VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ A AKTUALIZACE FRAGMENTŮ ANOTACÍ

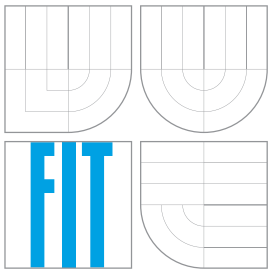
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. LUKÁŠ KUBÍK

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ A AKTUALIZACE FRAGMENTŮ ANOTACÍ

ANNOTATION FRAGMENTS SEARCHING AND UPDATES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. LUKÁŠ KUBÍK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV DYTRYCH
SUPERVISOR

BRNO 2014

Zadání diplomové práce/15631/2013/xkubik22

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2013/2014

Zadání diplomové práce

Řešitel: **Kubík Lukáš, Bc.**
Obor: Informační systémy
Téma: **Vyhledávání a aktualizace fragmentů anotací**
Annotation Fragments Searching and Updates
Kategorie: Web

Pokyny:

1. Seznamte se s jazykem Java, formátem XML a knihovnami pro práci s tímto formátem v jazyce Java.
2. Prostudujte dostupný server pro správu anotací vyvíjený v rámci evropského projektu Decipher a vybrané části tohoto serveru.
3. Navrhněte vylepšení aktualizací pozic anotovaných fragmentů po změně anotovaného dokumentu a detekci míry ovlivnění anotací těmito změnami. Zaměřte se při tom na určení vhodných prahů, kdy již fragment nelze správně nalézt a kdy je anotace změnou zneplatněna. Navrhněte také algoritmus pro určení, zda je vliv na anotace daného dokumentu natolik závažný, že by aktualizace dokumentu neměla být provedena bez interakce s uživatelem.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a srovnejte s alternativními přístupy.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Dytrych Jaroslav, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2013
Datum odevzdání: 28. května 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato diplomová práce analyzuje algoritmy anotačního serveru projektu Decipher pro hledání a aktualizaci fragmentů anotací. Analyzované algoritmy vylepšuje a nahrazuje nově navrženými algoritmy. Součástí projektu je také návrh nového algoritmu pro detekci míry ovlivnění anotací po aktualizaci dokumentu.

Abstract

This master's thesis analyzes annotation server algorithms for searching and updating annotation fragments. The annotation server is a part of the project Decipher. Analyzed algorithms are improved and replaced by newly designed algorithms in this project. A part of this project also designs a new algorithm for measuring how much is an annotation affected after updating the document.

Klíčová slova

Aktualizace fragmentu, fragment, anotace, anotační server, 4A, Decipher, editor anotací, Java Enterprise Edition.

Keywords

Fragment update, fragment, annotation, annotation server, 4A, Decipher, annotation editor, Java Enterprise Edition.

Citace

Lukáš Kubík: Vyhledávání a aktualizace fragmentů anotací, diplomová práce, Brno, FIT VUT v Brně, 2014

Vyhledávání a aktualizace fragmentů anotací

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Dytrycha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Kubík
12. května 2014

Poděkování

Děkuji svému vedoucímu práce Ing. Jaroslavu Dytrychovi za odborný dozor, připomínky a rady v průběhu vypracování mé diplomové práce.

Obsah

1 Úvod **3**

2 Projekt Decipher **4**
2.1 Struktura anotace 4
2.2 Architektura projektu 6
2.3 Anotační server 6

3 Analýza algoritmů vyhledávání a aktualizace fragmentů **9**
3.1 Datová struktura dokumentu, anotace a fragmentu 9
3.2 Aktualizace fragmentů 9
3.3 Metody kontroly a aktualizace fragmentů 11
3.4 Průchod dokumentem 12
3.5 Průchod textovým uzlem 13
3.6 Metody porovnání fragmentů 14
3.7 Analýza skupin vyhledávacích metod 15
3.8 Detekce míry ovlivnění anotací 15
3.9 Ukládání anotací do databáze 16

4 Technologie použité pro vývoj **17**
4.1 Java Enterprise Edition 17
4.2 GlassFish Server Open Source Edition 17
4.3 Java Persistence Api 18
4.4 MySQL 18
4.5 XML 18
4.6 DOM 18
4.7 XPath 18

5 Návrh nových algoritmů vyhledávání a aktualizace fragmentů **20**
5.1 Dvousměrný iterátor 20
5.2 Způsoby použití iterátorů k vyhledávání a aktualizaci fragmentů 21
5.3 Návrh nových metod průchodu textovým uzlem 24
5.4 Vyhodnocení nalezených fragmentů 26
5.5 Skupiny metod vyhledávání a aktualizace fragmentů 28

6 Návrh nového algoritmu detekce míry ovlivnění anotací **30**
6.1 Ohodnocení fragmentu 30
6.2 Ohodnocení anotace 32
6.3 Synchronizace dokumentu 32

7 Návrh vylepšení ukládání anotací do databáze **33**
7.1 Návrh aktualizace položky tmpid a referencí v atributech s odkazem na anotaci 33
7.2 Návrh opravy duplikace fragmentů a špatného uložení specifické kombinace atributů 34
7.3 Návrh opravy změny zanořené anotace 34
7.4 Návrh ukládání nabídek anotací 35

8 Implementace algoritmů **37**
8.1 Implementace vylepšení aktualizace fragmentů 37
8.2 Implementace rozšíření diplomové práce 38

9 Testování úspěšnosti implementovaných algoritmů **40**
9.1 Testování aktualizace fragmentů anotace 40
9.2 Zvolení nejvíce shodného fragmentu vůči původnímu fragmentu 46
9.3 Testování detekce míry ovlivnění anotací 47
9.4 Testování ukládání a aktualizace anotací a nabídek anotací 51

10 Závěr **53**

Literatura **54**

Kapitola 1

Úvod

Cílem této diplomové práce bylo navrhnout, implementovat a otestovat vylepšení algoritmů pro aktualizaci pozic anotovaných fragmentů po změně anotovaného dokumentu a algoritmu pro určení, zda je vliv na anotace daného dokumentu natolik závažný, že by aktualizace dokumentu neměla být provedena bez interakce s uživatelem. Algoritmy jsou součástí 4A serveru projektu Decipher.

Projekt Decipher je stručně rozebrán v kapitole 2. Kapitola je soustředěna na popis zaměření projektu a architektury projektu. Větší pozornost je věnována 4A serveru, jehož součástí jsou vylepšené algoritmy.

Analýzu algoritmů před vylepšením přibližuje kapitola 3. V kapitole je rozvedeno vyhledávání a aktualizace fragmentů po změně textu anotovaného dokumentu. Vyhledávání je rozděleno na průchod uzly dokumentu a na průchod textem uzlu. Ke konci kapitoly jsou popsány porovnávací metody a původní stav algoritmu pro detekci míry ovlivnění anotací.

V kapitole 4 se nachází použité technologie pro vývoj projektu. Kapitola technologie stručně definuje a zmiňuje jejich aktuální verze.

Návrh vylepšených a nových algoritmů pro vyhledávání a aktualizaci fragmentů popisuje kapitola 5. Kapitola se zaměřuje na způsoby využití iterátorů, navrhuje nový dvousměrný iterátor dokumentem, vylepšení dvou metod průchodu uzlem dokumentu, nové algoritmy průchodu uzlem dokumentu a ke konci navrhuje metodu pro vyhodnocení nalezených fragmentů.

Návrh algoritmu detekce míry ovlivnění anotací je popsán kapitole 6. Návrh se zaměřuje na ohodnocení ovlivnění fragmentů, jednotlivých anotací, veškerých anotací v dokumentu a reakcí algoritmu na výsledné ohodnocení.

Diplomová práce byla rozšířena o vylepšení ukládání anotací a ukládání nabídek anotací. Návrh vylepšení je uveden v kapitole 7.

Implementace navržených algoritmů je stručně shrnuta v kapitole 8. Kapitola je rozdělena na část implementace vylepšení aktualizace fragmentů a na část s implementací navrženého rozšíření.

Testy implementovaných algoritmů a jejich nastavení jsou provedeny v kapitole 9. Testování kontroluje zejména úspěšnost implementovaných metod při aktualizaci fragmentů.

Závěr je uveden v kapitole 10. Závěr zhodnocuje výsledky analýzy a navržené algoritmy.

Kapitola 2

Projekt Decipher

Kapitola je zaměřena na popis cílů projektu. Jakým způsobem těchto cílů projekt dosahuje, jak je postavena architektura projektu a kterou část tato diplomová práce vylepšuje.

Decipher je projekt pro Evropskou Unii, na kterém se spolu s VUT Brno [35] podílely další vysoké školy: Dublin Institute of Technology (Irsko) [11] a The Open University (Anglie) [33]. Mimo akademickou sféru jej vyvíjela také firma System Simulation [32]. Koncoví uživatelé projektu jsou evropská muzea, galerie a jejich zaměstnanci, kteří budou schopni pomocí projektu efektivněji organizovat díla a výstavy. Vzhledem k zaměření projektu byli hlavními partery National Gallery of Ireland [19] a Irish Museum of Modern Art [16].

Cílem projektu [9] je vyvinout nástroj, který by ukládal a prezentoval objekty kulturního dědictví z různých zdrojů. Uživatelům by to umožnilo vytvořit si vlastní kolekce příběhů s doplňujícími informacemi, které by projekt propojil do strukturovaného celku.

Jedním z prostředků, kterými projekt dosahuje cíle, je anotování textu. Anotace je dodatečná informace vztahující se k části textu. Ukázka anotace je znázorněna na obrázku 2.1. Anotován je vyznačený text *James Coleman*. Ostatní vyznačené části textu jsou součástí nabídek anotací.

2.1 Struktura anotace

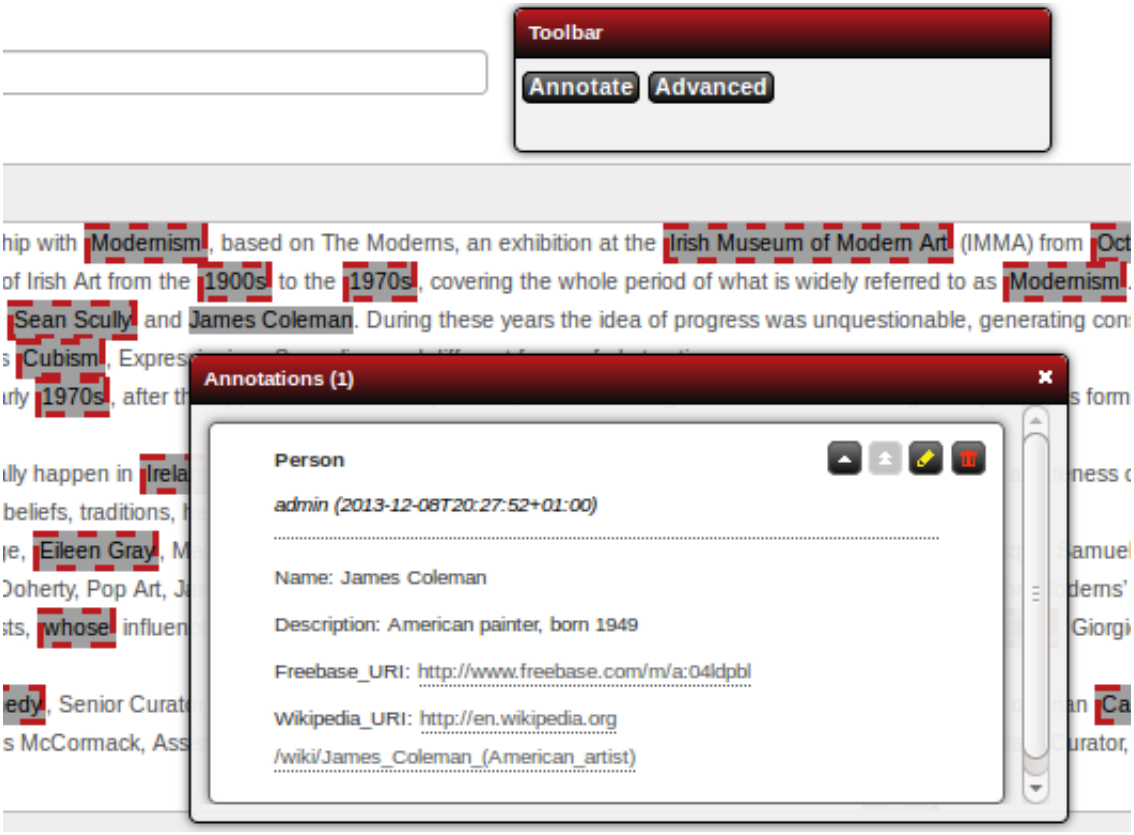
Anotace má svůj popis, typ a může mít fragmenty a atributy. Popis je krátká doplňující textová informace. Typy, fragmenty a atributy rozvádí obsah této sekce. Uživatel v editoru (editor bude popsán v další sekci) označuje fragmenty anotace a k nim následně přidává atributy. Zjednodušený diagram tříd pro anotaci lze vidět na obrázku 3.1 v další kapitole.

2.1.1 Fragmenty

Fragment anotace je vybraný text, který je anotován. Tato diplomová práce je zaměřena zejména na fragmenty a na jejich aktualizaci po změně dokumentu.

2.1.2 Typ anotace

Každá anotace má svůj typ, který dělí anotace na kategorie. Typ může obsahovat atributy. Atributy jsou stejné jako atributy anotace, ale neobsahují hodnoty. Pouze definují volitelné a povinné atributy anotace, která je daného typu.



Obrázek 2.1: Ukázka anotace v anotačním editoru

2.1.3 Atributy anotace

Ukázka atributů anotace je uvedena na obrázku 2.1 v okně s názvem *Annotations(1)*. Atributy dělíme na dva základní typy:

Jednoduché atributy

Obsahují hodnotu, která je uložena přímo v atributu, a doplňují informační hodnotu anotace. Jednoduché atributy se ještě dále dělí na následující podtypy:

- **Binární** – binární soubor
- **Celočíselný** – celočíselná hodnota
- **Časová délka** – délka trvání
- **Časový** – datum a čas
- **Desetinný** – 32 bitová desetinná hodnota
- **Entita** – jednoznačně identifikovatelná entita z kontrolovaného slovníku
- **Obrázek** – URI obrázku
- **Pravdivostní** – pravdivostní hodnota

- **Řetězcový** – krátká či dlouhá textová poznámka
- **Souřadnice** – GPS souřadnice
- **URI** – odkaz

Strukturované atributy

Typy strukturovaných atributů odpovídají typům anotací. Přidáním strukturovaného atributu k anotaci se vytváří odkazy na další anotace, případně zanořené anotace. Pomocí strukturovaných atributů můžeme v textu vytvořit vazby.

2.2 Architektura projektu

Projekt Decipher se skládá ze tří částí [31], které jsou popsané v této sekci. Zjednodušený diagram tříd obsahující balíčky projektu lze vidět na obrázku 2.2.

2.2.1 StoryScope

StoryScope je interaktivní grafické prostředí projektu Decipher, které je postaveno na redakčním systému Drupal. Zajišťuje klientskou část, kterou uživatelé využívají pro vytváření, uchování a editaci objektů kulturního dědictví, vyprávění, příběhů, obrázků, souborů a dalšího obsahu. Při vytváření a editaci textových dat můžeme text sémanticky anotovat v anotačním editoru, který je součástí StoryScope. StoryScope dále umožňuje ukládat časové události a na základě již uložených událostí a uloženého obsahu může navrhovat nové události.

2.2.2 Decipher Aggregator

Decipher Aggregator slouží pro ukládání dlouhodobě uložených dat. Uložená data zasílá serveru SEC, který je doplní o metadata a navrátí Aggregatoru. Aggregator poté data indexuje.

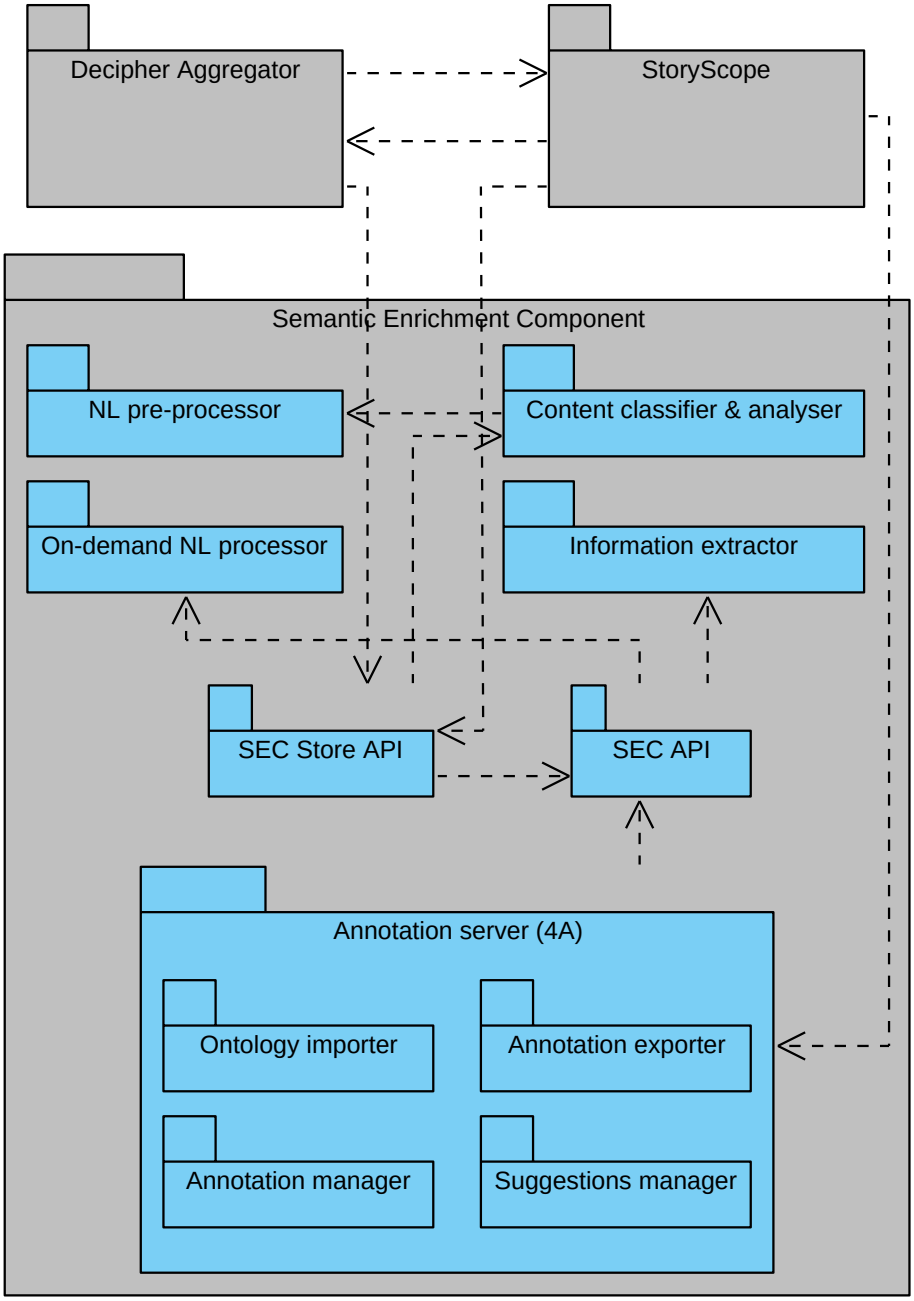
Klient StoryScope se připojuje k serveru Decipher Aggregator. Klient může vyhledávat v indexovaných datech serveru. Server také dlouhodobě uchovává příběhy a vyprávění ze všech klientů používajících daný server a opět je indexuje.

2.2.3 Semantic Enrichment Component

Semantic Enrichment Component (dále jen SEC) zpracovává nestrukturovaný text přijatý od StoryScope. SEC pro zaslaný text zasílá StoryScope vygenerované nabídky anotací pro zvolenou část textu. Uživatel může tyto nabídky ve StoryScope buď potvrdit, čímž se z nich stane plnohodnotná anotace, nebo odmítnout. Poté se mu již znovu nezobrazí. SEC je složen z více komponent. Tato diplomová práce rozšiřuje komponentu anotačního serveru.

