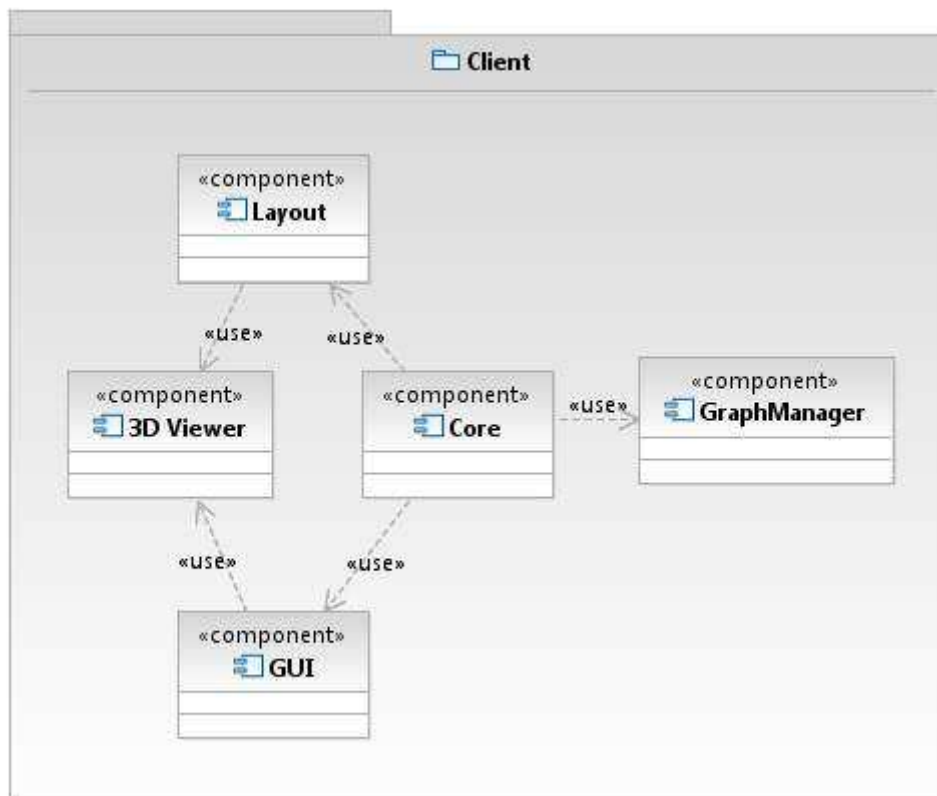
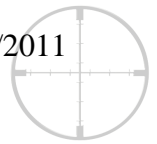
nakonfigurovať ju v XML, prekompilovať celý systém a opätovne nasadiť aplikáciu. Výhodou oddelenia je, že pri pridávaní nového algoritmu nebudú zasiahnuté iné časti systému. Vyhľadávacím algoritmom je k dispozícii Lucene API, ako aj KnowledgeBaseDomain API. Lucene používa vyhradený adresár v súborovom systéme pre ukladanie svojich indexov. KnowledgeBaseDomain je vrstva doménových objektov a business logiky okolo nich. Perzistencia je vyriešená pomocou frameworku Hibernate v spojení s PostgreSQL relačnou databázou. KnowledgeBaseService je vrstva, ktorá vystavuje rozhranie KnowledgeBase.

Nami navrhovaná zmena predstavuje zmenu technológie vzdialeného volania metód z použitého Java Remote Method Invocation na multiplatformový štandard XML Web Services, ktorý nám umožní volanie metód bázy znalostí na ľubovoľnej implementačnej platforme klienta, keďže komunikácia prebieha na báze SOAP (XML) nad protokolom HTTP.

### 3.1.2 Klient

Vzhľadom na rozdelenie serverovej a klientskej časti a možnosti použitia viacerých platforiem / programovacích jazykov bude možné jednotlivých klientov jednoducho nahrádzať, prípadne používať viacerých klientov naraz.

Vzhľadom na to, že nie je jednoznačne definovaná implementačná platforma, navrhovaná architektúra je na vyššej úrovni abstrakcie. Podobne ako serverová časť, vychádza z návrhu minuloročného tímu.

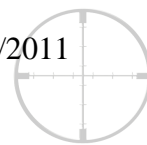


Obr. 12. Klient

Popis jednotlivých komponentov klienta zobrazeného na obrázku č. 12:

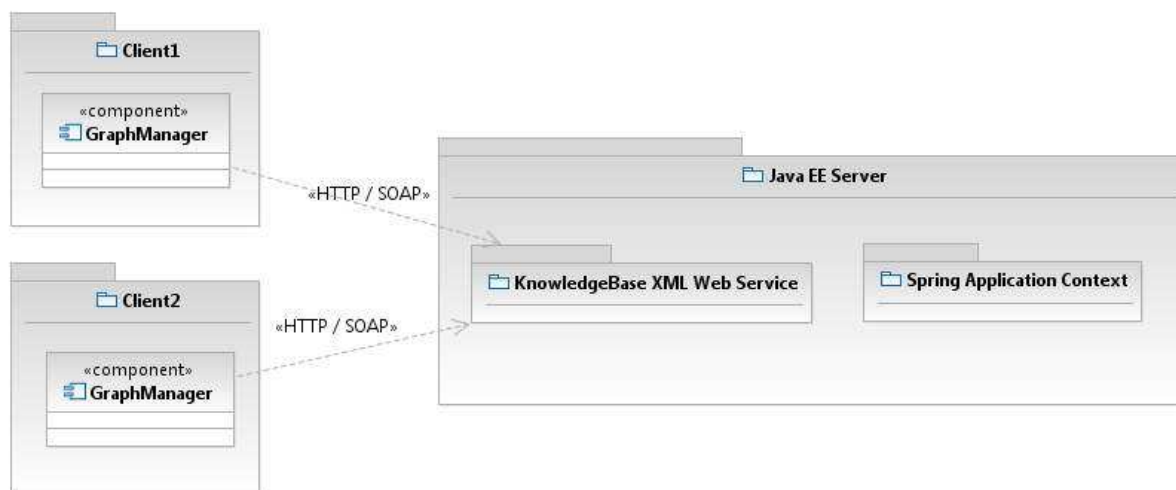
- Core – zabezpečuje samotný beh aplikácie, riadenie ostatných komponentov, ich synchronizáciu a pod.
- Layout – zabezpečuje rozmiestňovanie uzlov v trojdimenzionálnom priestore
- 3D Viewer – predstavuje komponent, ktorý bude realizovať vizualizáciu grafu
- GUI – komponent, ktorý bude obsahovať interakciu s používateľom nad zobrazovaním grafom
- GraphManager – komponent, ktorého hlavnou zodpovednosťou bude komunikácia so serverovou časťou

Implementácia bude prioritne prebiehať na platforme C++. Hlavným cieľom však je, aby bolo možné klientov jednoducho prepínať a v prípade potreby implementovať klienta aj inom prostredí. Vďaka tomuto prístupu bude umožnená aj súbežná činnosť už existujúceho 2D klienta a nami implementovaného 3D klienta.



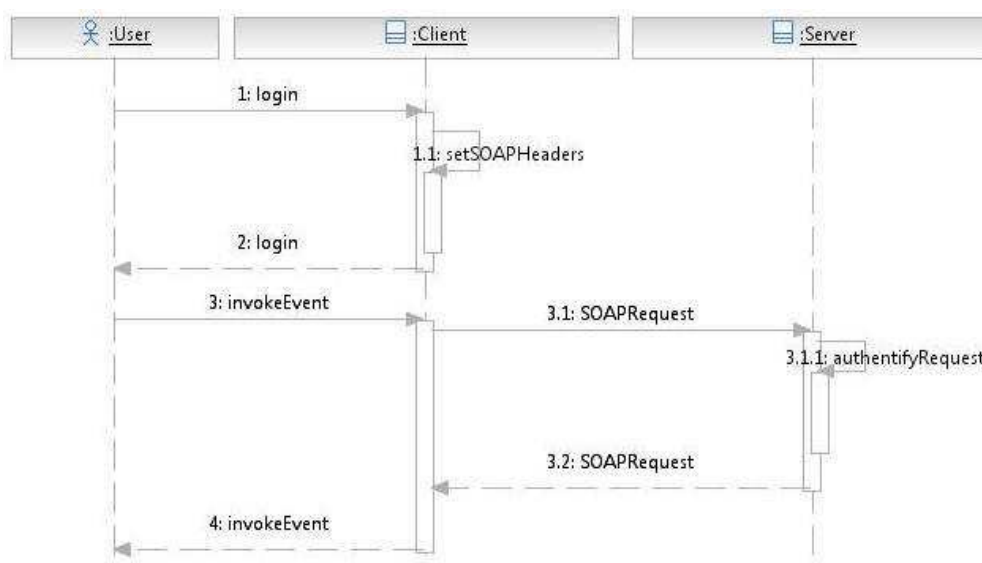
### 3.1.3 Komunikácia server – klient

Komunikácia medzi serverovou a klientskou aplikáciou, ako už bolo vyššie spomenuté, bude prebiehať prostredníctvom protokolu SOAP (Simple Object Access Protocol) nad protokolom HTTP. Komunikácia bude prebiehať medzi komponentom webovej služby na strane servera a komponentom graf manažéra na strane klienta. Znáročujeme ju na obrázku č. 13.

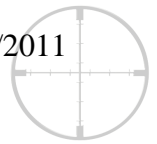


Obr. 13. Komunikácia klient-server

Keďže používanie serverovej aplikácie bázy znalostí si vyžaduje autentifikáciu používateľa, budú autentifikačné údaje súčasťou každej SOAP správy. Údaje, ktoré slúžia na identifikáciu používateľa budú zapísané v časti SOAP Header. Tieto údaje zadá používateľ pri spúšťaní klienta. Pri každom volaní nejakej vzdialenej metódy serverová časť overí používateľa a vráti odpoveď v závislosti od výsledku autentifikácie. Komunikácia je znázornená na obrázku č. 14.



Obr. 14. Sekvenčný diagram komunikácie klient-server



## 4 Prototyp

### 4.1 Ciele prototypovania

Naším cieľom v tejto fáze tvorby projektu bolo vytvoriť prototyp, ktorý bude ponúkať základnú funkcionality výsledného systému. Použijeme metódu evolučného prototypovania, teda hotový prototyp budeme postupne ďalej upravovať a vylepšovať až do stavu, ktorý bude zodpovedať špecifikácii systému - teda výsledný produkt. Vytvorenie prototypu je dôležité hlavne pre zákazníka (resp. nášho pedagogického vedúceho tímu), aby videl priebežné výsledky našej práce. Pri prezentácii prototypu je tiež možné odhaliť prípadné nezrovnalosti v požiadavkách zákazníka na hotový systém oproti predstavám vývojárskeho tímu.

Témou nášho projektu je prepojenie systému zobrazujúceho dokumenty a ich vzájomné väzby v 2D grafe a programu zobrazujúceho 3D graf. Prototyp by teda mal ukázať práve toto prepojenie existujúcich systémov do jedného celku. Na splnenie tohto cieľa je potrebné vytvoriť server, s ktorým bude klient komunikovať. Na tomto serveri bude bežať databáza, v ktorej bude uložená tzv. báza znalostí. Sú to dokumenty, v ktorých bude používateľ môcť vyhľadávať a zobrazovať ich v grafe. Ďalej bude potrebné upraviť existujúcich klientov tak, aby medzi sebou vedeli komunikovať. To je nevyhnutné na to, aby mohol používateľ premietnuť vytvorený 2D graf do 3D prostredia.

Ďalším cieľom prototypu je implementovať algoritmy na extrakciu informácií z dokumentu, presnejšie kľúčových slov priamo z textu dokumentu.

Prototyp nebude zahŕňať prihlásenie používateľa do systému, ani viacpoužívateľské prostredie. Taktiež nebude ešte implementované hľadanie väzieb medzi dokumentmi a pridávanie nových dokumentov do systému.

### 4.2 2D klient

Táto kapitola obsahuje zmeny vykonané v 2D klientovi oproti výsledku minuloročného tímu.

#### 4.2.1 Odstránenie nepotrebných častí

- Odstránenie závislostí aplikácie na knižnici Spring
  - Odstránenie triedy `ApplicationContextHolder`
  - Prepísanie závislých častí kódu





## 4.2.2 Konfigurácia

- Vytvorená nová trieda Config
  - statická inicializácia
  - dáta vo formáte XML
  - prístup k hodnotám v štýle kľúč-hodnota

Ukážka zápisu v XML

```
<add key="service" value="http://147.175.159.184:8080/D9vinaServer/docs" />
```

Prístup v kóde:

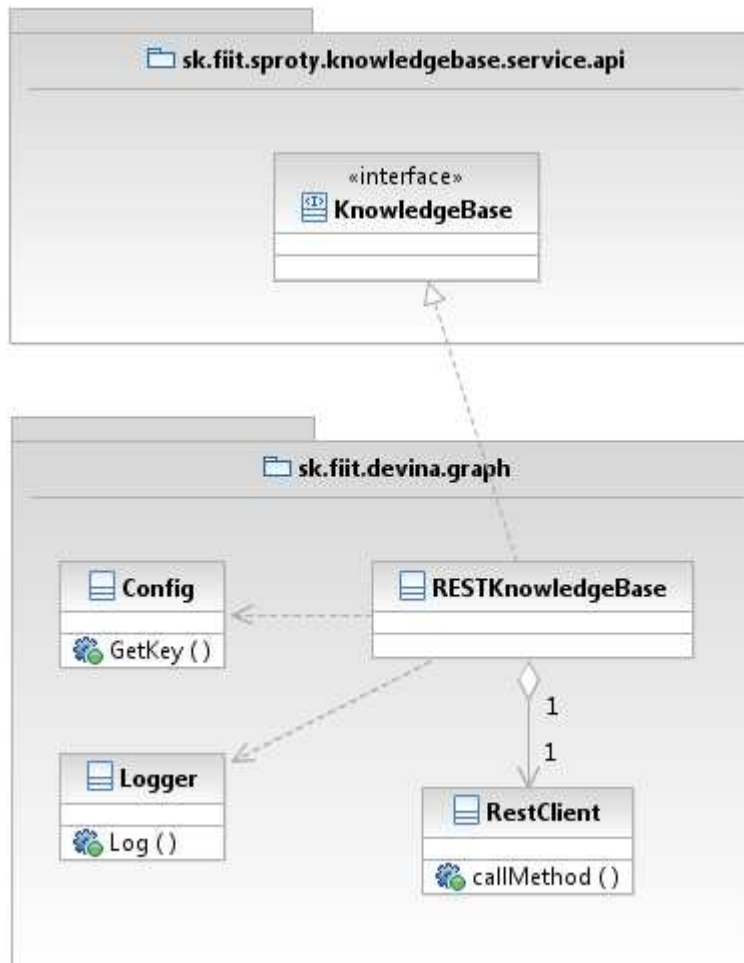
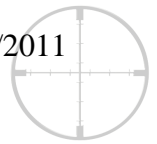
```
Config.get("service")
```