2.3 Anotační server

Zkráceně se nazývá 4A server, což je zkratka pro čtyři A: Annotations Anywhere, Annotations Anytime, v překladu anotace kdekoliv a kdykoliv. 4A server je založen na 4A Frameworku [1], je součástí SEC a je postavený na Java Enterprise Edition (dále jen Java EE)

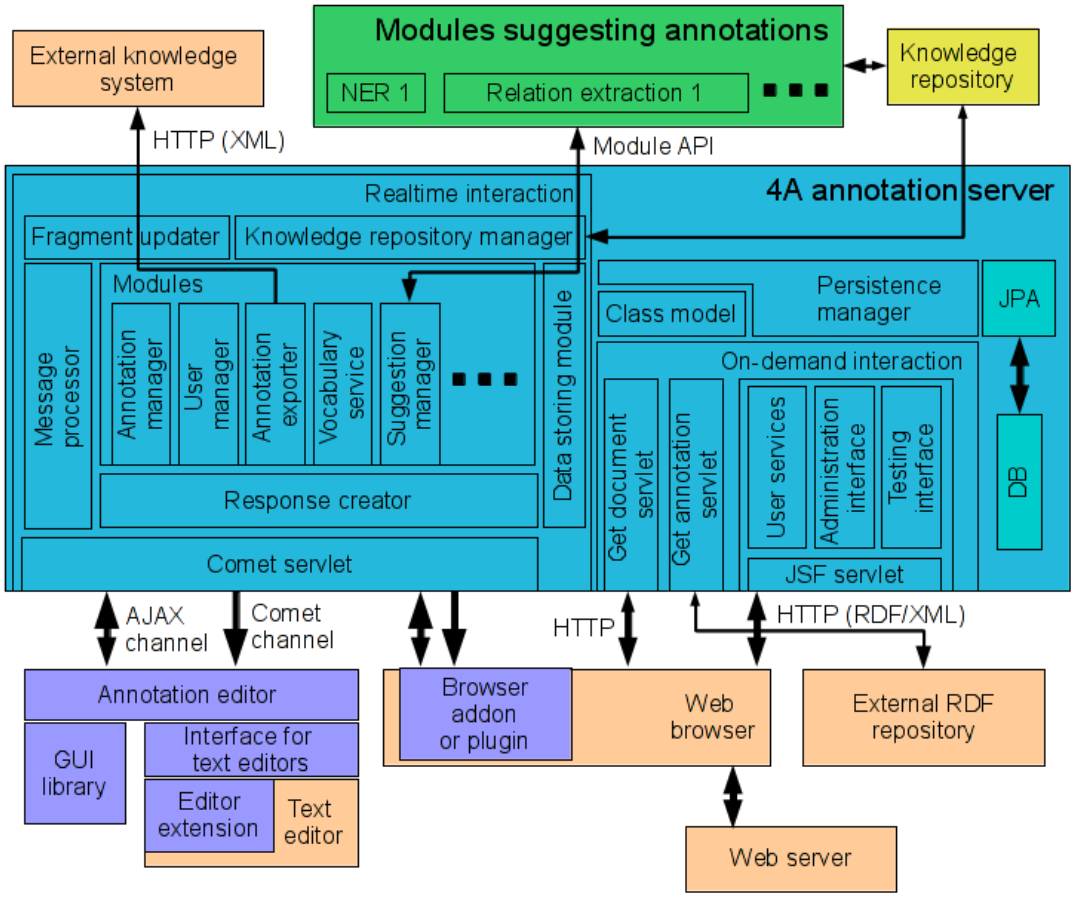


Obrázek 2.2: Zjednodušený diagram tříd projektu [31]

serveru [22], který je funkčním jádrem pro zpracování anotovaných dokumentů a jejich anotací. Kopie dokumentů a anotace ukládá do databáze, ve které spravuje také typy anotací, oprávněné uživatele, uživatelské skupiny a další. Moduly serveru jsou uvedeny na obrázku 2.3.

2.3.1 Komunikace klienta s 4A Serverem

Klientská část StoryScope komunikuje se serverem protokolem HTTP a předává mu zprávy v jazyce XML. Server zprávy přijímá a odpovídá na ně v objektu třídy `Comet servlet`.



Obrázek 2.3: Struktura 4A serveru [2]

Přijaté zprávy jsou zaslány ke zpracování do objektu třídy `Message Processor`. Ten jednotlivé zprávy zpracuje odpovídajícími metodami a z každé zprávy vytvoří příslušný požadavek na server, který je reprezentován datovou strukturou `RequestInfo`. Tato struktura je následně předána objektu třídy `Response creator`, který volá další moduly serveru určené ke zpracování požadavků. Mezi těmito moduly se nachází i modul `Annotation manager` implementovaný v třídě `CoreFuncModule`, který spravuje anotace a jejich atributy. K tomu využívá tříd v balíčku `Fragment updater`. Třídy řeší aktualizaci pozic fragmentů v jednotlivých anotacích. Třídu `CoreFuncModule` a třídy balíčku `Fragment updater` vylepšuji v této diplomové práci. Po průchodu všemi moduly zpracovávajícími požadavky je nutné uložit změny v dokumentu a jeho anotacích. To je řešeno v `Data storing` modulu, který uloží změny do databáze MySQL s využitím abstrakce, kterou poskytuje `Persitence manager`. `Persitence manager` je implementován v třídě `PersistM`. Do databáze MySQL se přistupuje pomocí frameworku `Java Persistence Api` a přístup zapouzdřuje třída `PersistM`. Po uložení dat se ještě znovu volají všechny moduly zpracovávající požadavek a v průběhu volání je generována odpověď pro klienta v objektu třídy `Response creator`. Odpověď je složena z návratových hodnot volaných modulů.

Kapitola 3

Analýza algoritmů vyhledávání a aktualizace fragmentů

Následující kapitola pojednává o původním stavu algoritmů vyhledávání a aktualizace fragmentů a také o algoritmu určení míry ovlivnění anotací při aktualizaci dokumentu. Analyzuje nedostatky algoritmů a určuje, v kterých místech budou v této diplomové práci vylepšeny.

3.1 Datová struktura dokumentu, anotace a fragmentu

Server přistupuje k databázi pomocí `Java Persistence Api` a ke všem tabulkám databáze jsou vytvořeny odpovídající třídy.

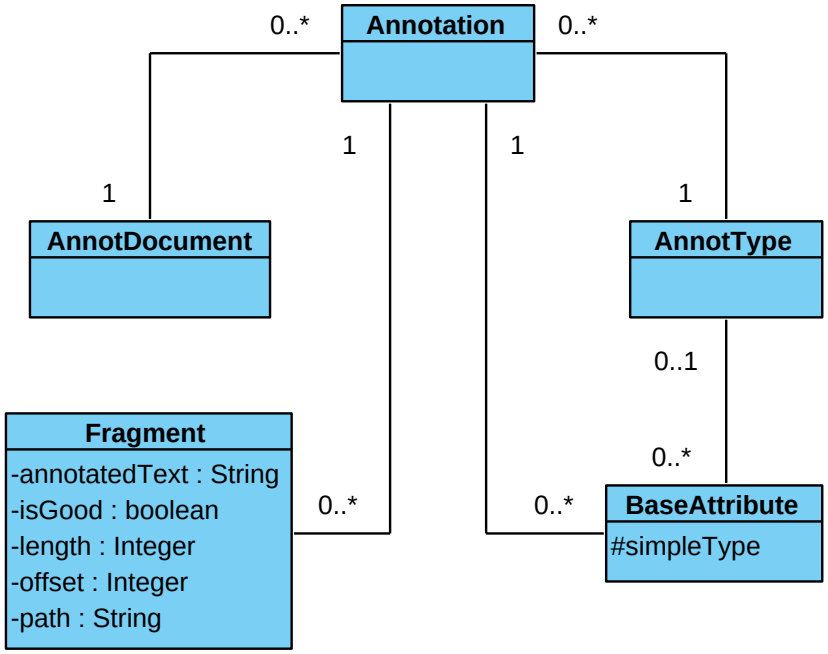
Pokud anotujeme dokument, jeho kopie je uložena do databáze serveru. Na serveru je dokument uložen ve formátu XML. V dokumentu je možné iterovat přes uzly XML pomocí iterátorů inicializovaných uzlem adresovaným jazykem XPath.

Při vytvoření anotace se anotace uloží do databáze a vztahuje se k určitému dokumentu. Fragmenty anotace jsou také uloženy v databázi a mají vztah k jedné z anotací. Anotace může mít jeden nebo více fragmentů. Více fragmentů má anotace typicky při anotování přes více uzlů dokumentu XML. Např. přes více odstavců. Zjednodušený diagram tříd pro anotaci se nachází na obrázku 3.1. Pro aktualizaci fragmentů jsou důležité následující vlastnosti fragmentu:

- **frPath** – XPath cesta k fragmentu v rámci anotovaného dokumentu
- **frOffset** – offset od začátku uzlu dokumentu, ve kterém se fragment nachází
- **frLength** – délka anotovaného textu
- **annotatedText** – anotovaný text
- **isGood** – pravdivostní hodnota, jestli je fragment v pořádku a součástí dokumentu

3.2 Aktualizace fragmentů

V této sekci je rozebrán postup vytvoření anotace a následná aktualizace fragmentů anotace při aktualizaci dokumentu, která probíhá po změně textu.



Obrázek 3.1: Zjednodušený diagram tříd pro anotaci

Kromě přidání, změny a odstranění anotace je v třídě `CoreFuncModule` implementována také synchronizace dokumentu pro přihlášeného uživatele v aktuálním sezení. V neposlední řadě také spravuje a udržuje závislosti mezi anotacemi kontrolou a aktualizací strukturovaných atributů.

3.2.1 Připojení k serveru

Uživatel otevře v anotačním editoru dokument, který chce anotovat, a připojí se k serveru. Pokud uživatel anotuje dokument poprvé, server si k němu vytvoří příslušnou datovou strukturu a uloží ji do databáze. V případě, že byl dokument již anotován, dojde k synchronizaci dokumentu. Při synchronizaci se aktualizuje dokument a jeho anotace. Po synchronizaci lze anotovat dokument.

3.2.2 Synchronizace dokumentu

Změna textu dokumentu v klientovi, který není připojen k serveru, může ovlivnit anotace a pozice jejich fragmentů. Po změně textu, která ovlivní původní anotace, je nutné fragmenty a text dokumentu na serveru porovnat s novou verzí od klienta, vyhodnotit míru ovlivnění anotací a případně aktualizovat. Automatická aktualizace bez interakce uživatele je povolena do ovlivnění anotací, které nepřesáhne práh. Pokud ovlivnění anotací přesáhne práh, uživatel má na výběr. Buď změny v dokumentu zahodí a použije verzi dokumentu uloženou na serveru, nebo se anotace ve změněném dokumentu nechají automaticky aktualizovat, i když se změní více anotací než udává práh. Porovnání a aktualizace probíhá po příchodu zprávy od klienta s požadavkem na synchronizaci dokumentu. Po úspěšném zpracování zprávy server zašle zpět aktualizované anotace klientovi. Aktualizace fragmentů anotací probíhá v třídách balíčku `FragmentUpdater`. Jakým způsobem probíhá, je popsáno v sekci 3.3.

3.2.3 Změna anotací po připojení klienta k serveru

Textová změna dokumentu po připojeni klienta k serveru vyvolá vytvoření zprávy XML se změnou, která je zaslána na server. Server vyhodnotí ovlivněnou část dokumentu, pro kterou musí zaktualizovat fragmenty anotací. Po aktualizaci fragmentů jsou nové fragmenty poslány zpět klientovi, který změnu fragmentů v dokumentu reflektuje. Rozdíl oproti změně bez připojení je, že se nevyhodnocuje míra ovlivnění anotací. Anotace se ihned automaticky aktualizují. Ve většině případů také nejde o změnu celého dokumentu, ale jen o změnu některé jeho části.

3.2.4 Editace anotací

Při každém uživatelském vytvoření/změně/odstranění anotace klient vytvoří zprávu XML s požadavky na server. Pokud jde o vytvoření, nebo změnu anotace, server znovu zkontroluje, jestli klient vytvořil/změnil fragmenty anotace správně a odpovídají dokumentu, který je anotován. Při požadavku na odstranění anotace se kontroluje existence mazané anotace. Pokud je vše v pořádku, server požadavek potvrdí po zpracování ostatních požadavků. Pokud není, server odpovídá chybovou zprávou.

3.3 Metody kontroly a aktualizace fragmentů

V průběhu aktualizace fragmentů se kontrolují vlastnosti uložených fragmentů na serveru. Při iteraci přes uložené fragmenty se u každého fragmentu kontroluje pozice fragmentu v dokumentu a jestli nebyl změněn jeho text. V případě změn se fragment aktualizuje.

Kontrola a aktualizace fragmentů provádí průchod uzly dokumentu XML pomocí iterátorů dokumentu, dokud nenajde původní fragment nebo vhodný fragment k aktualizaci. Iterátory se nacházejí v balíčku `fragmentUpdater.nodeIterators`. Iterátor je před průchodem inicializován počátečním uzlem adresovaným jazykem XPath. Více o XPath pojednává sekce 4.7. V již konkrétním uzlu se fragment vyhledává metodou na průchod textovým uzlem, což jsou metody třídy `Comparator`. Při průchodu textem uzlu se dále používají metody na porovnání řetězce původního fragmentu a podřetězce textu uzlu. Porovnávací metody lze je nalézt v balíčku `fragmentUpdater.compareMethods`.

Vyhledání, kontrolu a případnou aktualizaci fragmentů implementuje třída `Matcher`, které lze nastavit iterátor, způsob průchodu textovým uzlem a porovnávací metodu. Vyhledání a kontrola fragmentů je rozdělena na několik stupňů. Každý stupeň je reprezentován jedním objektem třídy `Matcher` a prioritou. Největší prioritu má hledání fragmentů s nejvíce přesnou shodou k původnímu fragmentu. Naopak nejmenší prioritu má hledání s největší tolerancí na změny fragmentu. Stupně hledání a kontroly jsou zapouzdřeny ve třídě `MatcherProvider` a v další části textu se o nich píše jako o skupinách vyhledávacích metod.

Aktuálně se v projektu fragmenty vyhledávají a aktualizují dvojím způsobem. Buď se prochází dokument, dokud se nenarazí na dostatečně podobný text vůči původnímu fragmentu, což provádí metoda `match()` třídy `MatcherProvider`. Případně se vyhledají všechny řetězce, které jsou dostatečně podobné, a vybere se z nich nejvíce podobný originálnímu textu a od tohoto textu nejméně vzdálený. Tento způsob implementuje metoda `matchAll()` třídy `MatcherProvider`. Nalezeným textem je poté fragment aktualizován. Metody třídy `MatcherProvider` bude třeba rozšířit o další způsoby vyhledání a aktualizace fragmentů v dokumentu.

3.4 Průchod dokumentem

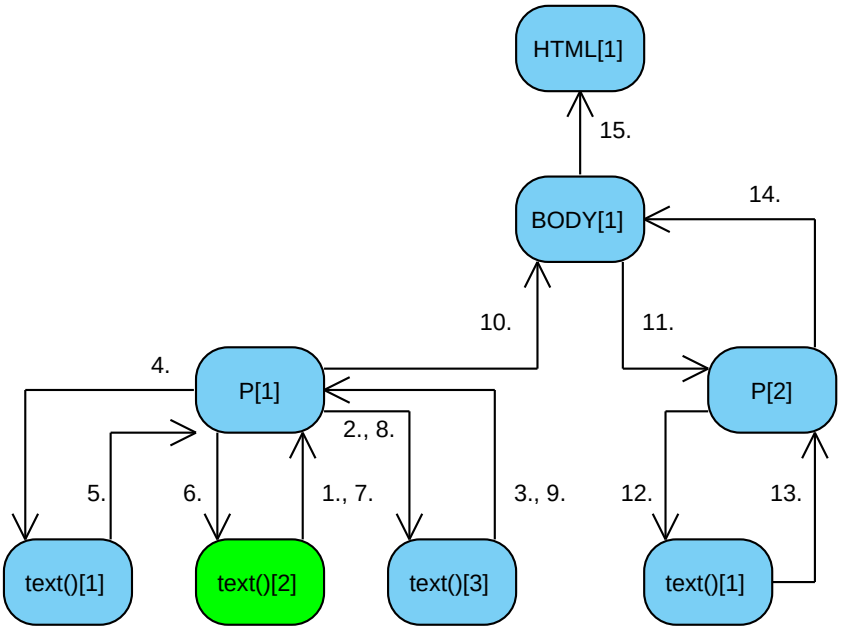
Před samotnou kontrolou a případnou aktualizací fragmentů je třeba projít dokumentem a dostat se k uzlům, ve kterých bude kontrola provedena. K tomu slouží tři iterátory popsané v třídách balíčku `fragmentUpdater.nodeIterators`. Jejich základ je popsán abstraktní třídou `NodeIterator`. Obsahuje pouze dvě metody. Jedna je pro inicializaci iterátoru a druhá navrací další uzel. Při tomto způsobu průchodu se lze dostat i k netextovým uzlům. Je tedy třeba kontrolovat, jestli je navrácený uzel textový, nebo uzel určující strukturu dokumentu.

3.4.1 Iterátor XPath

Iterátor je založený na vyhodnocení uzlů, které se nachází na původní XPath. XPath, kde se nacházel původní fragment, je vyhodnocen a do seznamu jsou navráceny všechny uzly nového dokumentu, jejichž XPath se shoduje s původním XPath. Tento seznam je pak sekvenčně procházen.

3.4.2 Rekurzivní iterátor

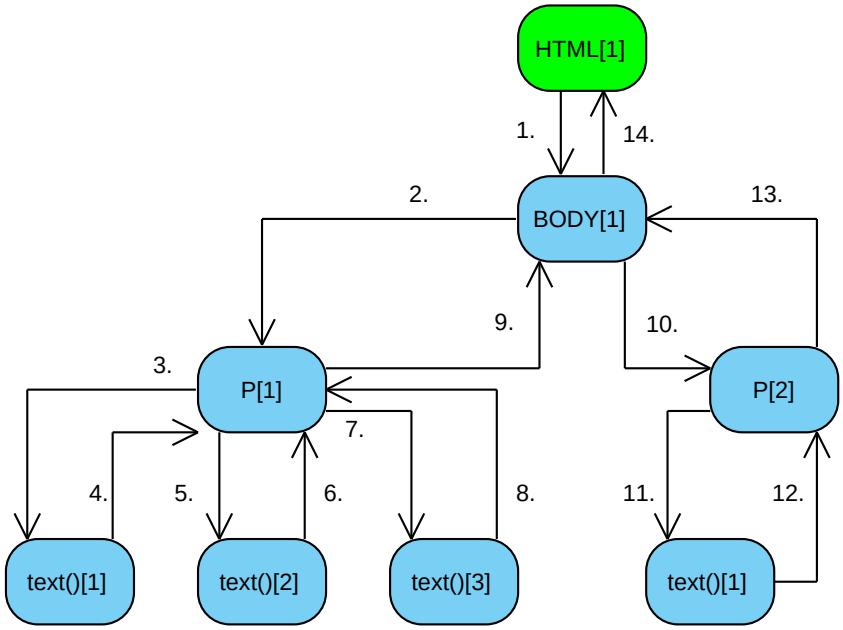
Rekurzivní iterátor funguje na principu průchodu pre-order upraveného pro n-ární strom a pro možnost začít v kterémkoliv uzlu. Prvotní uzel je nastaven vyhodnocením XPath. Uzel, který jako první vyhovuje původní XPath, je nastaven jako počáteční, od kterého se bude vyhodnocovat dále. V případě, že je to kořenový uzel, jedná se o jednoduchý pre-order průchod jako v případě průchodu sekvenčního iterátoru. Pokud je ovšem počáteční uzel zvolen kdekoliv jinde, je třeba projít celý strom od tohoto uzlu. V tuto chvíli je to vyřešeno druhým průchodem podstromu s kořenem zvoleným jako otcovský uzel počátečního uzlu. Průchod pro fragment s původním XPath: `/HTML[1]/BODY[1]/P[1]/text()[2]` je znázorněn na obrázku 3.2. V návrhu by se tento duplicitní průchod měl odstranit.