## 4.2.3 Logovanie

- Vytvorenie triedy Logger
  - statická inicializácia
  - nahradenie výpisov do konzoly
  - konfigurovateľná cez config.xml

## 4.2.4 Nová implementácia rozhrania KnowledgeBase

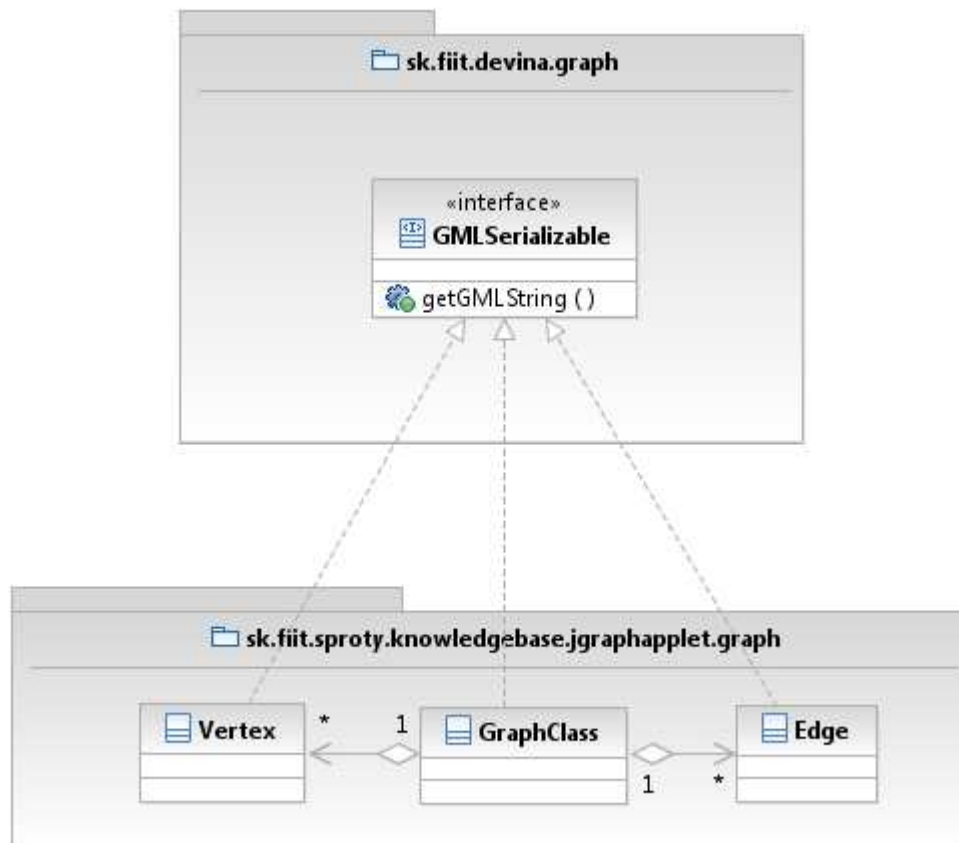
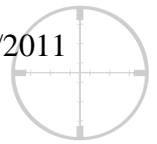
- Nová trieda RESTKnowledgeBase
  - implementuje pôvodné rozhranie KnowledgeBase
  - odstránenie technológie RMI
  - komunikácia prebieha na základe HTTP a architektúre REST
- Nová trieda RestClient
  - sprehľadňuje HTTP komunikáciu
  - volanie metód prebieha s využitím protokolu HTTP a XML s využitím serializačných mechanizmov Java platformy



Obr. 15. Nová implementácia rozhrania KnowledgeBase

#### 4.2.5 GraphML serializácia grafov

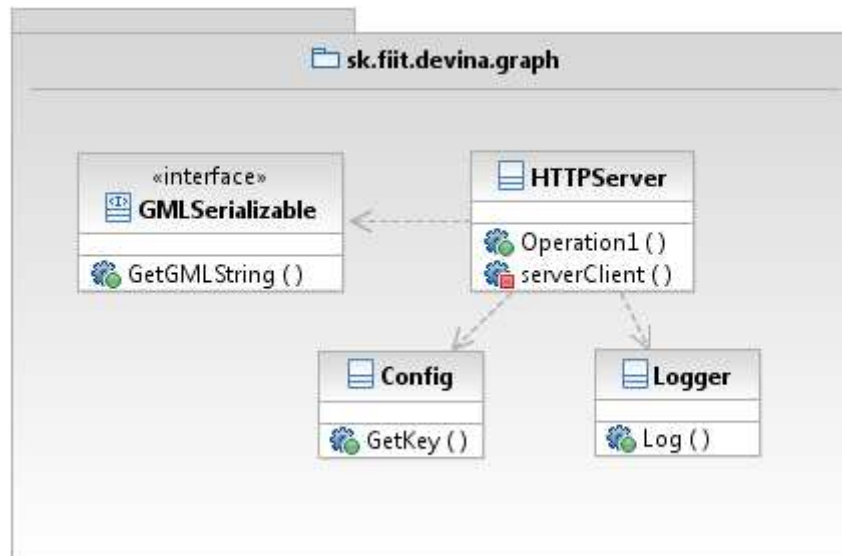
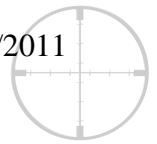
- Vytvorené nové rozhranie GMLSerializable:
  - doplnená implementácia pôvodných tried pre reprezentáciu grafu, vrcholov a hrán
- Serializáciu grafu do GraphML je možné vyvolať metódou rozhrania getGMLString() nad konkrétnou inštanciou grafu.



Obr. 16. GraphML serializácia grafov

#### 4.2.6 Server

- Vytvorená nová trieda `HTTPServer`, ktorá sprístupňuje graf vo formáte GraphML prostredníctvom HTTP protokolu.
- Pri každej požiadavke zo strany HTTP klienta je GraphML vygenerované nanovo na základe aktuálneho grafu.



Obr. 17. Server

## 4.3 Server

V kapitole uvádzame zmeny serveru oproti serveru minuloročného tímu spolu s vlastnosťami a konfiguráciou serveru.

### 4.3.1 Zmeny oproti serveru minuloročného tímu

- Nami vytvorený server po tíme z minulého roka takmer nič nepreberá.
- Využitie sú iba zdrojové kódy tried, ktoré reprezentujú tzv. DTO objekty (Data Transfer Object)
- Tieto triedy boli prepísané na komponenty Java Beans, z dôvodu, aby mohli byť plnohodnotne využívané triedami XMLEncoder a XMLDecoder, ktoré patria do balíčka java.beans. Tie sú využité na zakódovanie a dekodovanie DTO objektov do XML formy, ktoré sú použité pri komunikácii medzi klientom a serverom.

### 4.3.2 Vlastnosti serveru

- Server beží ako Java HTTP Servlet.
- Server beží na aplikačnom serveri Jetty, ktorý je celý napísaný v programovacom jazyku Java.



### 4.3.3 Konfigurácia serveru

- Konfigurácia je sústredená do súboru web.xml:
  - obsahuje názov triedy, ktorý predstavuje hlavný HTTP servlet (v našom prípade je to trieda MainServlet)
  - taktiež obsahuje vzor pre mapovanie daného servletu na konkrétne triedy URL (v našom prípade existujú dve triedy: /docs/\* - trieda zaoberajúca sa prácou s dokumentmi, väzbami atď. a /user/\* - trieda zaoberajúcou sa prácou s používateľom)
  - neobsahuje mapovanie konkrétnych URL na vybranú funkcionálnosť podľa architektúry REST (to je súčasťou triedy MainServlet)

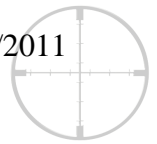
Štruktúra súboru web.xml:

```
<servlet>
  <servlet-name>názov servletu</servlet-name>
  <servlet-class>názov triedy, ktorá reprezentuje
servlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>názov servletu</servlet-name>
  <url-pattern>URL, na ktorú je namapovaný daný
servlet</url-pattern>
</servlet-mapping>
```

### 4.3.4 Komunikácia serveru s 2D klientom

- Komunikácia prebieha pomocou HTTP protokolu, výhradne pomocou metód POST
- Podľa architektúry REST sú konkrétne URL namapované na dané zdroje (napr. URL <http://147.175.159.184:8080/D9vinaServer/docs/findDoc> predstavuje funkcionálnosť pre vyhľadanie konkrétnych dokumentov na serveri).
- Jednotlivé Java objekty prenášané pomocou metódy POST protokolu HTTP sú vo forme XML, ktoré sú vytvorené triedou XMLEncoder. Tieto objekty možno jednoducho po ich prijatí deserializovať pomocou triedy XMLDecoder.

Príklad štruktúry XML vytvoreného triedou XMLEncoder pre objekt typu DtoDocument:



```
<object
class="sk.fiit.sproty.knowledgebase.service.api.dto.DtoDocumen
t">
  <void property="addedBy">
    <long>4</long>
  </void>
  <void property="author">
    <string>Klensin</string>
  </void>
  <void property="dateWhenAdded">
    <object class="java.util.Date">
      <long>1290294000000</long>
    </object>
  </void>
  ...
```

### 4.3.5 Komunikácia serveru s databázou Postgre SQL

- Servlet využíva konfiguračný súbor XML na napojenie sa na databázu Postgre SQL.
- Servlet taktiež využíva mapovacie súbory XML, vytvorené pre konkrétne DTO objekty.
- Servlet okrem toho využíva metódy napísané v programovacom jazyku Java, ktoré spolupracujú so spomínanými XML súbormi a vracajú konkrétne DTO objekty. Tieto objekty sú následne posielané klientskej časti aplikácie.