Obrázek 3.2: Průchod rekurzivního iterátoru

3.4.3 Sekvenční iterátor

Sekvenční iterátor je rozšířením třídy `RecursiveNodeIterator`, procházení dalších uzlů funguje stejně, ale rozdíl je v inicializaci. Sekvenční iterátor je inicializován na kořenový uzel dokumentu. Projde ho celý od začátku do konce. Postup průchodu je znázorněn na obrázku 3.3.



Obrázek 3.3: Průchod sekvenčního iterátoru

3.5 Průchod textovým uzlem

Kontrola fragmentů se provádí v průběhu iterace dokumentem při načtení příslušného textového uzlu dokumentu. Při kontrole se prochází text uzlu a vytváří se řetězce k porovnání s původním fragmentem. Porovnáním se hledá původní fragment a v případě neshody fragment k aktualizaci. Řetězce se vytváří v třídě `Comparator` a jsou tři různé metody k vytvoření řetězce. Statická metoda se používá pro zjištění, jestli fragment zůstal na svém původním místě. Porovnává pouze jeden podřetězec textového uzlu. Metody procházející uzel po slovech či znacích jsou určeny k vyhledání odsunutého či změněného fragmentu. Prochází celý textový uzel a vytváří z něj podřetězce, které předávají porovnávací metodě. První řetězec, který porovnávací metoda vyhodnotí jako dostatečně podobný původnímu fragmentu, je navrácen jako nový fragment k aktualizaci.

3.5.1 Statický

Textový řetězec je vytvořen jako podřetězec textového uzlu na offsetu původního fragmentu s původní délkou fragmentu.

3.5.2 Po slovech

Textový uzel je rozdělen na slova. Ze všech slov uzlu se postupně vytváří řetězce se stejným počtem slov jako měl původní fragment. Vytvořené řetězce jsou předány porovnávací metodě.

3.5.3 Po znacích

Metoda prochází textový uzel po znacích a vytváří z něj podřetězce o délce řetězce původního fragmentu. Počáteční znak podřetězce je na počátku shodný s počátečním znakem textového uzlu a při každém vytvoření nového podřetězce je jeho index v uzlu inkrementován, dokud je možné vytvořit řetězec o dostatečné délce. Podřetězce metoda předává porovnávací metodě.

3.6 Metody porovnání fragmentů

K nalezení původního fragmentu nebo k aktualizaci fragmentu se používají porovnávací metody. Tato sekce rozvádí typy implementovaných metod a analýzu jejich použitelnosti.

3.6.1 Přesná shoda

Porovnávací metoda přesné shody je založena na porovnání řetězců znak po znaku. Používá se pro ověření, že se fragment nezměnil. Jeden rozdílný znak je vyhodnocen jako neshoda.

3.6.2 Levenshteinova metoda

Levenshteinova metoda je založena na Levenshteinově vzdálenosti dvou řetězců [14], [17]. Matematicky je Levenshteinova vzdálenost funkce $lev_{a,b}(|a|, |b|)$, kde

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{pokud } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{a_i \neq b_j} \end{cases} & \text{jinak.} \end{cases}$$

$1_{(a_i \neq b_j)}$ indikuje, že funkce je rovna 0, pokud $a_i = b_j$, jinak se rovná 1.

Levenshteinova vzdálenost je vždy minimálně rozdíl délky dvou řetězců. Největší vzdálenost je rovna délce delšího z řetězců. Nultá vzdálenost je pouze tehdy, když jsou oba řetězce stejné. Obecně je vzdálenost rovna počtu rozdílných znaků. Například slova *example* a *esampre* mají vzdálenost 2, protože se liší ve dvou písmenech, ale mají stejnou délku.

Levenshteinova metoda rozšiřuje Levenshteinovu vzdálenost o možnost specifikovat procentuální podobnost nového fragmentu vůči původnímu fragmentu, aby bylo slovo navráceno jako dostatečně shodné.

3.6.3 Metoda Soundex

Je foneticky založený algoritmus k indexaci jmen podle jejich anglické výslovnosti [30]. Kóduje homofony[1] do stejného kódu, aby se shodovaly nehledě na výslovnost. Algoritmus je určen pro anglickou abecedu. Při použití jiného jazyka textu bude problém s kódováním speciálních znaků jazyka a s jinak založenou výslovností daného jazyka.

[1] Zvukově shodná slova, ale graficky odlišná. [3]

3.6.4 Metoda Metaphone

Metoda Metaphone je založena na podobném principu jako metoda Soundex [12]. Metoda počítá s rozdílnostmi jazyků, ale primárně je zaměřena na anglická slova. Nejvíce optimalizovaná implementace metody je placená a má název Metaphone 3 [18]. Ve 4A serveru byla dostupná implementace v první verzi s názvem Metaphone, která je volně šiřitelná pod licencí Apache License 2.0.

3.7 Analýza skupin vyhledávacích metod

Původní systém vyhledávání a aktualizace fragmentů je založen na dvou skupinách metod pro vyhledání a aktualizaci fragmentů (na dvou objektech třídy `MatcherProvider`). První objekt je určen pro vyhledání přesně shody bez tolerance změny pozice či textu fragmentu. V případě neúspěchu prvního objektu je druhý objekt určen pro vyhledání co nejpřesnější náhrady za změněný fragment s různými iterátory a porovnávacími metodami.

První skupina je zaměřena pouze na kontrolu původní pozice a textu fragmentu. Tato diplomová práce ji neupravuje a ponechává její použití ve stávající implementaci.

Druhá skupina obsahuje čtyři vyhledávací metody, jejichž výběr je třeba vylepšit. První metoda je kopií jediné metody v první skupině. Při nenalezení fragmentu první skupinou není třeba znovu používat stejnou metodu, která by fragment opět nenalezla. Její použití je tedy duplicitní.

Druhá metoda používá Rekurzivní iterátor, průchod uzlem po znacích a porovnávací metodu s přesnou shodou. Metoda je použita pro vyhledání odsunutého fragmentu. Rekurzivní iterátor a průchod po znacích v jednom směru není efektivní a v případě zrušení původního textového uzlu již nelze iterátor využít, protože jej nelze inicializovat.

Třetí metoda je inicializována pro použití iterátoru XPath, průchodu textovým uzlem po slovech a Levenshteinovy porovnávací metody s 60% shodou s původním fragmentem. Metoda je využita pro nalezení změněných a případně odsunutých fragmentů. Iterátor XPath prohledává pouze původní textový uzel. V případě odstranění uzlu jej opět nelze inicializovat. Metoda průchodu po slovech nebere v úvahu, že počet slov původního fragmentu lze zvýšit či snížit.

Čtvrtá metoda využívá sekvenční iterátor, statický průchod textovým uzlem a porovnávací metodu Metaphone. Tato metoda by měla najít specifické změny fragmentu, kdy se nezmění fonetická skladba slov. Sekvenční iterátor v kombinaci se statickým průchodem porovná maximálně jeden řetězec v textovém uzlu na původním offsetu fragmentu. V případě odsunutí fragmentu o jeden znak v rámci uzlu již nelze najít původní fragment.

Původní systém nevrací dobré výsledky a již jen o znak odsunutý fragment se zaměněným jedním znakem systém ve většině případů nenajde. Způsobeno je to zvolenou kombinací iterátorů, způsobů průchodu textovým uzlem, porovnávacích metod a nedostatky těchto algoritmů.

3.8 Detekce míry ovlivnění anotací

Algoritmus detekce, zda je vliv na anotace daného dokumentu natolik závažný, že by synchronizace dokumentu neměla být provedena bez interakce s uživatelem, byl prozatím v projektu omezen na kontrolu existence původních čí aktualizovaných fragmentů. Dokument se neaktualizoval pouze v případě, že některý z fragmentů nebyl nalezen. Pokud byl nalezen původní fragment, nebo byl fragment aktualizován nově nalezeným fragmentem, doku-

ment se aktualizoval bez jakéhokoliv upozornění a určení míry ovlivnění anotací. Tuto část projektu je třeba změnit a vylepšit.

Nový algoritmus bude při aktualizaci dokumentu ověřovat míru ovlivnění všech anotací. Drobné změny v anotacích, které lze snadno automaticky opravit, by se měly aktualizovat automaticky. Větší změny, jako je změna ve více anotacích, případně celé přepsání anotací, budou při aktualizaci potřebovat interakci uživatele.

3.9 Ukládání anotací do databáze

V rámci analýzy projektu bylo zjištěno, že ukládání anotací do databáze se nechová ve všech případech korektně. Bylo to způsobenou chybou na serveru a nedostatky algoritmů pro ukládání anotací do databáze. Analýzu jsem proto rozšířil o ukládání nových a aktualizovaných anotací.

Ukládání anotací může být zavoláno při dvou událostech. První událost je uložení změn po přijetí příslušného požadavku od klienta. Např. při požadavku na vytvoření, nebo změnu anotace. Po vyřízení požadavku se kontrolují a aktualizují fragmenty anotací. Při nalezení změn fragmentů nastává druhá událost, kdy jsou anotace zaktualizovány v databázi. Ukládání a aktualizaci anotací implementuje třída `Persister`, která má několik nedostatků.

Při ukládání nových a změněných anotací s atributy, které jsou typu odkaz na anotaci, se odkaz po uložení odkazované anotace neaktualizoval. Odkaz je třeba zaktualizovat, protože v některých atributech typu odkaz na anotaci může být použit identifikátor `tmpid` místo reference na objekt odkazované anotace. Vlastnost anotace `tmpid` je jednoznačný identifikátor nabídky anotace a je pomocí něj odkazována anotace při ukládání nového atributu typu odkaz na anotaci. Nejprve je nutné uložit anotaci referovanou přes identifikátor `tmpid`. Poté je nutné projít všechny doposud neuložené anotace a nahradit všechny atributy s odkazem na anotaci, kde se vyskytuje identifikátor `tmpid` uložené anotace, za referenci na objekt odkazované anotace. Teprve poté se uloží původní anotace s atributem odkaz na anotaci.

Vlastnost anotace `tmpid` není uložena do databáze. Po uložení anotace do databáze se tato vlastnost vymaže a je nutné ji znovu vyplnit, aby mohla být ve druhé fázi zpracování požadavku využita moduly.

Výše zmíněné postupy nebyly na serveru vyřešeny. Kromě toho se i špatně aktualizovaly fragmenty a atributy anotace při změně anotace. Zmíněné nedostatky jsem se rozhodl vyřešit v rámci rozšíření této diplomové práce.

Kapitola 4

Technologie použité pro vývoj

Obsah následující kapitoly pojednává o technologiích použitých na vývoji této diplomové práce. Anotační server byl od počátku stavěn na platformě Java Enterprise Edition s aplikačním serverem GlassFish Server Open Source Edition. V anotačním serveru byla použita databáze MySQL, do které se přistupuje pomocí frameworku Java Persistence Api. Pro komunikační zprávy používá anotační server formát XML. Pro přístup k uzlům anotovaných dokumentů používá jazyk XPath. Bylo proto nutné provést studium výše zmíněných technologií.

4.1 Java Enterprise Edition

Java Enterprise Edition (Java EE) rozšiřuje Java Standard Edition (Java SE). Java SE je objektově orientovaný jazyk od firmy Oracle pro všeobecné použití. Zdrojové kódy jsou překládány do mezikódu bytecode. Bytecode je sada instrukcí, které interpretuje virtuální stroj Java Virtual Machine. Vzhledem k použití virtuálního stroje je Java nezávislá na platformě. Správu paměti zajišťuje Garbage Collector, který automaticky uvolňuje alokované zdroje. Java EE rozšiřuje Java SE a je určena pro využití na serveru. Oproti Java SE obsahuje Java EE navíc podporu pro práci s webovými aplikacemi, API pro objektově-relační mapování a distribuované a vícevrstvé architektury. Návrh platformy Java EE je z velké části založen na modulárních komponentách běžících na aplikačním serveru. Anotační server je soubor komponent běžících na aplikačním serveru GlassFish Server Open Source Edition. V projektu Decipher je použita Java SE ve verzi 1.7.0_25. Aktuální verze Javy SE 7 je 1.7.0_45 [24]. Při studiu jazyka Java bylo čerpáno z knih [13], [27] a [29].

4.2 GlassFish Server Open Source Edition

GlassFish Server Open Source Edition je aplikační server pro platformu Java EE, jehož vývoj začala společnost Sun Microsystems a později převzala společnost Oracle Corporation [34]. GlassFish je referenční implementace Java EE a podporuje technologie: servlets, Java Persistence Api, JavaServer Faces[1] a další [21]. V projektu Decipher je použit GlassFish server ve verzi 3.1.2.2. Aktuální verze GlassFish serveru je 4.0 [20].

[1] JavaServer Faces je MVC framework použitý pro vývoj administračního rozhraní serveru.

4.3 Java Persistence Api

Java Persistence application programming interface (API) je rozhraní pro Java SE a pro Java EE, které popisuje způsob práce s objektově-relačními daty v aplikacích v jazyce Java [23]. Součástí API je jazyk pro databázové transakce, který se nazývá Java Persistence Query Language (JPQL). Aktuální verze Java Persistence API je 2.1 [10], která je použita i v projektu Decipher.

4.4 MySQL

MySQL je po Oracle druhý nejvíce používaný systém řízení báze dat [8]. MySQL je projekt s otevřeným zdrojovým kódem pod licencí GNU General Public License vyvíjený firmou Oracle [25]. Pro databázové transakce je určen rozšířený Structured Query Language (SQL). MySQL lze provozovat jak na Windows, tak i na Linuxu a dalších platformách. V projektu Decipher je použit server MySQL ve verzi 5.5.34. Aktuální stabilní verze MySQL je 5.6 [26].

4.5 XML

XML je obecný značkovací jazyk definovaný a standardizovaný World Wide Web Consortium (W3C) v roce 1998 [5]. Zkratka XML značí Extensible Markup Language. Jazyk XML byl vyvinut ze značkovacího jazyka SGML. Používá se pro vytváření pravidel určujících kódování dokumentů, které jsou čitelné jak pro člověka, tak pro počítač. Hlavní návrhové cíle jazyka XML byly jednoduchost, obecnost a použití na internetu. Nejčastěji se používá pro zápis dokumentů obsahující strukturovaná data. Poslední standard je ve verzi 1.1, která má oproti standardu 1.0 striktnější požadavky na znaky v názvech prvků, atributů a identifikátorů [6].

4.6 DOM

Document Object Model (DOM) je rozhraní pro reprezentaci dokumentů HTML, XHTML a XML. Rozhraní je nezávislé na jazyku i platformě a umožňuje interakci s jednotlivými objekty dokumentu. Objekty jsou strukturovány do tzv. stromu DOM, přičemž objekt, který je nadřazen všem dalším objektům, je kořen stromu. Například v dokumentu HTML je to uzel HTML. Na objekty ze stromu mohou být volány metody, které umožňují pracovat s objekty a adresovat další objekty stromu. DOM byl poprvé doporučen konsorciem W3C v roce 1998 [4] a od té doby se rozšiřuje jeho verze 1.0. Poslední doporučení je z roku 2004 nazvané Document Object Model Level 3 [15].

4.7 XPath

XPath je jazyk pro vyhledávání informací v dokumentu XML. Definovalo ho konsorcium W3C v roce 1999 [7]. Pomocí jazyka XPath lze vyhledávat uzly v DOM dokumentu a číst data a atributy uzlů. Základem jazyka je Path Expression – výraz určující cestu k uzlům. Výraz je zapsán jako sekvence přechodů mezi jednotlivými množinami uzlů. Přechody jsou odděleny lomítky. Každý přechod je určen třemi prvky: identifikátorem osy, testem uzlu a podmínkou (predikátem). Identifikátor a podmínka nemusí být uvedeny, pak se použije jejich implicitní hodnota. V anotačním editoru se používají cesty s testem uzlu. Příklad cesty

XPath: `/HTML[1]/BODY[1]/P[1]/text()[2]`. Příklad obsahuje cestu k druhému textovému uzlu prvního odstavce v těle uzlu HTML. Aktuální verze XPath je 3.0 [28].

Kapitola 5

Návrh nových algoritmů vyhledávání a aktualizace fragmentů

Kapitola navrhuje nové principy algoritmů pro vyhledávání a aktualizaci fragmentů.

U iterátorů bylo potřeba odstranit dvojitý průchod podstromem u rekurzivního iterátoru, ale zároveň bylo třeba zachovat průchod stromem od uzlu, kde se nacházel původní fragment, protože v případě, že uživatel změní text fragmentu, se bude fragment s největší pravděpodobností nacházet v původním textovém uzlu. Problémy rekurzivního iterátoru budou odstraněny v dvousměrném iterátoru, který je popsán v sekci 5.1.

Způsoby využití iterátorů ve vyhledávacích metodách jsou popsány v sekci 5.2.

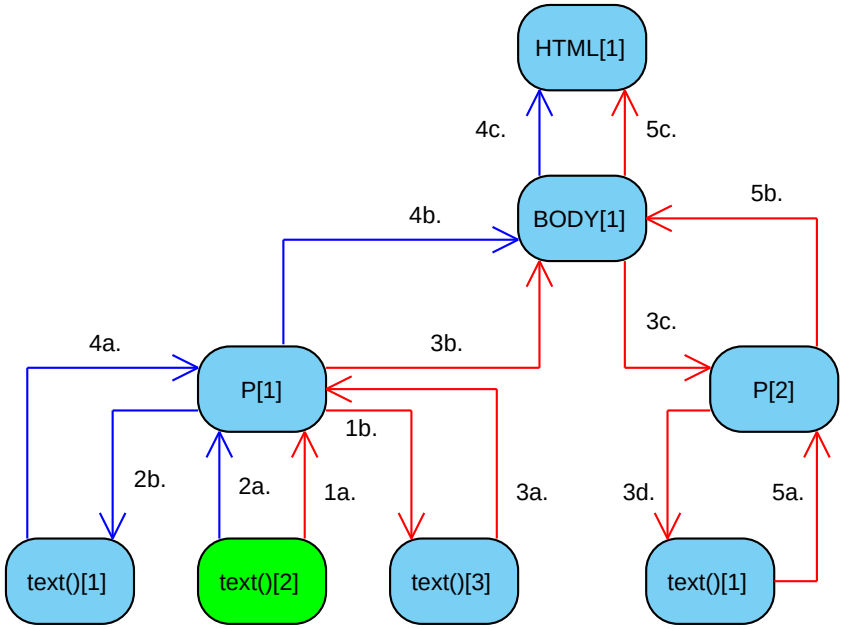
Typy průchodů textovým uzlem budou rozšířeny o další metody popsané v sekci 5.3. Porovnávací metody budou využity v původní implementaci.

Nový systém bude postaven na třech skupinách metod pro vyhledání fragmentů. Skupiny jsou rozepsány v sekci 5.5.

5.1 Dvousměrný iterátor

Dvousměrný iterátor vychází z rekurzivního iterátoru popsaného v sekci 3.4.2 a odstraňuje dvojitý průchod podstromem. V iterátoru je řešen průchod stromem přes dvě cesty. Průchod je znázorněn na obrázku 5.1.

Počátek průchodu je v textovém uzlu, ve kterém se nacházel původní fragment. Další průchod je rozdělen na dvě části. Každá část vždy dojde až k listovému uzlu. První cesta (na obrázku je vyznačena červenou barvou) obsahuje uzly, které následují za původním textovým uzlem, a druhá cesta (na obrázku je vyznačena modrou barvou) obsahuje uzly, které předchází původní textový uzel. V krajních uzlech průchod pokračuje stejným způsobem od otcovského uzlu. Vyhodnocují se až listové uzly, které obsahují text. Tímto průchodem je zaručeno, že se budou vybírat uzly, které se postupně vzdalují od původního textového uzlu.