### 4.4 3D klient

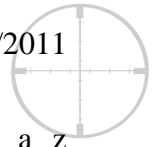
V klientovi (C++) boli vykonané nasledujúce zmeny:

- prídanie funkcionality na získanie grafu uloženého vo formáte GraphML zo vzdialeného servera prostredníctvom http protokolu
- upravenie vytvárania návěstí pri uzloch a hranách zobrazovaného grafu

#### 4.4.1 Získanie grafu zo vzdialeného servera

Implementovanie tejto funkcionality bolo nevyhnuté v dôsledku komunikácie klienta so serverom. Momentálne prostredníctvom nej komunikuje 3D klient s 2D klientom, avšak v budúcnosti sa plánuje, že 3D klient bude prostredníctvom tejto funkcionality komunikovať priamo so serverom.

Táto časť je implementovaná preťažením metódy **parseGraph()**, ktorá ako parameter obdrží URL adresu vzdialeného servera. Táto adresa je uložená v konfiguračnom súbore pod



položkou *GraphMLParser.serverUrl*. Následne je vytvorené spojenie zo serverom a z odpovede, ktorá je odoslaná zo servera je vytvorený graf.

#### 4.4.2 Upravenie vytvárania návěstí vrcholov a hrán

Návestia zobrazované pri jednotlivých vrcholoch a hranách sa získavajú zo zdrojového GraphML súboru. Štruktúra toho súboru je nasledovná.

```
<graphml>
  <graph id="G" edgedefault="directed">
    <node id="n0">
      <data key="type">TYP_DOKUMENTU</data>
      <data>Nazov dokumentu</data>
    </node>
    <node id="n1">
      <data key="type">TYP_DOKUMENTU</data>
      <data>Nazov dokumentu</data>
    </node>
    <edge source="n0" target="n1">
      <data key="relation">TYP_VAZBY</data>
      <data key="KW">4</data>
      <data key="Auth">2</data>
    </edge>
  </graph>
</graphml>
```

Návestie pri vrchole je potom tvorené názvom dokumentu a návestie pri hranách vytvorené nasledovne: **KW : 4 | Auth : 2**

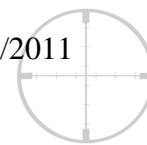
### 4.5 Algoritmy na extrakciu kľúčových slov a získavanie metadát z dokumentov

Táto kapitola obsahuje popis metódy extrakcie kľúčových slov z dokumentov a informácií z metadát.

#### 4.5.1 Pridaná funkcionálnosť

Minuloročný tím, ktorý pracoval na získavaní znalostí z dokumentov, neimplementoval extrakciu kľúčových slov priamo z dokumentu. Tím navrhol získavanie kľúčových slov a iných potrebných dát iba z metadát.

- Pridali sme reálnu extrakciu kľúčových slov z dokumentov.
- Implementovali sme získavanie dát (napr. autor) z metadát dokumentov.



- Podpora MS Word Document, PDF, plain text.
- Podpora slovenského jazyka – stop slová, lematizácia.

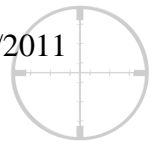
#### 4.5.2 Extrakcia kľúčových slov z dokumentov

- Spracovanie DOC, PDF, TXT súborov
- Konverzia DOC, PDF do TXT
  - V rámci spracovania sa volá externá aplikácia ConvertDoc spúšťaná s parametrami (vstup, výstup, možnosti).
- Problémy s kódovaním PDF, čiastočne vyriešené.
- Odstránenie stop slov a čísel
  - vlastný zoznam slovenských stop slov, možnosť použiť anglické stop slová.
  - odstraňovanie stop slov - dva krát.
    - 1. pred lematizáciou, aby sme zrýchlili proces lematizácie (aby neboli vyhľadávané základné tvary stop slov)
    - 2. po lematizácii (odstránenie stop slov prevedených do základného tvaru)
  - odstránenie čísel, pretože ponechanie čísel - čísla sú nevhodné (číslované zoznamy...)
- Lematizácia
  - zmena slov do základného (slovníkového tvaru) - potreba slovníka - súbor form2lemma.cdb (pochádza z JÚLŠ SAV). práca s týmto databázovým súborom – využitie manuálu k CDB a konkrétne použitie knižnice strangegizmo.cdb.
- Početnosti slov – TF (term frequency)
- Výber kľúčových slov
  - vybraté slová, ktorých početnosť v dokumente nie je menšia ako konštanta  $(0,5) * \text{početnosť najviac vyskytujúceho sa slova}$ . Nastaviteľná metóda.

#### 4.5.3 Získavanie informácií z metadát

- Využitie knižnice ICEpdf
  - extrakcia metadát z dokumentu pre následné použitie
  - potrebné informácie – autor, prípadne keywords





## 4.6 Databáza Postgre SQL

Kapitola opisuje špecifiká vytvorenia, naplnenia a práce s Postgre SQL databázou.

### 4.6.1 Vytvorenie novej databázy

- Vytvorenie databázy na základe analýzy
- Vzťahy m:n nahradené väzobnými tabuľkami
- Dedenie a špecializácia využité pri vytváraní tabuliek, ktoré sa týkajú väzieb

### 4.6.2 Naplnenie databázy

Väzby medzi dokumentmi pre naplnenie databázy prototypu zobrazené v Prílohe A zachytáva tabuľka č. 17.

**Tab. 17.** Väzby medzi dokumentmi

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14
14			x					x			x			
13		x			x					x				
12			x	x		x	x		x					
11				x		x		x						
10	x		x		x		x							
9		x				x								
8		x		x	x									
7	x		x	x		x								
6	x		x		x									
5		x		x										
4	x		x											
3		x												
2	x													
1														

### 4.6.3 iBatis

Nahradenie zbytočne robustného nástroja Hibernate iBatisom.



#### 4.6.4 Konfigurácia

Prostredníctvom konfiguračného XML nastavujeme umiestnenie databázy, prihlasovacie údaje a mapovacie XML, s ktorými ďalej pracujeme.

Štruktúra xml je nasledovná:

```
<sqlMapConfig>
    <transactionManager>
        <!--Obsahuje údaje o pripojení (meno, heslo, adresa,
meno databázy)-->
    </transactionManager>
        <!--Použité mapovacie xml dokumenty-->
    <sqlMap resource="Contact.xml"/>
    <sqlMap resource="DtoDocument.xml"/>
    <sqlMap resource="DtoBinding.xml"/>
</sqlMapConfig>
```

#### 4.6.5 Mapovacie súbory

Údaje z databázy získavame prostredníctvom funkcií definovaných v mapovacích súboroch.

Pre zvýšenie prehľadnosti sú tieto funkcie rozdelené do súborov podľa objektov, ku ktorým sa viažu.

Používané objekty:

- DtoBinding
- DtoAutomaticBinding
- DtoDocument

Pri mapovaní údajov z databázy na objekty triedy DtoBinding som vytvoril triedu DbDtoBinding, ktorá je presným obrazom dát v databáze a umožňuje jednoduchšie mapovanie.

#### 4.6.6 Príklad funkcie v DtoDocument.xml

```
<select id="getByKeyWords" resultClass="DtoDocument"
parameterClass="list">
```

- Id názov ktorým je funkcia select volaná z Java prostredia



- `resultClass` meno triedy objektu, na ktorý sa majú dáta namapovať
- `parameterClass` druh vstupných parametrov (map, list, object)

Pri použití vstupného parametru list môžeme nad ním iterovať a vytvoriť požadovaný select

```
<iterate open="( " close=")" conjunction="OR">
  keyword LIKE '%$[]$%'
</iterate>
```

- `open="( " close=")"` určuje ohraničenie príkazu vytvoreného iteráciou
- `conjunction="OR"` určuje spojenie medzi časťami príkazu
- `keyword LIKE '%$[]$%'` určuje, že porovnávané pole zo vstupného listu bude keyword, na porovnanie sa použije funkcia LIKE a daný reťazec sa bude hľadať kdekoľvek v slove

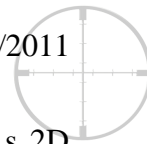
## 4.7 Zhodnotenie výsledkov prototypovania

Cieľom prototypovania bolo vytvorenie prototypov všetkých súčastí systému. Podarilo sa nám implementovať analýzu dokumentov, vizualizáciu v 2D a 3D prostredí, editáciu a manipuláciu s grafmi.

Tiež sme v rámci implementácie zmenili predchádzajúce technológie a to:

- Hibernate na iBatis
  - Nahradenie zbytočne robustného nástroja so zachovaním potrebnej funkcionality
- RMI na REST
  - Umožnenie prepojenia 3D klienta (C++) na server
  - Univerzálnosť architektúry REST
- JBoss na Jetty
  - Využitie plnej funkcionality Jetty namiesto čiastočnej funkcionality serveru JBoss.

Čo sa týka klientov, vytvorili sme prepojenie 2D a 3D klienta prostredníctvom http, zobrazenie editovaného grafu z 2D do 3D prostredníctvom GML, jednoduchú editáciu grafu v 2D, možnosti grafu - collapse, expand, delete, zmenu rozloženia uzlov.