Obrázek 5.1: Průchod dvousměrného iterátoru

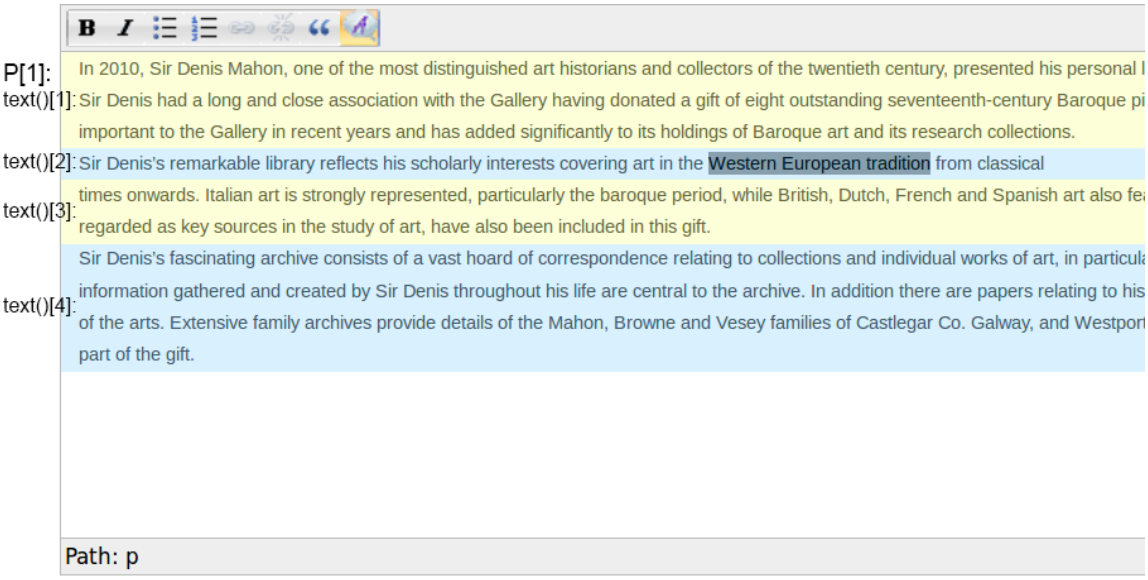
5.2 Způsoby použití iterátorů k vyhledávání a aktualizaci fragmentů

Původní systém vyhledávání a aktualizace fragmentů používal pouze dvě metody pro vyhledání a aktualizaci fragmentu `match()` a `matchAll()` třídy `MatcherProvider`, které byly popsané v posledním odstavci sekce 3.3. Nově je tento systém rozšířen o metody, které počítají s dvousměrným iterátorem, a o metodu, která používá sekvenční iterátor. Metody popsané v této sekci jsou určeny k aktualizaci pozice fragmentu v případě, že při kontrole původní pozice fragmentu fragment nebyl nalezen.

5.2.1 Průchod od původního uzlu

Princip průchodu je použitelný v případě, že textový uzel, kde se nacházel původní fragment, zůstal na svém místě v rámci XPath v dokumentu, nebo ho nahradil jiný uzel. Pro vyhledání nového fragmentu je použit dvousměrný iterátor. V případě pouhé změny textu fragmentu je textový uzel s fragmentem nalezen okamžitě. V případě odsunutí textového uzlu s fragmentem a případné změny textu fragmentu jsou dvě možnosti odsunutí. Příklad části původního dokumentu použitého v ukázkách změny dokumentu je na obrázku 5.2. První možností je, že uživatel vloží před textový uzel s původním fragmentem jeden, nebo více jiných textových uzlů. Druhá možnost je, že odstraní jeden, nebo více textových uzlů před textovým uzlem s původním fragmentem. V obou případech dojde k posunutí textového uzlu s původním fragmentem a je nutné nalézt nový XPath fragmentu. Nejvýhodnější je použít dvousměrný iterátor, protože bude hledat na oba směry odsunutí od původního fragmentu a pokryje obě možnosti odsunutí.

V novém systému budou implementovány dvě metody popsané v následujících dvou odstavcích založené na principu průchodu dokumentem popsaném v počátku této sekce. Metody jsou vhodné i pro případ změny textu a pozice fragmentu v rámci textového uzlu, protože vyhledání v textovém uzlu implementuje až metoda průchodu textovým uzlem.



Obrázek 5.2: Původní dokument

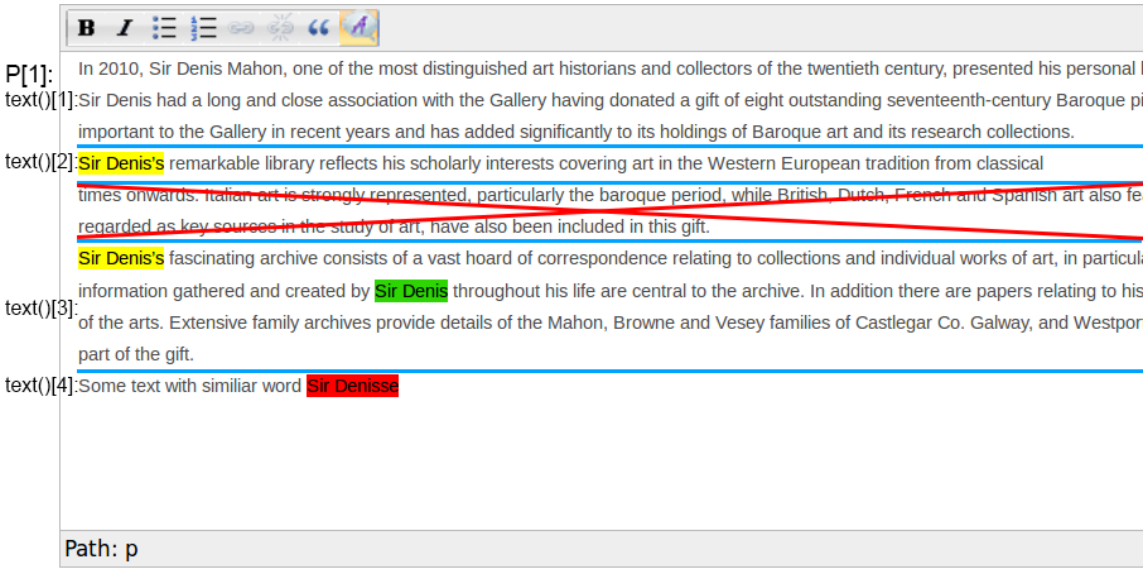
Neomezený průchod

Metoda s neomezeným průchodem projde celý textový dokument dvousměrným iterátorem od textového uzlu, kde se původně fragment nacházel. Pro každý textový uzel bude volat metody pro průchod textovým uzlem a nalezení fragmentu podle jejich priorit od největší k nejmenší. Pří prvním nalezeném dostatečně shodném fragmentu metoda skončí a navrátí výsledný fragment.

Omezený průchod

Metoda s omezeným průchodem je vylepšení metody s neomezeným průchodem. Projde tři nejbližší textové uzly od textového uzlu, kde se původně fragment nacházel. Pro každý textový uzel bude volat metody pro průchod textovým uzlem a nalezení fragmentu podle jejich priorit od největší k nejmenší. Každý nalezený fragment bude uchován v poli a po skončení metody bude pole předáno k vyhodnocení nově nalezených fragmentů. V případě, že se nenajde žádný dostatečně shodný fragment, metoda bude pokračovat v neomezeném průchodu a navrátí první dostatečně shodný fragment.

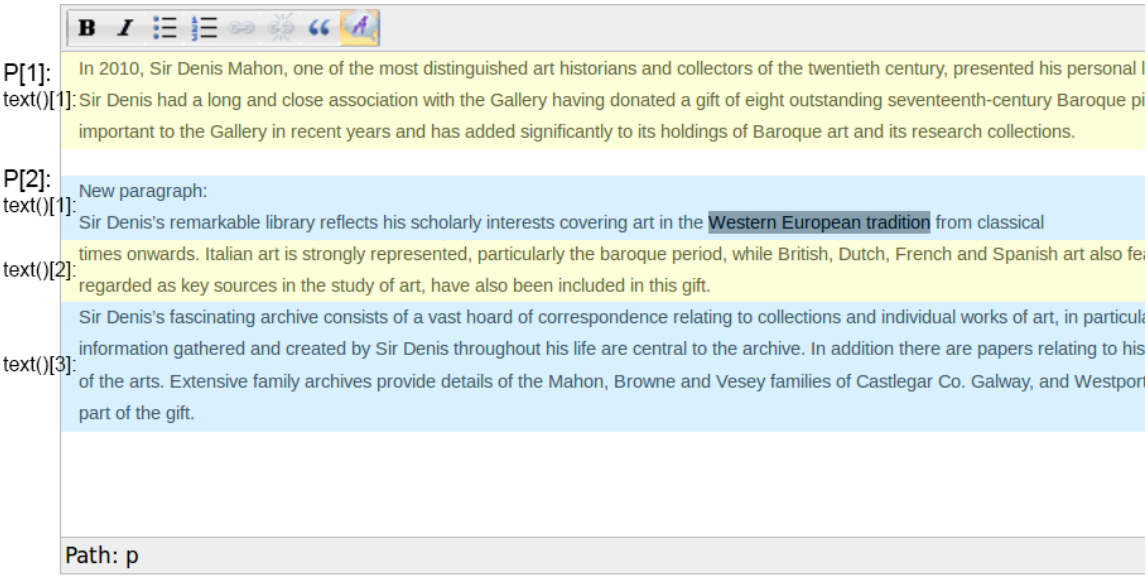
Porovnání vyhledaných fragmentů omezeným a neomezeným průchodem je na obrázku 5.3. Původní dokument měl anotován text „**Sir Denis**“. Textový element obsahující fragment anotace byl odsunut o jednu pozici. Omezený průchod začíná v původním textovém uzlu a pokračuje, dokud nenajde první dostatečně shodný text. V tomto případě by našel červeně vyznačený text „**Sir Denisse**“. Omezený průchod nejprve vyhodnocuje tři nejbližší uzly a v nich hledá nejlepší shodu. Omezený průchod v tomto případě najde fragmenty v pořadí: červeně vyznačený fragment, žlutě vyznačený fragment v třetím textovém uzlu, zeleně vyznačený fragment a žlutě vyznačený fragment v druhém textovém uzlu. Zeleně označený fragment je následně vyhodnocen jako nejlepší shoda.



Obrázek 5.3: Porovnání omezeného a neomezeného průchodu

5.2.2 Sekvenční průchod

Sekvenční průchod se použije v případě, že nelze použít průchod od původního uzlu. Průchod od původního uzlu nelze použít v případě, že se na XPath původního fragmentu nenachází žádný textový uzel. Důsledkem toho je, že neexistuje referenční uzel pro inicializaci dvousměrného iterátoru a je nutné použít sekvenční iterátor. Ukázka případu použití sekvenčního iterátoru je na obrázku 5.4. Textový uzel, ve kterém se nacházel fragment, byl přesunut do nového odstavce. Na XPath odsunutého textového uzlu se již nenachází žádný uzel a vyhodnocení XPath nelze použít pro inicializaci iterátoru.



Obrázek 5.4: Vytvoření nového odstavce a smazání původního textového uzlu

Metoda založená na sekvenčním průchodu vylepšuje metodu `matchAll()`, která byla

použita ve starém systému. Průchod začíná v prvním uzlu dokumentu a končí v posledním uzlu dokumentu. V každém uzlu je volána jedna z metod pro průchod textovým uzlem a nalezení fragmentu. Původní metoda navracela pole s nalezenými fragmenty, které byly určeny jako dostatečně shodné při použití vyhledávacích metod se všemi prioritami. Nová metoda hledá fragmenty s jednou vyhledávací metodou, a pokud není žádný fragment nalezen v celém dokumentu, teprve poté použije metodu s nižší prioritou a hledá znovu v celém dokumentu. Nová metoda navrací výsledky pouze v rámci jedné priority vyhledávací metody.

5.3 Návrh nových metod průchodu textovým uzlem

Původní metody pro průchod textovým uzlem nebyly dostatečně efektivní a výběr porovnávaného řetězce z textového uzlu byl omezený na délku řetězce původního fragmentu či na počet slov původního fragmentu. Metody pro průchod textovým uzlem tato práce rozšiřuje a vylepšuje.

5.3.1 Dvousměrný průchod po znacích

Jednosměrný průchod po znacích implementovaný ve starém systému začínal na počátku textového uzlu a iteroval do konce textového uzlu. Nový systém začíná na offsetu původního fragmentu a iteruje směrem k začátku a konci uzlu. V případě, že je textový uzel zkrácen a offset původního fragmentu je mimo uzel, začíná se na konci uzlu a iteruje se pouze směrem k začátku uzlu. Pokud uživatel posune offset fragmentu o N znaků, je fragment nalezen touto metodou v N iteracích. Dokud průchod nenarazí na začátek či konec uzlu, v každé iteraci proběhnou dvě porovnání. Pro porovnání se vytváří dva podřetězce uzlu ve směru iterací. Dvousměrný iterátor je efektivnější a má větší pravděpodobnost vyhledání původního odsunutého fragmentu. Pokud textový uzel obsahuje text fragmentu více než jednou, pravděpodobnost nálezu původního fragmentu jednosměrným průchodem od počátku uzlu se snižuje s každým dalším výskytem původního textu.

Příklad využití dvousměrného průchodu po znacích

Mějme dokument s jedním textovým uzlem obsahující větu:

Academic notes and material containing information gathered and created by ***Sir Denis*** *throughout his life are central to the archive.*

Anotace je vytvořena ke jménu „**Sir Denis**“. Poté je fragment odsunut o slovo „text“ a jednu mezeru. Věta se změní na:

Academic notes and material containing information gathered and created by text ***Sir Denis*** *throughout his life are central to the archive.*

Dvousměrný průchod je znázorněn v tabulce 5.1, která obsahuje porovnávané řetězce při průchodu.

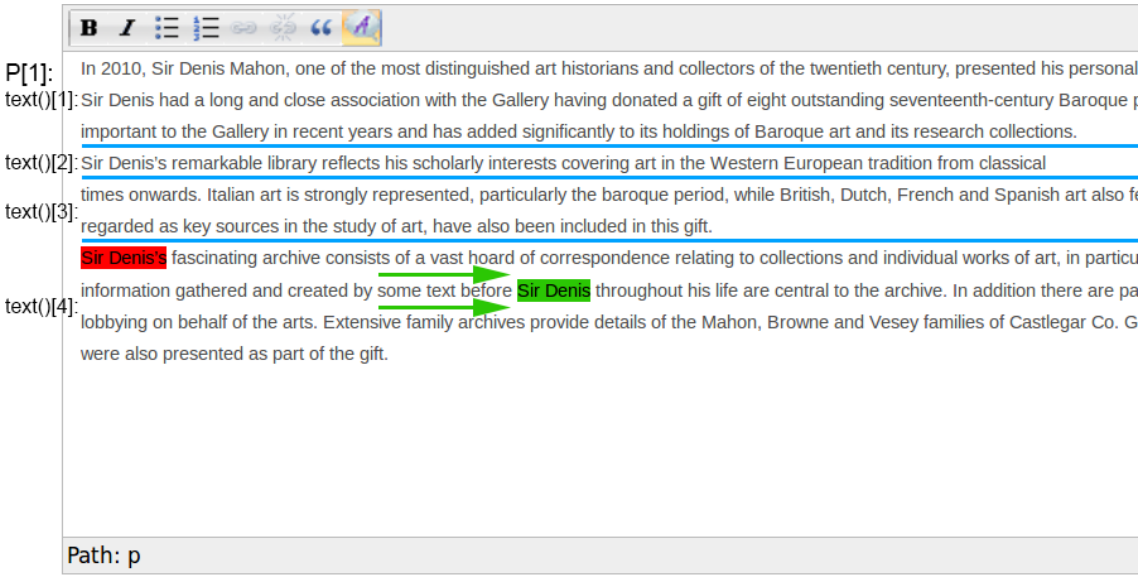
5.3.2 Vylepšení průchodu po slovech

Vylepšení metody spočívá v neukončení průchodu při vyhodnocení fragmentu jako dostatečně podobného původnímu fragmentu. Průchod uzlem pokračuje do konce uzlu a všechny

Iterace ve směru ke konci uzlu	Iterace ve směru k začátku uzlu
"text Sir "	"text Sir "
"ext Sir D"	" text Sir"
"xt Sir De"	"y text Si"
"t Sir Den"	"by text S"
" Sir Deni"	" by text "
"Sir Denis"	"d by text"

Tabulka 5.1: Porovnávané řetězce při dvousměrném průchodu po znacích

nalezené dostatečně podobné fragmenty jsou uchovány v poli. Pole je po skončení průchodu předáno k vyhodnocení nalezených fragmentů. Ukázka nalezených fragmentů po odsunutí fragmentu je na obrázku 5.5. Původní metoda by nalezla při použití Levenshteinovy metody s maximálně 23% odchylkou od původního slova fragment vyznačený červeně. Nová metoda nalezne dva podobné fragmenty a hodnotící metoda zvolí zeleně vyznačený fragment.



Obrázek 5.5: Text s nalezenými fragmenty po odsunutí původního fragmentu

5.3.3 Shodné první a poslední slovo

Průchod textovým uzlem iteruje přes jednotlivá slova textového uzlu, dokud porovnávací metoda neshledá slovo dostatečně shodné prvnímu slovu původního fragmentu. Poté se z následujících slov uzlu vytváří řetězec, dokud porovnávací metoda neshledá slovo dostatečně shodné poslednímu slovu původního fragmentu a dokud délka řetězce nepřekročí limit pro rozšíření délky textu fragmentu. Pokud nalezený řetězec není kratší než limit pro zmenšení délky textu fragmentu, uchová se v poli. Po skončení průchodu je pole s nalezenými fragmenty předáno k vyhodnocení. Vyhledávací metoda je určena k nalezení změny textu fragmentu při zachování prvního a posledního slova. Mějme dokument s jedním textovým uzlem obsahující větu:

*In 2010, **Sir Denis Mahon**, one of the most distinguished art historians and collectors of the twentieth century, presented his personal library and archive to the National Gallery of Ireland.*

Anotace je vytvořena ke jménu „**Sir Denis Mahon**“. Fragment je možné změnit v rámci těla fragmentu. Tělo je zde reprezentováno slovem „**Denis**“. Fragment můžeme změnit například takto: „**Sir D. Mahon**“. Pokud změna nerozšíří či nezkrátí fragment o předem určený limit, fragment bude nalezen.

5.3.4 Shodné první, nebo poslední slovo a poslední, nebo první znak

Průchod je založen na podobném principu jako průchod s vyhodnocením shodného prvního a posledního slova. Rozdíl je, že při průchodu uzlem se porovnávají slova s prvním i posledním slovem původního fragmentu. V případě vyhodnocení slova jako dostatečně shodného se rozšiřuje vytvářený řetězec směrem ke konci uzlu, když je shodné první slovo původního fragmentu, nebo k začátku uzlu, když je shodné poslední slovo původního fragmentu. Řetězec je rozšiřován, dokud není překročen limit pro rozšíření délky textu fragmentu a dokud slovo, o které řetězec rozšiřujeme, nekončí posledním znakem posledního slova původního fragmentu, v případě rozšíření ve směru ke konci uzlu, nebo nezačíná prvním znakem prvního slova původního fragmentu, v případě rozšíření ve směru k začátku uzlu. Další zpracování řetězce je provedeno stejně jako v metodě s vyhodnocením shodného prvního a posledního slova. Vyhledávací metoda je určena k nalezení změny textu fragmentu při zachování prvního/posledního slova a posledního/prvního znaku. Pokud je vytvořena anotace „**William John Leech**“, fragment je možné změnit v rámci těla fragmentu a prvního, nebo posledního slova. Oproti metodě se shodou prvního a posledního slova metoda nalezne i změnu ve tvaru: „**W. John Leech**“. Fragment po změně však nesmí být delší/kratší než předem určený limit pro zvětšení/zmenšení fragmentu.

5.3.5 První slovo a maximální shoda

Poslední metoda průchodu umožňuje nejvíce změn v textu fragmentu. Princip průchodu textovým uzlem je stejný jako princip průchodu, který vyhodnocuje shodné první a poslední slovo. Rozdíl je v ukončující podmínce při vytváření řetězce po nalezení shodného prvního slova. Po nalezení dostatečně shodného prvního slova se vytvářený řetězec rozšiřuje o další slova, dokud se Levenshteinova vzdálenost od textu původního fragmentu zmenšuje a dokud není překročen limit pro rozšíření délky textu fragmentu. Další zpracování řetězce je provedeno stejně jako v metodě s vyhodnocením shodného prvního a posledního slova. Vyhledávací metoda je určena k nalezení změny textu fragmentu při zachování prvního slova a řeší maximální změnu konce fragmentu. Pokud je vytvořena stejná anotace jako v ukázce metody shodného prvního a posledního slova, fragment je možné změnit v rámci těla fragmentu a posledního slova. Metoda najde změny ve tvaru: „**Sir Denis M.**“. Fragment po změně však nesmí být delší/kratší než předem určený limit pro zvětšení/zmenšení fragmentu.