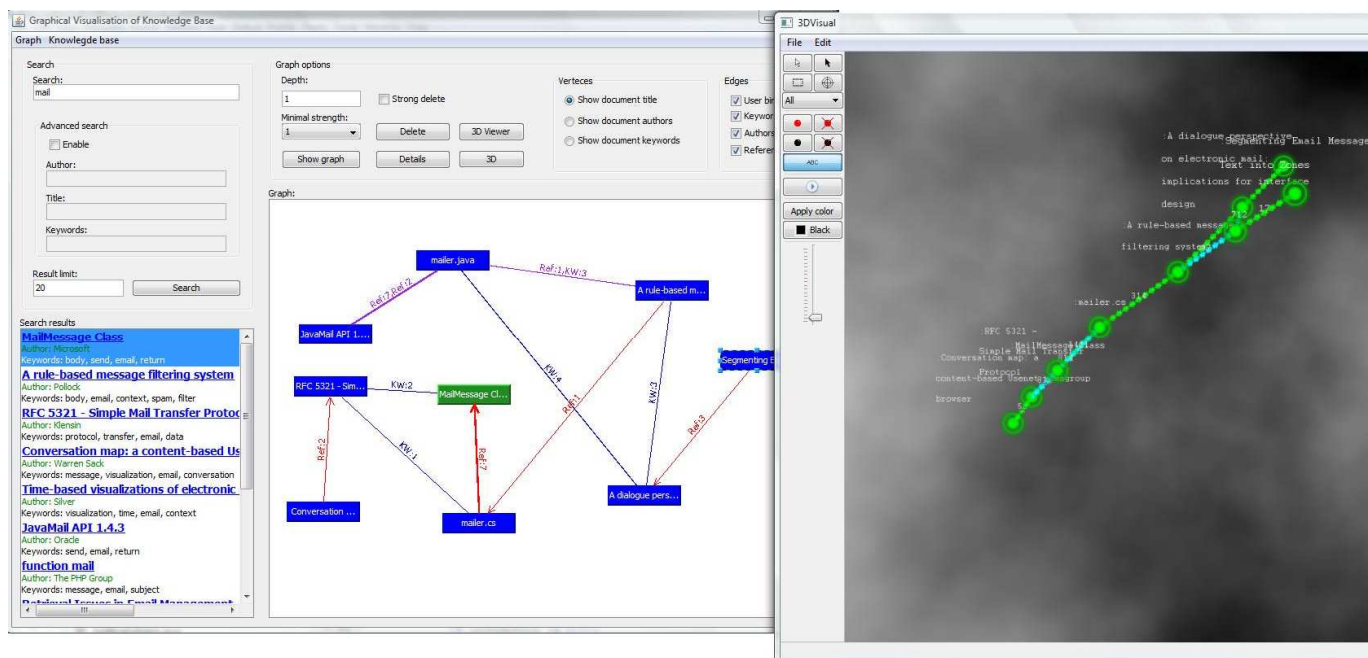


Server predstavuje Java HTTP servlet, beží na aplikačnom serveri Jetty, komunikuje s 2D klientom na báze architektúry REST, s databázou komunikuje prostredníctvom frameworku iBatis.

PostgreSQL databáza využíva dedenie pri vytváraní tabuliek. Urobili sme zmenu v dátovom modeli - väzobnú tabuľka pre kľúčové slová. iBatis nám umožňuje mapovanie dát na objekty - XML mapovacie súbory, konfiguráciu pripojenia prostredníctvom XML.

Tiež sme implementovali reálne získavanie kľúčových slov z dokumentov a informácií z metadát dokumentov.

Na obrázku č. 18 vidíme ukážku používateľských rozhraní oboch klientov.



Obr. 18. Screenshot prototypov 2D a 3D klienta



## Príloha A

### Naplnenie databázy prototypu (tabuľka)

Documents														
id	1	2	3	4	5	6	7	8	9	10	11	12	13	14
type	4	4	4	4	4	4	4	2	2	2	2	1	1	1
addedBy	1	2	3	4	1	2	3	4	1	2	3	4	1	2
fileName	D09-1096.pdf	dredze_ceas05.pdf	p232-pollock.pdf	nssilver-025.pdf	warsac-213.pdf	ajovo-613.pdf	kseadp-916.pdf	jklein-326.pdf	jma143.htm	fmphg.htm	mmmsc.htm	mailer.java	mailer.php	mailer.cs
name	Segmenting Email Message Text into Zones	Reply Expectation Prediction for Email Management	A rule-based message filtering system	Time-based visualizations of electronic mail	Conversation map: a content-based Usenet newsgroup browser	Retrieval Issues in Email Management	A dialogue perspective on electronic mail: implications for interface design	RFC 5321 - Simple Mail Transfer Protocol	JavaMail API 1.4.3	function mail	MailMessage Class	mailer.java	mailer.php	mailer.cs
author	Andrew Lampert	Mark Dredze	Pollock	Silver	Warren Sack	Jovicic	Severinson-Eklundh	Klensin	Oracle		Microsoft	Marek	Juraj	Matej
keywords	email, segmentation, email zones	email, reply, context, email management	spam, context, body	email, visualization, time, context	map, browser, visualization	email, issue, segmentation	vsualization, email, message, design	email, protocol, tranfer, data	email, send, return	email, subject, message	email, send, return, body	send, return, throw, try, catch	mail, message	send, try, catch
Doc Type														
id	1	2	3	4										
type	Source code	Documentation	Task	Article										