5.4 Vyhodnocení nalezených fragmentů

Statický průchod a průchod po znacích navrací jeden nalezený fragment k případné aktualizaci. Ostatní metody průchodu navrací pole s nalezenými fragmenty. Z fragmentů v poli

je nutné vybrat nejlepší fragment pro aktualizaci. Vyhodnocení probíhá buď na konci průchodu textovým uzlem, nebo po ukončení vyhledávací metody.

5.4.1 Vyhodnocení v rámci více uzlů

Při vyhodnocení v rámci více uzlů se nalezené fragmenty porovnávají s původním fragmentem podle XPath, offsetu a textu. Každý fragment je ohodnocen body, které značí rozdíl oproti původnímu fragmentu. Hodnocen je rozdíl v offsetu o jeden znak, Levenshteinova vzdálenost textu vůči původnímu textu fragmentu a Levensteinova vzdálenost XPath vůči původnímu XPath. Ohodnocení lze získat vzorcem 5.1.

$$\begin{aligned}(\text{ohodnocení fragmentu}) = (\text{offset}) + (\text{Lev. vzdálenost textu}) * 5 + \\ (\text{Lev. vzdálenost XPath}) * 355\end{aligned} \tag{5.1}$$

Rozdíl v offsetu o jeden znak je hodnocen jedním bodem. Levenshteinova vzdálenost XPath je násobena 355 body. V dokumentech poskytnutých v rámci projektu Decipher měly všechny dokumenty pouze jeden odstavec a více textových uzlů. Žádný jiný typ uzlu se v dokumentech nevyskytoval. V XPath fragmentů se proto měnilo pouze číslo indexu textového uzlu. Průměrný počet znaků na jeden textový uzel v dostupných dokumentech byl 355. Pokud by se změnil offset o počet znaků jednoho průměrného textového uzlu, byl by fragment ohodnocen 355 body, proto je konstanta násobení jedné změny XPath určena touto hodnotou.

Konstanta pro násobení Levenshtenovy vzdálenosti textu fragmentu ovlivňuje stejně jako offset, jestli daný fragment není výhodnější posunout do jiného textového uzlu. Ohodnocení však musí být vyšší. Pokud se připíše text před fragment, musí být posun v offsetu fragmentu hodnocen méně přísně než ohodnocení změny textu na nově připsaný text. Ve znalostní bázi nástroje pro rozpoznání pojmenovaných entit v textu Knowledge Base projektu Decipher je 63 % všech záznamů typu osoba a umělec. Průměrný počet slov na název umělců a osob je 2,24. V celé databázi je průměrný počet slov na název 2,27. Z toho se dá odvodit, že většina názvu je dvouslovných až tříslovných, kdy se jedná převážně o jména umělců. V případě, že se změní 2/3 jména, jedná se už nejpravděpodobněji o jiného umělce, a je proto výhodnější zvolit nezměněné jméno v dalším textovém uzlu. Změna 2/3 jména musí být hodnocena stejnou vahou jako posunutí offsetu o průměrnou délku uzlu. Konstantu hodnocení textového uzlu lze spočítat vzorcem 5.2, který je spočítán v rovnici 5.3.

$$(\text{konstanta}) = (\text{průměrný počet znaků textového uzlu})/(2/3 \text{ ovlivnění názvu}) \tag{5.2}$$

$$5,325 = 355/66,\overline{6} \tag{5.3}$$

Na konci vyhodnocení je z pole vybrán fragment, který má nejmenší ohodnocení rozdílu vůči původnímu fragmentu.

5.4.2 Vyhodnocení v rámci uzlu

Oproti vyhodnocení v rámci více uzlů není hodnocena Levenshteinova vzdálenost XPath. Konstanty pro ohodnocení jsou určeny stejně. Ohodnocení lze získat vzorcem 5.4.

$$(\text{ohodnocení fragmentu}) = (\text{offset}) + (\text{Lev. vzdálenost textu}) * 5 \tag{5.4}$$

5.5 Skupiny metod vyhledávání a aktualizace fragmentů

Nový systém vyhledávání a aktualizace fragmentů bude postaven na třech skupinách vyhledávacích metod (objektech třídy `MatcherProvider`). První se shoduje s původní skupinou, slouží k vyhledání původního fragmentu a používá se ve stejné implementaci jako ve starém systému. Druhá je určena pro nalezení fragmentu co nejblíže původní pozici fragmentu. Pokud je původní pozice smazána, je použita třetí skupina, která projde celý dokument a snaží se najít co nejvíce podobný fragment v každém textovém uzlu.

Každý z objektů třídy `MatcherProvider` je třeba inicializovat vyhledávacími metodami (objekty třídy `Matcher`). Vyhledávací metody je třeba inicializovat vhodnou kombinací iterátoru, porovnávací metody a metody pro průchod textovým uzlem. Z iterátorů má smysl vybírat pouze ze **sekvenčního** a **dvousměrného iterátoru.** Rekurzivní iterátor oproti dvousměrnému nenabízí žádnou výhodu a iterátor XPath se používá pouze pro kontrolu, jestli se fragment nezměnil. Z porovnávacích metod lze použít **porovnání přesné shody** a **Levenshteinovu metodu**. Metoda Soundex neumí pracovat s diakritikou, a proto nebude použita. Nejvíce optimalizovaná implementace metody Metaphone je placená. Metoda Metaphone je primárně zaměřena na anglické názvy. V projektu se však vyskytují i názvy v jiných jazycích, proto tato metoda nebude použita. Metoda není použita i proto, že není příliš tolerantní ke zkrácení a prodloužení fragmentu. V případech, kde by tato metoda byla úspěšná, je úspěšná i Levenshteinova metoda. Metoda je užitečná pouze při velmi specifických změnách především anglických slov, ale výskyt těchto změn je málo pravděpodobný, proto by vyhodnocování touto metodou spíše prodloužilo proces aktualizace fragmentů. Kromě toho je zde i riziko, že se vyhodnotí špatný fragment jako náhrada, protože by musel obsahovat již velké množství změn, aby ho nenašla Levenshteinova metoda. Tím se také zvyšuje pravděpodobnost, že nalezený fragment bude mít jiný význam. Dostupné metody průchodu textovým uzlem se nachází v následujícím výčtu:

- **Statická metoda**
- **Metoda po slovech**
- **Dvousměrný průchod po znacích**
- **Metoda kontrolující první a poslední slovo**
- **Metoda kontrolující první/poslední slovo a poslední/první znak**
- **Metoda kontrolující první slovo a maximální shodu**

Poslední tři metody ve výčtu budou v implementaci sloučeny v jednu inicializační metodu, která používá postupně metody, jak jsou uvedeny ve výčtu. Při nalezení dostatečně shodného fragmentu vůči původnímu fragmentu se již další metoda nevyužije.

První skupina obsahuje pouze jednu metodu vyhledávání fragmentů, která je inicializována iterátorem XPath, porovnávací metodou, která hledá přesnou shodu, a statickým průchodem textovým uzlem. Hledá pouze na původní pozici fragmentu a navrací jen stejný fragment. Implementace metody je stejná jako v původním systému a je určena pro kontrolu výskytu fragmentu na původní pozici se stejným textem.

5.5.1 Skupina metod pro vyhledání nejbližšího fragmentu

Skupina obsahuje tři metody pro vyhledávání fragmentů. Všechny metody jsou inicializovány dvousměrným iterátorem z důvodu nalezení fragmentu co nejblíže původnímu uzlu.

První metoda využívá porovnávací metodu na přesnou shodu a dvousměrný průchod textovým uzlem. Metoda je určena k nalezení odsunutého fragmentu s nezměněným textem.

Druhá metoda používá vylepšeny průchod textovým uzlem po slovech a je určena k nalezení odsunutého fragmentu a změn jednoho slova v rámci fragmentu. Porovnává potenciální fragmenty Levenshteinovou metodou. Procentuální shoda fragmentu s původním fragmentem je nastavena na 56 %. Názvy z Databáze Knowledge Base se skládají průměrně z 2,27 slov. Změna je povolena na opravu jednoho slova. Podíl jednoho slova na celém názvu je 44 %, proto je zachování původního fragmentu nastaveno na 56 %.

Třetí metoda prochází textový uzel po slovech. Nejprve se snaží najít fragmenty se stejným prvním a posledním slovem k původnímu fragmentu. Při porovnání je použita Levenshteinova metoda a záleží na podobnosti prvního a posledního slova, zbytek textu se již neporovnává a umožňuje se tím zvětšení nebo zmenšení fragmentu. Pokud není nalezen žádný fragment v textovém uzlu, pokračuje se hledáním se shodou prvního/posledního slova a posledního/prvního písmena vůči původnímu fragmentu. Zde se již porovnává pouze první, nebo poslední slovo. Pokud i tato metoda průchodu selže, je vyhledáváno pomocí prvního slova a maximální shody. Porovnání Levenshteinovou metodou je inicializováno na podobnost s původním slovem 83 %. Délka průměrného slova z databáze je 6 znaků. Porovnává se vždy jen jedno slovo a nastavením je umožněna změna jednoho znaku pro opravení překlepu. U metody s porovnáním prvního a posledního slova se jednotlivá slova porovnávají zvlášť, proto platí nastavení i pro tuto metodu. Metoda je určena k nalezení změn většiny slov fragmentu a k možnému rozšíření či zmenšení počtu slov fragmentu.

Druhé a třetí metodě ve skupině je nastaveno maximální rozšíření a maximální zmenšení fragmentu. Rozšíření je nastaveno na 144 % původního názvu a zmenšení na 56 % původního názvu. Rozšíření a zmenšení je nastaveno, aby dovolovalo přidání/ubrání jednoho průměrně dlouhého slova v průměrně dlouhém názvu.

5.5.2 Skupina metod pro průchod celým dokumentem

Tato skupina je použita v případě, že selže druhá skupina, a je určena k prohledání celého dokumentu. Všechny metody ve skupině jsou inicializovány sekvenčním iterátorem. Metody jsou kromě iterátoru shodné s metodami druhé skupiny.

Inicializační metody pro vyhledávání a aktualizaci fragmentů jsou ve druhé a třetí skupině seřazeny od nejméně tolerantní po nejvíce tolerantní. Místo druhé a třetí skupiny by se dala využívat pouze jedna skupina, která by obsahovala inicializační data z obou skupin, ale to by přineslo pouze nevýhodu v obtížné práci s již navrženými metodami pro práci se skupinami.

Kapitola 6

Návrh nového algoritmu detekce míry ovlivnění anotací

Algoritmus je třeba navrhnout jako vylepšení starého systému, který ovlivnění anotací dokumentu při synchronizaci dokumentu indikoval pouze při nenalezení některého z původních fragmentů. Algoritmus bude fungovat na principu prahování, kdy se ohodnotí ovlivnění všech anotací v dokumentu a pak se rozhodne, jestli ohodnocení není horší než daný práh a jestli bude dokument aktualizován bez interakce uživatele, nebo bude muset uživatel rozhodnout, jestli se provede synchronizace a automatická aktualizace anotací.

6.1 Ohodnocení fragmentu

Ohodnocení bude založeno na porovnání nových a původních hodnot XPath, offsetu a textu fragmentu. Pro každý typ porovnávané hodnoty je vytvořen práh. Pokud výsledek porovnání hodnot bude vyšší než práh pro porovnávaný typ hodnoty, ohodnocení ovlivnění fragmentu bude zvýšeno o konstantu 1/3. Konstanta byla zvolena rovnoměrnou distribucí ovlivnění fragmentu podle XPath, offsetu i textu fragmentu. Tato distribuce umožňuje při vyhodnocení fragmentu povolit jednu změnu fragmentu typu: připsání odstavce (posun fragmentu v rámci XPath), připsání textu do odstavce (posun offsetu fragmentu v rámci uzlu) nebo oprava textu fragmentu. Celkové vyhodnocení bude probráno na konci kapitoly. Výsledek ohodnocení může nabývat hodnot v intervalu $\langle 0, 1 \rangle$ a určuje procentuální ovlivnění fragmentu pomocí desetinné hodnoty.

6.1.1 Práh Levenshteinovy vzdálenosti XPath

Textové uzly v dokumentech poskytnutých v rámci projektu Decipher měly jednotný tvar XPath: `/HTML[1]/BODY[1]/P[X]/text()[Y]`, přičemž proměnná X vyjadřuje, v kolikátém odstavci se fragment nachází, a proměnná Y vyjadřuje, v kterém textovém uzlu odstavce se fragment nachází. Podle vzorce 6.1 lze matematicky spočítat ovlivnění XPath fragmentu. Pro převod z desetinného čísla na procenta je třeba hodnotu ještě vynásobit číslem 100. V tabulce 6.1 jsou spočítané hodnoty, kterých lze dosáhnout.

$$\text{(Ovlivnění XPath)} = \text{(Leven. vzdálenost XPath)}/\text{(Počet znaků původního XPath)} \quad (6.1)$$

Hodnoty 4 % a 7 % povolují maximálně 1 změnu a 2 změny. Nastavení prahu má smysl vybírat pouze mezi těmito dvěma konstantami (případně žádnou povolenou změnou),

Počet změn	Ovlivnění
0	0 %
1	3,2 %
2	6,5 %
3	9,7 %
4	12,9 %
5	16,1 %
6	19,4 %
7	22,6 %
8	25,8 %

Tabulka 6.1: Závislost počtu změn znaků XPath na ovlivnění XPath při délce XPath 31 znaků. Hodnoty jsou zaokrouhleny na jedno desetinné číslo

protože jakákoliv vyšší konstanta již povoluje velké odsunutí fragmentu (až 999 textových uzlů). Jelikož se v dokumentech poskytnutých v rámci projektu Decipher vyskytovaly pouze dokumenty s jedním odstavcem a více textovými uzly, byla vybrána hodnota 4 %, která dovoluje minimální změnu při uvedeném způsobu vyhodnocení, a to jednu změnu v XPath. Změna může odsunout uzel o jeden až devět textových uzlů. Případná změna o násobek deseti je zanedbána, protože průměrný počet textových uzlů dostupných dokumentů s více než jedním uzlem je 8. Ostatní dokumenty obsahují pouze jeden uzel. Jedná se o změnu indexu textového uzlu například z hodnoty 2 na hodnotu 12, 22, ..., 92, která je také hodnocena 4 %.

6.1.2 Práh offsetu

Práh umožňuje posunutí fragmentu v rámci textového uzlu. Při ovlivnění offsetu půjde ve většině případů o ovlivnění pouze jednoho textového uzlu v rámci dokumentu. Typicky půjde o dopsání/smazání textu v rámci textového uzlu. V dokumentech poskytnutých v rámci projektu Decipher byl průměrný počet znaků na jednu větu 177,6. Práh bude dovolovat změnu posunutí o délku jedné průměrné věty. Shodou okolností tato hodnota udává také polovinu počtu znaků v průměrném textovém uzlu dokumentů.

6.1.3 Práh Levenshteinovy vzdálenosti textu

Práh Levenshteinovy vzdálenosti textu je zvolen podobně jako u vyhodnocování nejlepšího fragmentu z pole. Práh je nastaven na maximální změnu 67 %. Při průměrném počtu slov v názvech z databáze 2,27 dovoluje změnu jednoho až dvou slov ve fragmentu. Při majoritním výskytu jmen umělců s dvěma až třemi slovy ve jméně je překročením prahu jakákoliv změna 2/3 jména, kdy už s největší pravděpodobností jde o jiného umělce.

6.1.4 Celkové ohodnocení fragmentu

Následující tabulka 6.2 sumarizuje nastavení konstant, které ovlivňují vyhodnocení fragmentu.

Typ ovlivnění	Práh	Podíl ovlivnění
Levenshteinova vzdálenost XPath	4 %	$33,\overline{3}$ %
Absolutní hodnota rozdílu offsetů	178	$33,\overline{3}$ %
Levenshteinova vzdálenost textů	67 %	$33,\overline{3}$ %

Tabulka 6.2: Tabulka s hodnotami prahů a ovlivnění fragmentu při překročení prahu

6.2 Ohodnocení anotace

Ovlivnění anotace je ohodnoceno procentuálním podílem ovlivnění fragmentů. Ohodnocení lze určit vzorcem 6.2.

$$(\text{ohodnocení anotace}) = \sum_{i=1}^{n}(\text{ohodnocení fragmentu})_i * (1/n) \tag{6.2}$$

Proměnná n značí počet fragmentů anotace. Výsledkem je desetinná hodnota v intervalu $\langle 0, 1\rangle$ určující procentuální ovlivnění anotace. Fragmenty mají rovnoměrný podíl na ovlivnění anotace.

6.3 Synchronizace dokumentu

Při synchronizaci dokumentu bude spočteno ohodnocení ovlivnění anotací. Pokud bude ovlivněno více anotací, než určuje práh, změny v dokumentu se buď zahodí a použije se verze dokumentu uložená na serveru, nebo se anotace ve změněném dokumentu nechají automaticky aktualizovat. Práh určuje procentuální ovlivnění všech anotací, které je tolerováno. Ohodnocení ovlivnění anotací lze určit vzorcem 6.3.

$$(\text{ohodnocení ovlivnění anotací}) = \sum_{i=1}^{n}(\text{ohodnocení anotace})_i * (100/n) \tag{6.3}$$

Písmeno n ve vzorci značí počet anotací v dokumentu. Výsledkem je hodnota v intervalu $\langle 0, 100\rangle$ určující procentuální ovlivnění všech anotací v dokumentu. Práh je zvolen na hodnotu 34 %. Vzhledem k rovnoměrným podílům vlivů lze tolerovat právě jeden faktor ovlivnění fragmentu. Práh dovoluje změnit u každého fragmentu jeden posuzující faktor (XPath, offset, text). Jak bylo již popsáno v počátku kapitoly, tímto je dovolena jedna operace s každou anotací. Vzhledem k této skutečnosti lze synchronizovat dokument, který má připsaný jeden, nebo více textových uzlů mezi dvěma sousedními uzly původního dokumentu bez interakce s uživatelem. Interakce uživatele není vyžadována ani při připsání dalšího textu do jednoho ze všech textových uzlů, nebo při změně textu 1/3 anotací.

Kapitola 7

Návrh vylepšení ukládání anotací do databáze

Na základě nedostatků zjištěných v analýze jsem se rozhodl v této rozšiřující kapitole navrhnout jejich opravy a vylepšení ukládání anotací do databáze. Na konci kapitoly navrhuji strukturu třídy, která má na starosti ukládání a aktualizace návrhů anotací.

7.1 Návrh aktualizace položky tmpid a referencí v atributech s odkazem na anotaci

Při požadavku na uložení nových/změněných anotací je třeba odkazovat právě ukládané anotace, které ještě nemají vyplněný primární klíč ID. Anotace v požadavku lze odkazovat pomocí vlastnosti `tmpId`, který je součástí třídy reprezentující anotaci. Vlastnost anotace `tmpId` se však neukládá do databáze, a proto je po uložení anotace do databáze nastavena na hodnotu `null` a je třeba ji znovu nastavit, aby mohla být ve druhé fázi zpracování požadavku využita moduly.

Jak již bylo popsáno v analýze, dále je při ukládání anotací nejprve třeba vyhodnotit atribut typu odkaz na anotaci. Pokud je v atributu anotace odkazována pomocí vlastnosti `tmpId`, musí se nejprve uložit odkazovaná anotace. Při ukládání anotace s atributem obsahujícím odkaz mohou nastat dvě varianty:

1. Pokud je řešen požadavek na uložení nových anotací, všechny anotace ještě nemusí být uloženy a je nutné řešit odkazy na neuložené anotace. Může nastat případ, kdy by se ukládala anotace, která by odkazovala na anotaci, která ještě není uložená. Neuložená anotace může odkazovat zpět na ukládanou anotaci. Metoda na ukládání se volá rekurzivně a pokud by se ukládala nejprve odkazovaná anotace, problém odkazu to nevyřeší. Nastal by nekonečný cyklus. Proto když server při ukládání anotací narazí na anotaci s atributem obsahujícím odkaz, anotace bude uložena bez atributu s odkazem a odkaz bude přidán teprve až po uložení všech anotací z požadavku. Kontrolovat se budou i atributy zanořených anotací, protože zanořené anotace se ukládají zároveň se svou nadřazenou anotací. V závěru se provede aktualizace položky `tmpId` nově ukládané anotace a nahrazení všech odkazů pomocí vlastnosti `tmpId` za referenci na objekt odkazované anotace.

2. Nové anotace z požadavku jsou již uložené a řeší se pouze změněné anotace. Zde nemůže nastat situace vzájemného odkazu dvou anotací, kde by obě anotace ještě ne-

byly uložené, a ukládání lze vyřešit přednostním uložením anotace z odkazu. V závěru se stejně jako u předchozího případu provede aktualizace položky `tmpId` nově ukládané anotace a nahrazení všech odkazů pomocí vlastnosti `tmpId` za referenci na objekt odkazované anotace.

7.2 Návrh opravy duplikace fragmentů a špatného uložení specifické kombinace atributů

Při aktualizaci již uložené anotace se před nahrazením staré anotace za novou musí provést dvě změny. Musí se zvlášť zaktualizovat atributy a fragmenty anotace, jinak by došlo k duplikaci, nebo špatnému nahrazení, protože jsou ukládány ve vlastní databázové tabulce mimo tabulku s anotacemi.

Pro korektní provedení aktualizace fragmentů se nastaví novým fragmentům identifikační čísla již uložených fragmentů. Pokud je nových fragmentů menší počet, přebývající fragmenty v databázi se odstraní. Pokud je naopak nových fragmentů více, přebytečné fragmenty nemají nastavené identifikační číslo, aby byly korektně přidány do databáze.

Aktualizace atributů je více komplikovaná. Třída popisující atribut je odvozena od třídy `BaseAttribute`. Třída `BaseAtrribute` obsahuje veškeré vlastnosti pro uložení kteréhokoliv odvozeného atributu. Odvozené atributy obsahují zejména operace pro práci se specifickým typem atributu. Při výběru atributu z databáze se atribut za určitých situací nenačetl v podobě správného typu odvozené třídy. Způsobeno je to pravděpodobně verzí EclipseLink, kterou projekt Decipher používá. Tato skutečnost komplikovala aktualizaci. Atributy se navíc načítaly z databáze v jiném pořadí oproti pořadí při uložení. Pro korektní aktualizaci by se tedy muselo iterovat přes veškeré atributy v databázi a buď je nahradit správným nově ukládaným atributem, nebo je smazat. Nalezení původního atributu, který by se měl nahradit, je dále komplikováno možností výskytu reference na zanořenou anotaci v atributu. Najít původní atribut se zanořenou anotací není možné, protože v nově ukládané anotaci se může vyskytovat více atributů se zanořenou anotací a nově ukládané atributy nemají identifikační čísla. Bylo nemožné rozpoznat původní atribut, proto jsem zvolil nový typ aktualizace. Všechny původní atributy se při aktualizaci smažou. Nové atributy se poté uloží do původní anotace, která už nebude obsahovat žádný atribut a nedojde k žádné duplikaci, nebo špatnému nahrazení atributu. Při aktualizaci také není nutné procházet všechny původní atributy. Původní metoda staré atributy nahrazovala v databázi novými, ale vzhledem ke špatně odvozeným typům a změněnému pořadí nahrazení nefungovalo korektně.

7.3 Návrh opravy změny zanořené anotace

V anotačním serveru bude opravena chyba klienta StoryScope. Klient při změně zanořené anotace poslal změnu pouze v zanořené anotaci. Tato změna je však v rozporu s komunikačním protokolem. Správně by se měla poslat změna v anotaci, která není zanořená a změněnou zanořenou anotaci obsahuje v atributu, nebo v některém atributu již zanořené anotace, nebo případně ve větším zanoření. V rámci ukládání změněných anotací je třeba iterovat přes všechny změněné anotace a podívat se, jestli nejsou zanořené. Pokud by některá anotace byla zanořena, musí být nahrazena za svoji hierarchicky nejvíce položenou otcovskou anotaci. Dále musí být místo provedení aktualizace zanořené anotace provedena aktualizace celé hierarchie zanoření.

7.4 Návrh ukládání nabídek anotací

Při vývoji projektu Decipher vznikla potřeba ukládat nabídky anotací. Nabídky vytváří uživatel buď tím, že vytvoří anotaci, nebo je vytváří modul NER, který nabídky generuje. Uživatelé mohou pomocí potvrzení nabídky anotace rozšířit vlastní anotace k textu. Pokud uživatel nabídku potvrdí, vytváří se zpětná vazba nabídky a zapíše se do ní uložená anotace podle nabídky, aby se nabídka příště již nezobrazila. Pokud uživatel nabídku odmítne, je také vytvořena zpětná vazba, ale anotace v ní není vyplněna. Nabídka se opět příště nezobrazí. Datová struktura nabídky mimo jiné obsahuje položku `confidence`, která určuje důvěryhodnost nabídnuté anotace. Pokud více uživatelů potvrdí nabídku, její důvěryhodnost roste. Pokud ji odmítnou, důvěryhodnost naopak klesá.

Nabídky se ukládají ve třídě `SugPersister`. Koncepce třídy byla převzata ze třídy `Persister`. Třída zpracovává požadavek na uložení přidaných a změněných nabídek anotací. Před vlastním uložením nabídek se uloží nové typy anotací. Uložení proběhne buď podle přidaných a změněných anotací, nebo podle generovaných nabídek modulem NER a je více rozepsáno v sekcích 7.4.1 a 7.4.2.

Po uložení všech nabídek se nahradí odkazy na nabídky v již vytvořených nabídkách anotací. Z databáze se vyberou všechny uložené nabídky, které obsahují odkaz na některou z nově uložených nabídek. Odkaz je poté nahrazen správnou referencí v rámci serveru a správným identifikačním číslem v databázi.

7.4.1 Ukládání nabídek anotací podle vytvořených anotací

Pokud se vytvoří anotace, automaticky se vytvoří nabídka anotace. Nabídka je vytvořena na základě obsahu vytvořené anotace. Při jejím ukládání je nutné nejprve uložit nabídku anotace bez atributů s odkazem. V rámci ukládání se převádí atributy se zanořenou anotací na atributy nabídky se zanořenou nabídkou a nastavuje se důvěryhodnost ukládané nabídky. Zpětná vazba ukládané nabídky se při ukládání aktualizuje pro uživatelskou skupinu, ve které je uživatel, který vytvořil anotaci, ze které se vytvořila nabídka. Uživatelská skupina bude mít nabídku potvrzenou, aby se jí nezobrazila zároveň s vytvořenou anotací.

Aktualizace nabídky proběhne, pokud uživatel potvrdí nabídku. Je tím navýšena její důvěryhodnost. Přidává se také nová zpětná vazba k vytvořené anotaci a potvrzené nabídce.

Pokud uživatel změní anotaci, aktualizace vytváří novou nabídku. Již uložené nabídce se zruší zpětná vazba a nastaví se její důvěryhodnost stejně jako při odmítnutí. Nově vytvořené nabídce se zpětná vazba zaktualizuje.

Po uložení všech nabídek je třeba přidat atributy s odkazem k právě uloženým nabídkám. Iteruje se přes vytvořené a změněné anotace. Pokud má některá anotace atribut s odkazem na anotaci, je tento atribut převeden na atribut nabídky anotace a přidán k nabídce vytvořené z anotace.

7.4.2 Ukládání nabídek anotací generovaných modulem NER

Před ukládáním nabídek je využito metod pro vyhledávání a aktualizaci fragmentů. Uložené nabídky jsou znovu vyhledány v dokumentu. Pokud se aktualizovaná nabídka nachází i v nabídkách generovaných modulem NER, je předána k aktualizaci fragmentů. Pokud aktualizovaná nabídka není nalezena, záleží na její zpětné vazbě. Pokud nemá zpětnou vazbu, nabídce je snížena důvěryhodnost na 0 % a již není zobrazena. Pokud má zpětnou vazbu, s nabídkou se nic neděje, protože ji některý uživatel již potvrdil, nebo se jedná o nabídku vytvořenou z anotace. Pokud by se snižovala důvěryhodnost nenalezených nabídek se zpětnou

vazbou, snížila by se tím funkčnost nabízení anotací. Nabídky vytvořené z anotací by se při další aktualizaci nabídek od modulu NER přestaly zobrazovat.

Zbytek nenalezených nabídek generovaných modulem NER je uložen jako nové nabídky, které ještě nebyly potvrzeny.

Při ukládání nabídek generovaných modulem NER jsou již všechny nabídky uloženy, nebo součástí požadavku. Může se proto kontrolovat pouze atribut s odkazem. Pokud se vyskytuje v nabídce atribut s odkazem, uloží se nejprve odkazovaná nabídka a poté nabídka, ve které se nacházel odkaz.

Kapitola 8

Implementace algoritmů

Kapitola s implementací popisuje řešení některých částí 4A serveru. Implementace proběhla podle návrhu popsaného v kapitolách s návrhem. Kapitola je rozdělena na část implementace vylepšení aktualizace fragmentů a na část s implementací navrženého rozšíření.

8.1 Implementace vylepšení aktualizace fragmentů

Obsah této sekce je zaměřen na implementaci aktualizace fragmentů.

8.1.1 Dvousměrný iterátor - rozšíření třídy iterátor

Původní iterátory byly navrženy, aby iterace v DOM stromu anotovaného dokumentu probíhala hierarchicky po všech uzlech v jednom směru (kromě statického iterátoru XPath). Tomu odpovídal i návrh metod abstraktní třídy `NodeIterator`, od které jsou všechny iterátory odvozené. Třída obsahuje pouze dvě metody. Jednu k inicializaci podle XPath počátečního uzlu a druhou k navrácení dalšího uzlu. Dle návrhu bylo třeba implementovat metodu, která navrací další uzel tak, aby procházela strom dvěma směry a zároveň byla kompatibilní s již používaným rozhraním. Do třídy byla přidána proměnná, která určuje aktuální směr průchodu stromem. Průchod může být doleva nebo doprava. Podle toho, jestli se od původního uzlu iteruje ve směru předchozích uzlů, nebo dalších uzlů. Iterace je implementována ve dvou vlastních větvích, které řeší přístup k dalším uzlům. Směr se při každém zavolání metody změní a iteruje se na opačnou stranu. Do metody byly mimo jiné přidány další dvě proměnné indikující, že v daném směru již není žádný další uzel. Vzhledem k těmto proměnným bylo možné vytvořit metodu na zjištění, jestli daný iterátor ještě může navrátit další uzel. Tato metoda je využita v některých sofistikovanějších metodách průchodu dokumentem.

8.1.2 Vyhodnocení pozitivního porovnání při průchodu uzlem po slovech

Průchod textovým uzlem po slovech byl v návrhu vylepšen o rozšíření porovnávaného textu, kdy se můžou přidávat ke konci porovnávaného textu další slova, dokud je text nového fragmentu při porovnání vyhodnocen jako dostatečně shodný původnímu fragmentu. Text potenciálního fragmentu je vzhledem k této skutečnosti při průchodu prodlužován na nejdelší možný řetězec, který je dostatečně shodný s původním fragmentem. Finální text je vzápětí předán metodě `selectBestPart`, která text zkrátí, aby text se co nejvíce podobal původnímu fragmentu. Text je zkracován nejprve ze začátku, protože při prvotním průchodu

se přidávalo ke konci, ale začátek zůstal stále stejný. Text je zkracován po znaku střídavě z obou stran textu. Pokud je po zkrácení text stále dostatečně shodný s původním fragmentem, je přidán do pole s nejlepšími částmi. Zkracuje se, dokud je výsledek porovnání Levenshteinovou metodou lepší než v předchozí iteraci. Na konci metody je pole s nejlepšími částmi předáno k vyhodnocení podle textu, offsetu a XPath. Z pole se vybere jeden nejvíce podobný fragment vůči původnímu fragmentu.

8.1.3 Sdílení prvního a posledního slova v metodách průchodu

Z důvodu větší efektivity iterace přes textový uzel s porovnáním prvního/posledního slova a posledního/prvního znaku se musí nejprve provést iterace přes daný textový uzel s porovnáním prvního a posledního slova. V Iteraci s porovnáním prvního a posledního slova se uloží do pole shodná první a poslední slova. V další iteraci s porovnáním prvního/posledního slova a posledního/prvního znaku se již neiteruje přes celý text, ale pouze přes části textu, kde se nacházely shodná první a poslední slova v první iteraci. Druhá iterace nastane pouze tehdy, když se v první iteraci nenajde dostatečně shodný fragment vůči původnímu fragmentu.

8.2 Implementace rozšíření diplomové práce

Obsah této sekce je zaměřen na implementaci navrženého rozšíření.

8.2.1 Aktualizace atributů se zanořenými anotacemi při aktualizaci anotace

V návrhu rozšíření se změnil způsob aktualizace atributů. Nově se atributy mažou z databáze a jsou nahrazeny novými atributy. Vyřešila se tím chyba duplikace zanořených anotací na serveru a zároveň se tím odstranil problém s EclipseLink při ukládání. Při nahrazování atributů nastává komplikace při aktualizaci atributu se zanořenou anotací. Před smazáním atributu je nutné odebrat z atributu referenci na zanořenou anotaci, protože jinak by se smazala i zanořená anotace, což je špatně. Zanořená anotace se má zaktualizovat stejně jako nezanořená. Pokud by se smazala a byla znovu uložena do databáze, narušilo by se provázání anotací přes odkazy. Reference na zanořené anotace se při mazání atributů uloží do asociativního pole, kdy je jako klíč zvoleno identifikační číslo zanořené anotace. Následně se při uložení nových atributů kontroluje, jestli identifikační číslo zanořené anotace v ukládaném atributu není v poli. Pokud by bylo, je provedena aktualizace a nahrazení anotace, jinak se uloží jako nová anotace. Nezaktualizované anotace v poli se smažou z databáze, protože je uživatel odstranil.

8.2.2 Oprava typu objektu třídy popisující atribut

Jak již bylo zmíněno v kapitole návrhu rozšíření, ve 4A serveru se používá EclipseLink, což je rozšiřitelný framework pro jazyk Java, který zajišťuje služby pro ukládání objektů do více možných typů úložišť. Ve 4A serveru se používá pro objektově-relační mapování a přístup do databáze přes Java Persistence API 2.1. Konkrétní verze použitá na serveru vykazovala chybu, se kterou se v návrhu nepočítalo a bylo ji třeba v implementaci opravit. Pokud se do databáze ukládal objekt třídy, která byla odvozena od jiné nadřazené třídy, objekt se správně uložil, ale jeho opětovné načtení z databáze vykazovalo v některých

případech chybu. Konkrétně se jednalo o objekt třídy odvozené z třídy `BaseAttribute`. Odvozené třídy neobsahují žádné další vlastnosti, které by bylo třeba ukládat navíc, proto jsou všechny odvozené objekty ukládány do stejné databázové tabulky. V případě, že se objekt načetl jako instance jiné odvozené třídy, bylo s ním manipulováno špatně. Zejména pokud se jednalo o atributy obsahující zanořenou anotaci, nebo odkaz na jinou anotaci, nesprávná manipulace duplikovala fragmenty zanořených anotací a zanořené anotace jako celek. Bylo proto nutné vytvořit kontrolu správnosti instancí tříd atributů anotací a atributů nabídek anotací při ukládání do databáze a nahrávání z databáze. Ve třídách `AttributeManager` a `SugAttributeManager` je metoda `hasAttributeRightInstance` pro kontrolu instance a metoda `changeAttributeInstance`, která přetypuje objekt na jeho správnou instanci. Kontrola a přetypování jsou založeny na vlastnosti atributů `simpleType`, která určuje typ atributu, a je ukládána do databáze.

Kapitola 9

Testování úspěšnosti implementovaných algoritmů

V kapitole testování je otestováno navržené řešení aktualizace fragmentů anotace, detekce míry ovlivnění a také navržené rozšíření diplomové práce.

9.1 Testování aktualizace fragmentů anotace

V této sekci je otestováno nastavení skupin pro vyhledávání a aktualizaci změněných fragmentů. Nastavení skupin je uvedeno v tabulce 9.1. Metoda průchodu po slovech a metody s porovnáním prvního slova jsou nastaveny na možné zvětšení fragmentu na 144 % a zmenšení na 56 %.

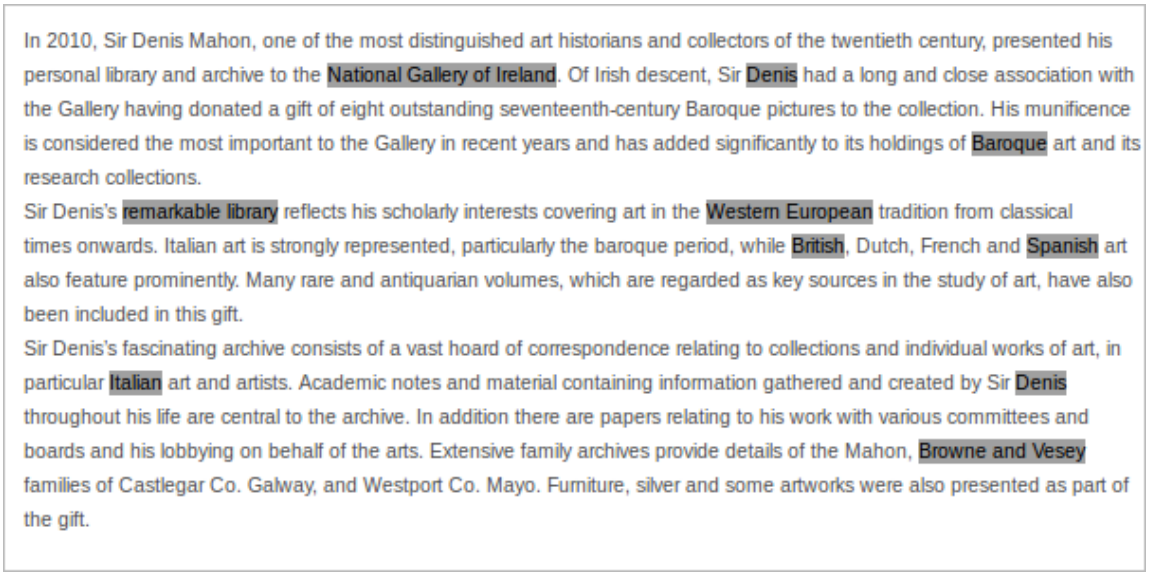
Číslo	Porovnávací metoda	Metoda průchodu
1	Přesná shoda	Dvousměrný průchod po znacích
2	Levenshteinova metoda (56 %)	Průchod po slovech
3	Levenshteinova metoda (83 %)	Tři metody s prvním slovem

Tabulka 9.1: Inicializační metody pro skupinu s dvousměrným iterátorem a skupinu se sekvenčním iterátorem

9.1.1 Specifikace dokumentu a anotací pro testy

Metodika testů bude ukázána na dokumentu s názvem „Sir Denis Mahon“, který je ve výchozím nastavení dostupný přes uživatelské rozhraní StoryScope. Dokument obsahuje čtyři textové uzly v jednom odstavci. Pro účely testování bude později rozšířen o více odstavců. Vytvořené anotace pro testování jsou uvedeny na obrázku 9.1. Souhrn použitých anotací i s jejich vlastnostmi je uveden v tabulce 9.2. Anotace „**remarkable library**“ a „**Browne and Vesey**“ byly vybrány manuálně. Ostatní anotace jsou potvrzené nabídky anotací. Rozptyl anotací v textu byl zvolen rovnoměrně, protože i většina textů v projektu měla nabídnuté anotace rozložené rovnoměrně.

Před provedením testů jsem do implementace přidal výpis nalezených výsledků do konzoly. Výpis jsem upravil tak, aby se mi při nalezení fragmentu například metodou číslo 2 vypsaly výsledky i dalších metod (1 a 3). Vyhledávání tedy neskončilo první úspěšnou metodou a vypsaly se výsledky vyhledávání všech metod.



Obrázek 9.1: Dokument a anotace použité v testu

Fragment	XPath	Offset	Délka
National Gallery of Ireland	/HTML[1]/BODY[1]/P[1]/text()[1]	162	27
Denis	/HTML[1]/BODY[1]/P[1]/text()[1]	213	5
Baroque	/HTML[1]/BODY[1]/P[1]/text()[1]	495	7
remarkable library	/HTML[1]/BODY[1]/P[1]/text()[2]	12	18
Western European	/HTML[1]/BODY[1]/P[1]/text()[2]	84	16
British	/HTML[1]/BODY[1]/P[1]/text()[3]	91	7
Spanish	/HTML[1]/BODY[1]/P[1]/text()[3]	118	7
Italian	/HTML[1]/BODY[1]/P[1]/text()[4]	142	7
Denis	/HTML[1]/BODY[1]/P[1]/text()[4]	246	5
Browne and Vesey	/HTML[1]/BODY[1]/P[1]/text()[4]	481	16

Tabulka 9.2: Anotace vytvořené v dokumentu pro testování

9.1.2 Připsání textu do textového uzlu

Připsání textu ovlivňuje fragmenty v rámci jednoho textového uzlu. V tomto testu bude přidán text do prvního textového uzlu:

„In 2010, Sir Denis Mahon, one of the most distinguished art historians and collectors of the twentieth century, presented his **nice** personal library and archive to the *National Gallery of Ireland*.“

Jelikož šlo o změnu v rámci uzlu a byl dostupný uzel s původním XPath, všechny fragmenty nalezla skupina s dvousměrným iterátorem. Ovlivněny byly všechny fragmenty z prvního textového uzlu jediného odstavce. Výsledky vyhledávání jsou uvedeny v tabulce 9.3.

Metody s čísly 2 a 3 připojily k fragmentu „**National Gallery of Ireland**“ tečku za větou, protože skládají fragmenty z posloupnosti slov. Tečka není odebrána, protože mohlo jít o zkrácení jména a přidání tečky. V testech tato změna není považována za chybu.

Fragmenty „**Baroque**“ a „**Denis**“ našla metoda číslo 2 v textu dvakrát, ale byl zvolen původní odsunutý fragment. Metoda číslo 3 nenalezla jednoslovné fragmenty, protože hledá minimálně dvě slova.

Tento typ změny je nalezen v jakémkoliv provedení, pokud není narušen některý z fragmentů, nebo se změnou nezmění i XPath uzlu odsunutím části textu do dalšího odstavce, případně do dalšího textového uzlu.

Fragment	Číslo metody	Nalezeno
National Gallery of Ireland	1	National Gallery of Ireland
National Gallery of Ireland	2	National Gallery of Ireland.
National Gallery of Ireland	3	National Gallery of Ireland.
Denis	1	Denis
Denis	2	Denis
Denis	3	nenalezeno
Baroque	1	Baroque
Baroque	2	Baroque
Baroque	3	nenalezeno

Tabulka 9.3: Výsledek testu připsání textu do textového uzlu

9.1.3 Zkrácení fragmentu

Nalezení zkráceného fragmentu je závislé na typu změny. Pokud se fragment zkrátí beze změny prvního a posledního slova, je vysoká pravděpodobnost, že bude nalezen.

Testovaná změna 1:

„**National Gallery of Ireland**“ => „**National Ireland**“
Nalezeno: „**National Ireland.**“
Číslo použité metody: 3 (první a poslední slovo)
Použitý iterátor: dvousměrný

Na nalezení této změny má vliv nastavení možnosti prodloužení/zkrácení fragmentu u průchodu textem s výběrem fragmentu se shodným prvním a posledním slovem. V tomto případě šlo o zkrácení na 63 % původní délky. Pokud by nastavení bylo méně tolerantní, fragment by nebyl nalezen.

Testovaná změna 2:

„**remarkable library**“ => „**remarkable**“
Nalezeno: „**remarkable**“
Číslo použité metody: 2 (průchod po slovech)
Použitý iterátor: dvousměrný

9.1.4 Vepsání textu do fragmentu

Vepsání textu má podobné vyhodnocení jako zkrácení fragmentu. Pokud se dopisuje do fragmentu se zachováním prvního a posledního slova, fragment je ve většině případů nalezen.

Pokud by se připisoval text před fragment nebo za fragment a nový text by od fragmentu byl oddělen mezerou, metody by takovou změnu ignorovaly, protože by nedošlo k ovlivnění řetězce fragmentu.

Testovaná změna:

„**remarkable library**“ => „**remarkable epic library**“
Nalezeno: „**remarkable epic library**“
Číslo použité metody: 3 (první a poslední slovo)
Použitý iterátor: dvousměrný

Zde opět záleží na nastavení možnosti prodloužení/zkrácení fragmentu. Fragment byl prodloužen o 28 %.

9.1.5 Nahrazení fragmentu jiným textem

Při nahrazení záleží na podobnosti textu, kterým byl fragment nahrazen, a jestli se změnilo první a poslední slovo. Pokud není změněno první a poslední slovo, je fragment nalezen v závislosti na velikosti nastaveného povoleného zvětšení. V případě záměny posledního slova i písmena fragmentu lze fragment nalézt pomocí metody porovnávající první slovo a největší shodu.

Testovaná změna 1:

„**National Gallery of Ireland**“ => „**National Gallery of Irsko**“
Nalezeno: „**National Gallery of Irsko.**“
Číslo použité metody: 2 (průchod po slovech), 3 (první slovo a maximální shoda)
Použitý iterátor: dvousměrný

Změnilo se 19 % fragmentu. Fragment by našly obě zmíněné metody.

Testovaná změna 2:

„**National Gallery of Ireland**“ => „**National galerie z Irska**“
Nalezeno: „**National galerie z Irska.**“
Číslo použité metody: 2 (průchod po slovech), 3 (první slovo a maximální shoda)
Použitý iterátor: dvousměrný

Změnilo se 37 % fragmentu. Fragment by našly obě zmíněné metody.

Testovaná změna 3:

„**Spanish**“ => „**Spain**“
Nalezeno: „**Spain**“
Číslo použité metody: 2 (průchod po slovech)
Použitý iterátor: dvousměrný

Změnilo se 43 % fragmentu. V případě méně unikátního fragmentu v rámci textového uzlu je tu však riziko, že by došlo k nalezení špatného fragmentu. Méně unikátní fragmenty jsou testovány dále v odstavci s názvem Test duplicitních fragmentů.

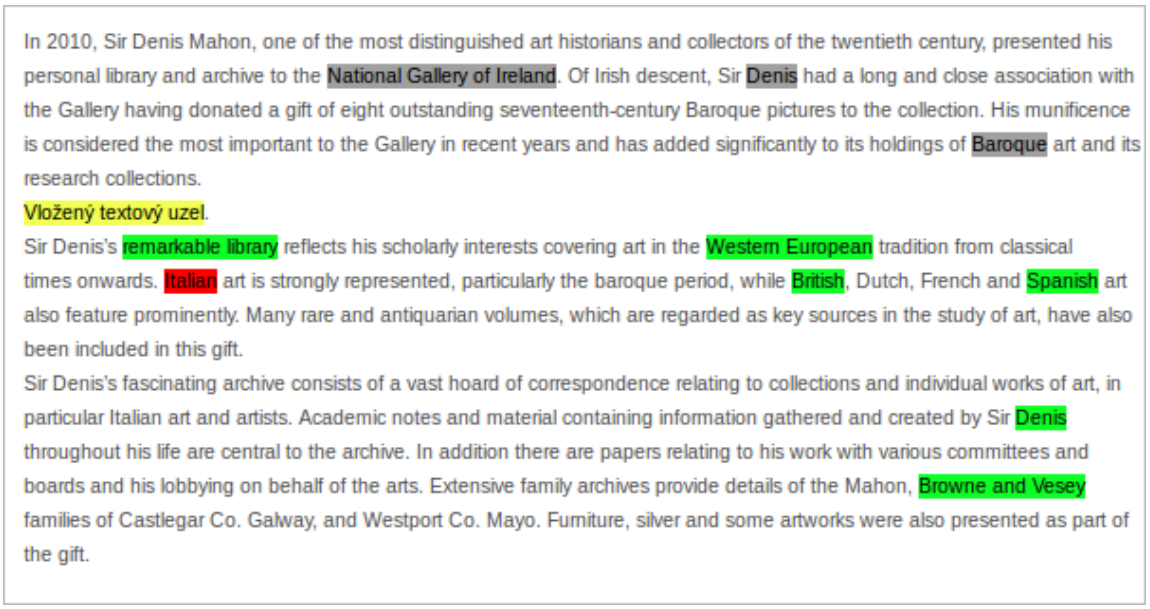
Testovaná změna 4:

„**Western European**“ => „**Eastern Asian**“
Nenalezeno
Číslo použité metody: 1,2,3
Použitý iterátor: nejdříve dvousměrný, poté sekvenční

Změnilo se 50 % fragmentu a fragment byl správně odmítnut. Šlo již o velkou změnu textu, která změnila i význam.

9.1.6 Přesunutí fragmentu do jiného textového uzlu

Posun v XPath ovlivňuje rozhodnutí o použitém iterátoru. Pokud není dostupný žádný uzel na původním XPath, použije se sekvenční iterátor. Jinak je vždy použit dvousměrný iterátor. Žlutě vyznačený text je provedená změna v dokumentu. Červeně vyznačený text jsou špatně nalezené anotace. Zeleně vyznačený text jsou správně nalezené anotace. Vyznačený text šedou barvou udává změnou neovlivněné anotace. Testovaná změna je na obrázku 9.2.


In 2010, Sir Denis Mahon, one of the most distinguished art historians and collectors of the twentieth century, presented his personal library and archive to the National Gallery of Ireland. Of Irish descent, Sir Denis had a long and close association with the Gallery having donated a gift of eight outstanding seventeenth-century Baroque pictures to the collection. His munificence is considered the most important to the Gallery in recent years and has added significantly to its holdings of Baroque art and its research collections.
Vložený textový uzel.
Sir Denis's remarkable library reflects his scholarly interests covering art in the Western European tradition from classical times onwards. Italian art is strongly represented, particularly the baroque period, while British, Dutch, French and Spanish art also feature prominently. Many rare and antiquarian volumes, which are regarded as key sources in the study of art, have also been included in this gift.
Sir Denis's fascinating archive consists of a vast hoard of correspondence relating to collections and individual works of art, in particular Italian art and artists. Academic notes and material containing information gathered and created by Sir Denis throughout his life are central to the archive. In addition there are papers relating to his work with various committees and boards and his lobbying on behalf of the arts. Extensive family archives provide details of the Mahon, Browne and Vesey families of Castlegar Co. Galway, and Westport Co. Mayo. Furniture, silver and some artworks were also presented as part of the gift.

Obrázek 9.2: Vložení textového uzlu do dokumentu

Vložení nového textového uzlu mezi první a druhý textový uzel ovlivnilo všechny anotace z původního druhého textového uzlu a uzlů pod ním. Celkem šlo o 7 anotací. Správně nalezeno bylo 6 anotací. Anotace s fragmentem „**Italian**“ byla nalezena ve špatném uzlu, protože na původním XPath se vyskytoval stejný fragment. 86 % anotací však bylo po odsunutí vyhodnoceno správně.

9.1.7 Test duplicitních fragmentů

V prvním textovém uzlu se nachází 2x slovo „**Denis**“. Slovo s větším offsetem je součástí fragmentu anotace. V takovém případě existuje nebezpečí záměny fragmentu při textové změně fragmentu.

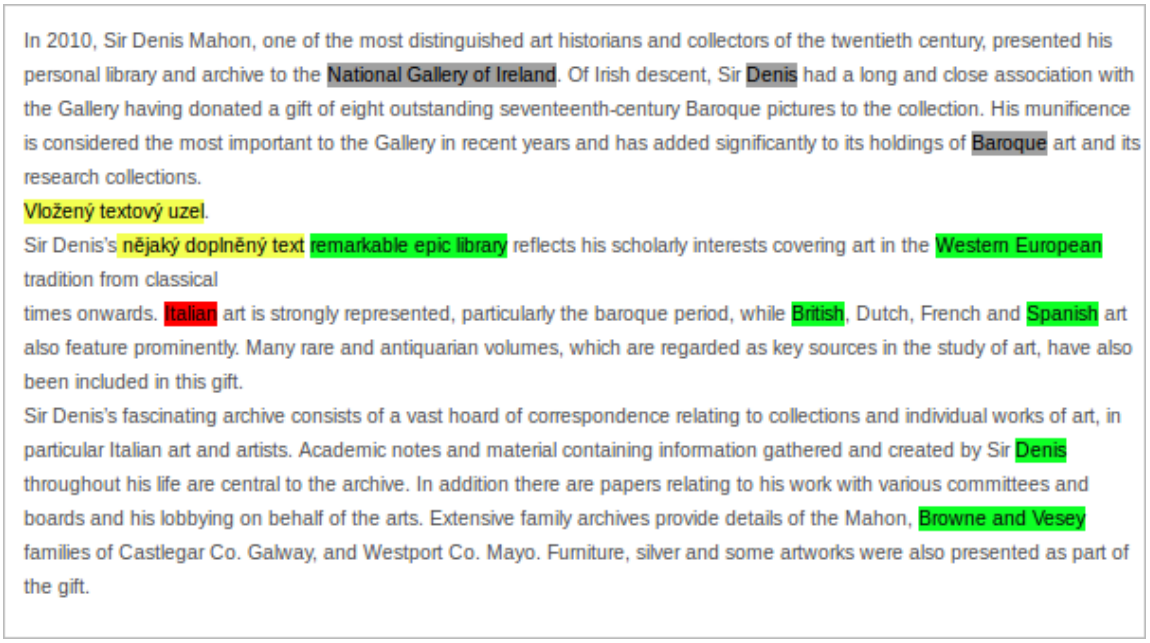
Testovaná změna:

„**Denis**“ => „**Denim**“
Nalezeno: Denim
Číslo použité metody: 2 (průchod po slovech)
Použitý iterátor: dvousměrný

Změnilo se 20 % fragmentu. V tomto případě byla změna offsetu vyhodnocena jako horší než textová změna. V případě větší změny by již došlo k záměně fragmentů.

9.1.8 Složení více jednoduchých změn

V tomto testu bude ukázáno posunutí fragmentu do jiného textového uzlu v kombinaci s textovou změnou fragmentu a posunutím offsetu v rámci uzlu. Barva vyznačených textů má význam jako u testu 9.1.6. Testovaná změna je na obrázku 9.3.



Obrázek 9.3: Kombinace změn

Fragment „**remarkable library**“ byl odsunut v rámci XPath, offsetu a změněn na „**remarkable epic library**“. Fragment byl v pořádku nalezen. Vložení textového uzlu ovlivnilo XPath fragmentů z původního druhého až posledního uzlu. Ze sedmi ovlivněných anotací bylo správně nalezeno 6. Nastala však stejná chyba jako při vložení uzlu bez změny v rámci uzlu. Fragment „**Italian**“ byl nalezen ve špatném uzlu.

9.1.9 Závěry testů

V případě ubraní kterékoliv metody ze skupin by se množina nalezených potenciálních fragmentů zmenšila. Při odebrání více tolerantních metod by nebyl v některých případech fragment vůbec nalezen. Naopak při odebrání méně tolerantních metod by byl původní fragment nalezen ve většině případů. Ovšem při malých změnách, jako je například posunutí offsetu o pár znaků, by se vyhodnocovalo zbytečně mnoho potenciálních fragmentů a prohledával by se zbytečně velký úsek textového uzlu.

Inicializační konstanty použité ve skupinách jsou statisticky určené pro projekt Decipher. Při nasazení serveru na jiný projekt by bylo třeba zjistit úspěšnost nastavených metod a případně konstanty optimalizovat na konkrétní dokumenty a anotace.

9.2 Zvolení nejvíce shodného fragmentu vůči původnímu fragmentu

Následující test je zaměřen na vyhodnocení více nalezených fragmentů při aktualizaci. Dělí se na dvě části: vyhodnocení nalezených fragmentů v rámci více uzlů a vyhodnocení nalezených fragmentů v rámci jednoho uzlu, které se od první metody liší tím, že se neporovnává XPath fragmentu. Test bude proveden opět s dokumentem „Sir Denis Mahon“. Testy byly provedeny v rámci testů aktualizace fragmentů. Zeleně vyznačený fragment v tabulce je vybraný fragment z pole.

9.2.1 Vyhodnocení nalezených fragmentů v rámci více uzlů

Vyhodnocení proběhlo pro nalezené potenciální fragmenty při změně fragmentu popsané v sekci 9.1.4. Vyhodnocovaly se fragmenty uvedené v tabulce 9.4.

Původní fragment
Text: „**remarkable library**“
XPath: `/HTML[1]/BODY[1]/P[1]/text()[2]`
Offset: 12

Fragment	XPath	Offset	Ohodnocení
remarkable epic	/HTML[1]/BODY[1]/P[1]/text()[2]	12	190
personal library	/HTML[1]/BODY[1]/P[1]/text()[1]	126	659
remarkable epic library	/HTML[1]/BODY[1]/P[1]/text()[2]	12	135

Tabulka 9.4: Nalezené potenciální fragmenty při vyhodnocení v rámci více uzlů

9.2.2 Vyhodnocení nalezených fragmentů v rámci jednoho uzlu

Vyhodnocení proběhlo pro nalezené potenciální fragmenty při změně fragmentu popsané v sekci 9.1.7. Vyhodnocovaly se fragmenty uvedené v tabulce 9.5.

Původní fragment
Text: „**Denis**“
XPath: `/HTML[1]/BODY[1]/P[1]/text()[1]`
Offset: 213

9.2.3 Závěry testů

Testované vyhodnocení je použito v případě nalezení více fragmentů. Testy ukázaly, že se změnami překlepů nemá vyhodnocení problém vybrat původní fragment. Při nalezení více podobných fragmentů u změny jednoho slova fragmentu, který se skládá z více než jednoho

Fragment	XPath	Offset	Ohodnocení
Denis	/HTML[1]/BODY[1]/P[1]/text()[1]	13	200
Denim	/HTML[1]/BODY[1]/P[1]/text()[1]	213	100

Tabulka 9.5: Nalezené potenciální fragmenty při vyhodnocení v rámci jednoho uzlu

slova, také nebyl problém. Problém je pouze s fragmenty, jejichž text je obsažen vícekrát ve stejném či jiném uzlu. V případě opravy překlepu je vybrán původní fragment. Pokud je však rozsah změny větší než oprava jednoho znaku, při výběru původního fragmentu záleží na jeho délce a dalších vlastnostech. Nastavení ohodnocení je velmi subjektivní záležitost a při nasazení musí být konzultována s konkrétními uživateli serveru.

9.3 Testování detekce míry ovlivnění anotací

Testování detekce míry ovlivnění anotací se zaměřuje na možnosti změny textu při synchronizaci dokumentu. Použit bude stejný dokument zobrazený na obrázku 9.1 se stejnými anotacemi vypsanými v tabulce 9.2 jako při testování aktualizace fragmentů. Obrázky a tabulka se nachází na začátku předchozí sekce. Testovány budou nejprve jednodušší změny textu a poté kombinace změn. Žlutě a modře vyznačený text je provedená změna v dokumentu. Červeně vyznačený text jsou špatně nalezené anotace. Zeleně vyznačený text jsou správně nalezené anotace. Vyznačený text šedou barvou udává změnou neovlivněné anotace.

9.3.1 Připsání textu do uzlu

Po připsání textu do posledního textového uzlu byly všechny ovlivněné fragmenty nalezeny správně. Byl ovlivněn offset tří fragmentů: „**Italian**“, „**Denis**“, „**Browne and Vesey**“. Práh je nastaven tak, aby při maximální tolerované změně dokumentu umožnil změnu v offsetu všech fragmentů anotací, takže tato změna byla povolena. Dokument po změně je znázorněn na obrázku 9.4.

Výsledná míra ovlivnění: **10 %**
Změna: Ovlivněn offset tří fragmentů.
Synchronizace proběhla bez interakce s uživatelem.

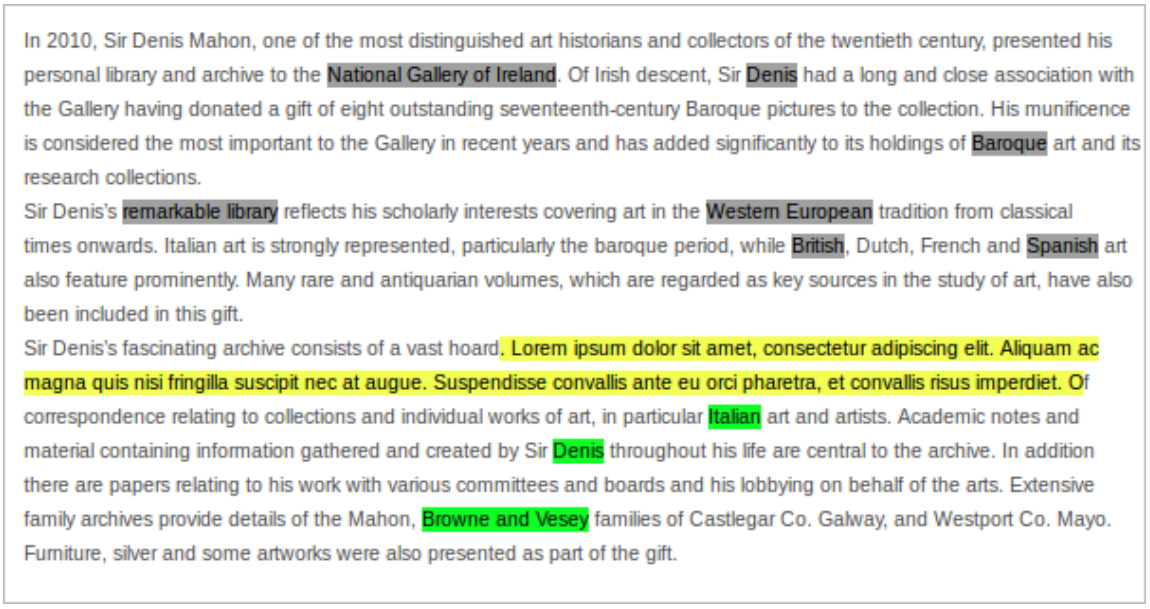
9.3.2 Změna/smazání části textu s anotací v uzlu

Změna části textu zobrazená na obrázku 9.5 způsobila ovlivnění jediného fragmentu anotace „**Western European**“, která byla správně označena za špatnou, protože nebyla nalezena. Smazání stejné části dokumentu bylo vyhodnoceno stejně jako změna.

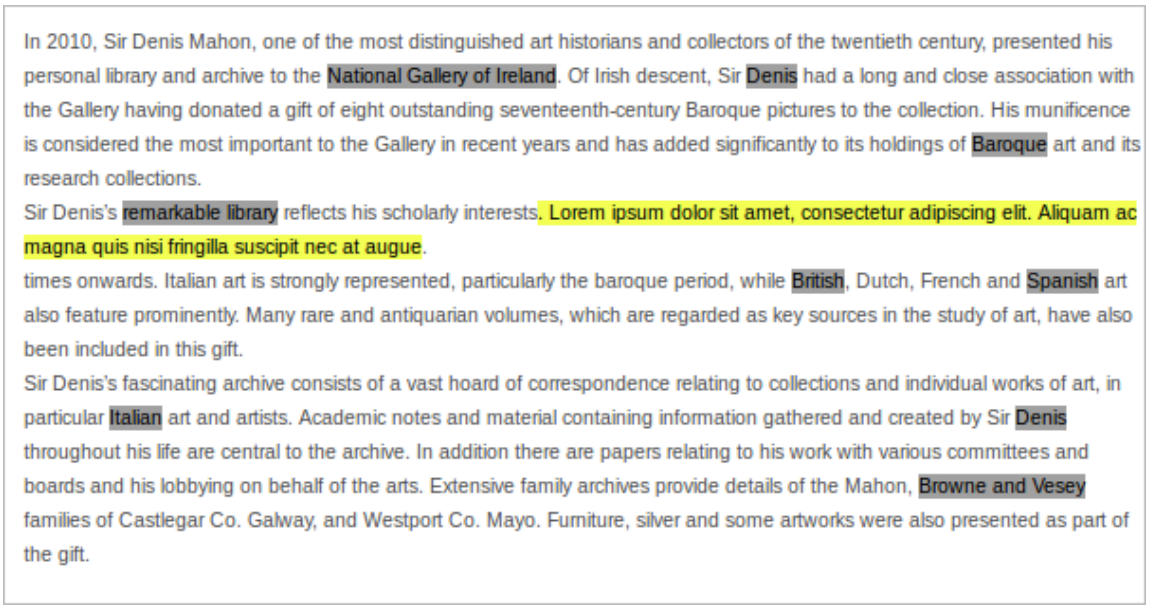
Výsledná míra ovlivnění: **10 %**
Změna: Nenalezeny všechny fragmenty jedné anotace
Synchronizace proběhla bez interakce s uživatelem.

9.3.3 Dopsání textového uzlu

Dopsání textového uzlu mezi druhý a třetí textový uzel ovlivnilo XPath fragmentů: „**British**“, „**Spanish**“, „**Italian**“, „**Denis**“ a „**Browne and Vesey**“. Tato změna je však v toleranci prahu (jedna změna v XPath), takže změny nebyly ohodnoceny. Fragment anotace



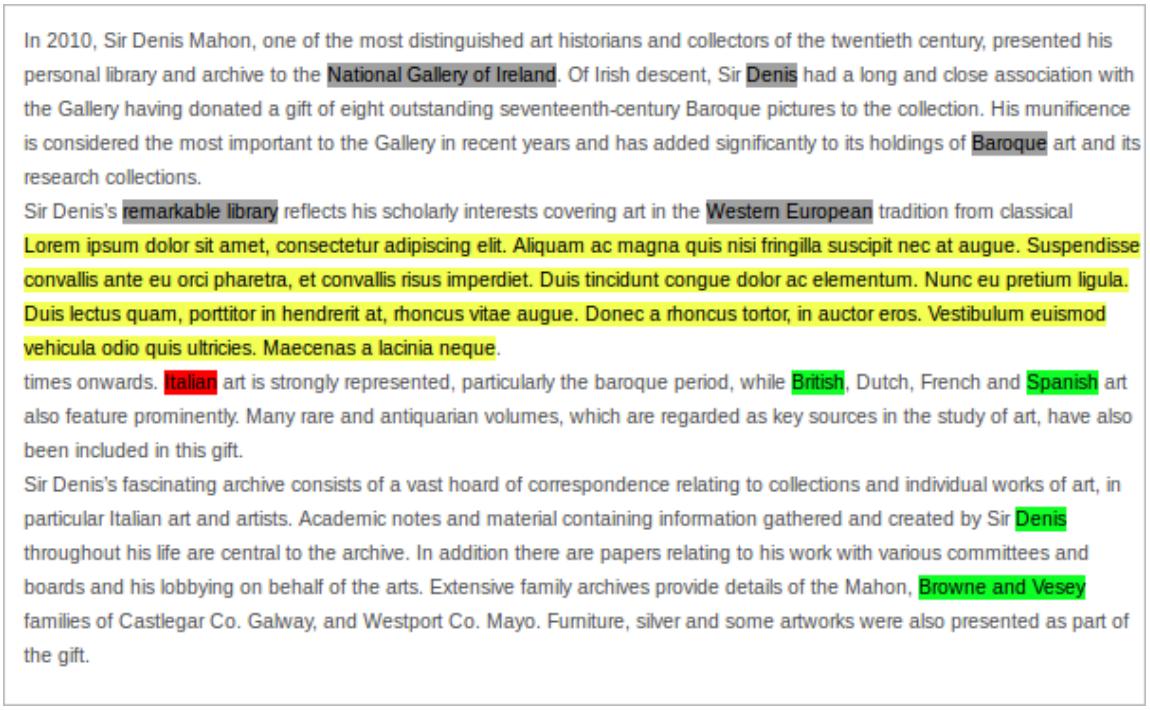
Obrázek 9.4: Připsání textu do uzlu



Obrázek 9.5: Změna/smazání části textu s anotací v uzlu

„**Italian**“ byl zvolen špatně. Tato chyba je způsobena duplicitním fragmentem na XPath původního fragmentu. Ostatní ovlivněné anotace byly zvoleny správně. Dokument po změně je znázorněn na obrázku 9.6.

Výsledná míra ovlivnění: **0 %**
Změna: Odsunutí 50 % fragmentů do dalšího textového uzlu.
Synchronizace proběhla bez interakce s uživatelem.



Obrázek 9.6: Dopsání textového uzlu

9.3.4 Záměna dokumentu

Při nahrazení celého dokumentu za text Lorem Ipsum byla nahlášena chyba synchronizace. Výsledná míra ovlivnění byla 80 %, protože dvě krátké anotace s fragmentem „**Denis**“ se nahradily za podobné slovo „**felis**“. Změna byla v toleranci prahu, proto nebyla započítána. Ostatní anotace nebyly nalezeny. Dokument po změně je znázorněn na obrázku 9.7.

Výsledná míra ovlivnění: **80 %**
Změna: Nahrazení textu dokumentu jiným textem
Synchronizace potřebovala interakci uživatele.

9.3.5 Kombinace 1

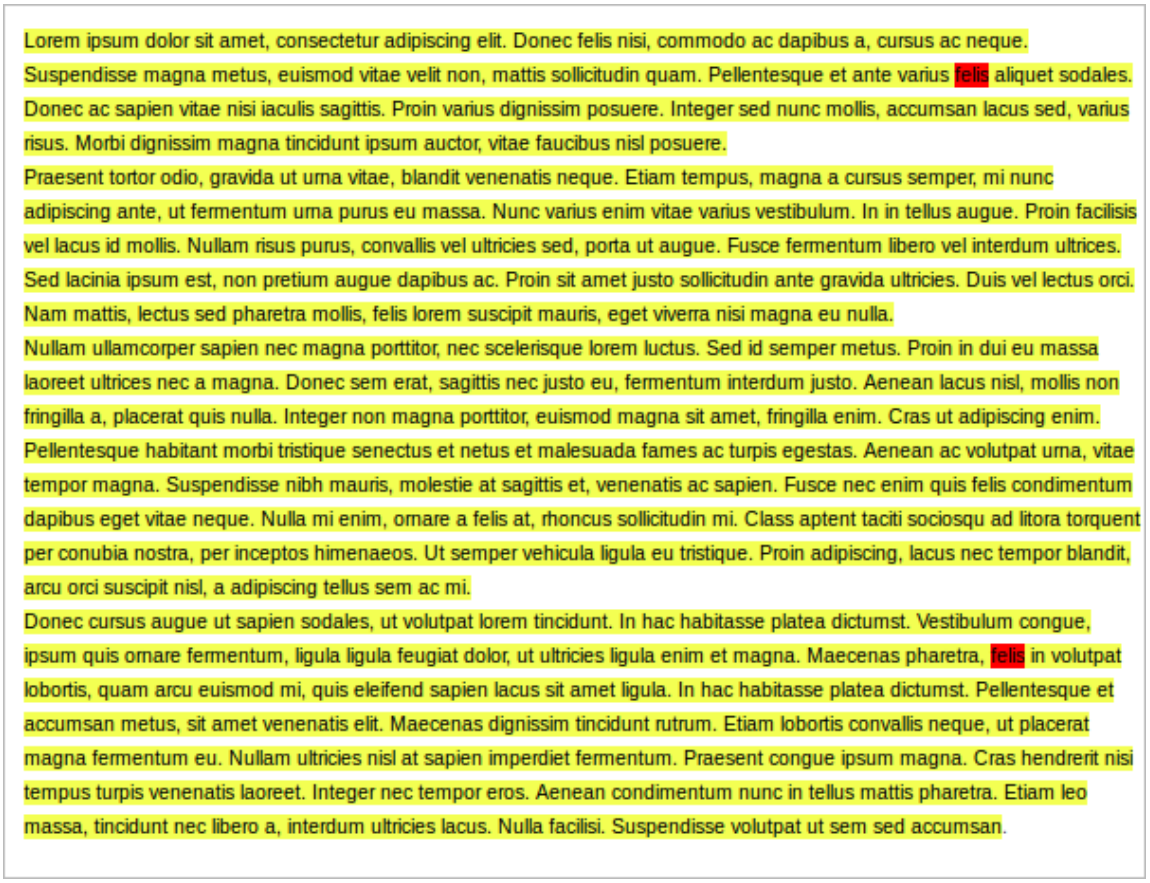
Testovaná kombinace se skládala z nahrazení první věty v prvním uzlu, které ovlivnilo fragment „**National Gallery of Ireland**“. Fragment nebyl nalezen. Změna byla ohodnocena 10 %. Dále byly ovlivněny fragmenty „**Denis**“ a „**Baroque**“, kterým se změnil offset, ale hodnota byla v toleranci prahu, takže změny nebyly ohodnoceny.

Změnu dále tvořilo smazání části textu v druhém uzlu, které smazalo fragment „**Western European**“. Fragment nebyl nalezen a tato změna byla ohodnocena 10 %.

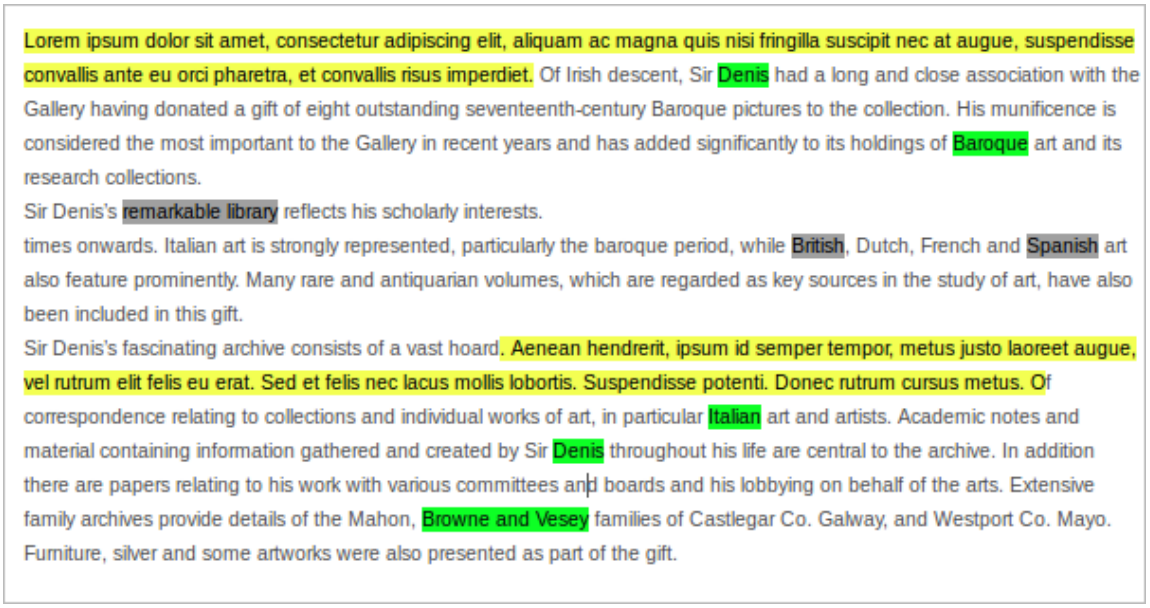
Poslední část změny bylo připsání textu do posledního uzlu, které změnilo offset fragmentů: „**Italian**“, „**Denis**“ a „**Browne and Vesey**“. Změna byla celkem ohodnocena 10 %.

Dokument po změně je znázorněn na obrázku 9.8.

Výsledná míra ovlivnění: **30 %**
Změny: Výše zmíněná kombinace
Synchronizace proběhla bez interakce s uživatelem.



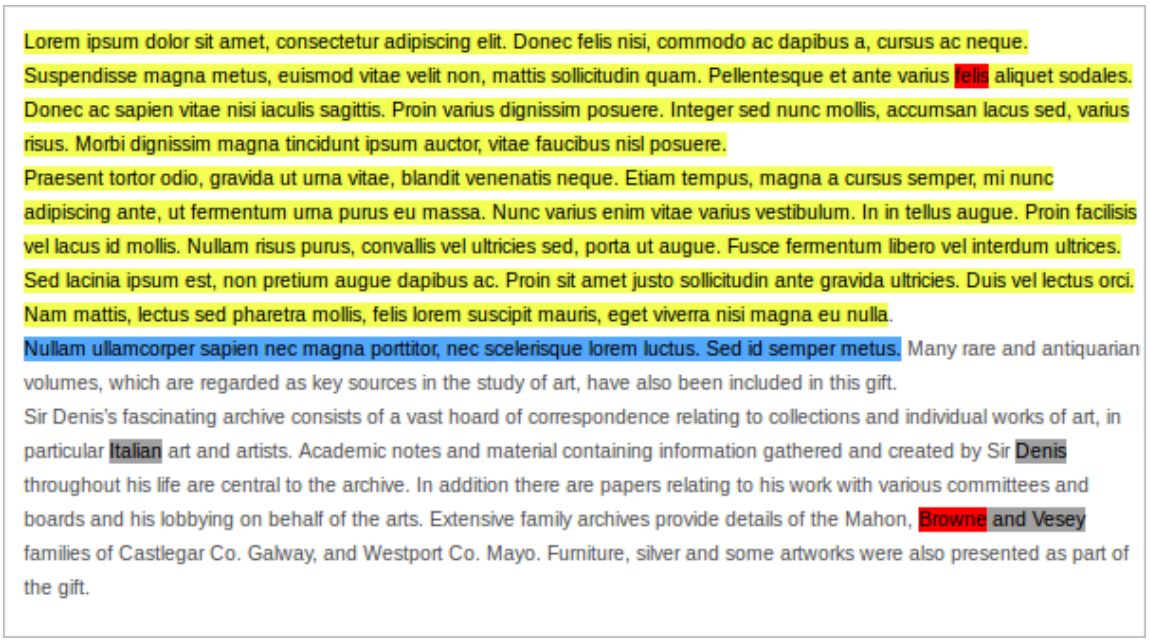
Obrázek 9.7: Záměna dokumentu



Obrázek 9.8: Kombinace 1

9.3.6 Kombinace 2

Druhá testovaná kombinace se skládala z nahrazení prvních dvou textových uzlů a nahrazení dvou vět v druhém uzlu. Změna ovlivnila celkem 7 fragmentů: „**National Gallery of Ireland**“, „**Denis**“, „**Baroque**“, „**remarkable library**“, „**Western European**“, „**British**“ a „**Spanish**“. Pro fragmenty „**Denis**“ a „**Baroque**“ byla nalezena náhrada za podobná slova „**felisý**“ a „**Browne**“. Ostatní ovlivněné fragmenty nebyly nalezeny. Dokument po změně je znázorněn na obrázku 9.9.



Obrázek 9.9: Kombinace 2

Výsledná míra ovlivnění: **50 %**
Změny: Výše zmíněná kombinace
Synchronizace potřebovala interakci uživatele.

9.3.7 Závěry testů

Testy ukázaly, že nastavení míry ovlivnění je velmi tolerantní. Interakce uživatele je k automatické synchronizaci dokumentu vyžadována až při velkých změnách v jeho textu. Nastavení prahu, kdy bude vyžadována interakce uživatele, je však opět velmi subjektivní záležitost a při nasazení musí být konzultována s konkrétními uživateli serveru.

9.4 Testování ukládání a aktualizace anotací a nabídek anotací

Při uložení anotace se automaticky vytváří i nabídka anotace. Pro otestování funkčnosti ukládání anotací a nabídek anotací do databáze stačilo uložit a aktualizovat možné kombinace anotací a atributů. Otestovány byly následující kombinace:

- Anotace bez atributů

- Anotace s každým typem jednoduchého atributu
- Anotace se strukturovaným typem zanořená anotace
- Anotace se strukturovaným typem odkaz na anotaci
- Anotace s každým typem jednoduchého atributu a každým typem strukturovaného atributu (kombinace předchozích tří možností)

Zvlášť byly otestovány různé kombinace odkazů a zanořených anotací. Symbol **A=>B** značí, že anotace A obsahovala atribut s odkazem na anotaci B. Symbol **A|-B** značí, že anotace A obsahovala zanořenou anotaci B. Jednotlivé anotace obsahovaly pouze strukturované atributy. Otestovány byly následující kombinace:

- A=>B=>C
- A=>B, A<=B
- A|-B, A=>B, B=>A
- A=>B=>C|-D
- A|-B|-C, C=>A

V průběhu testování nebyl nalezen žádný problém s ukládáním a aktualizací anotací a nabídek anotací. Po testování byly všechny anotace smazány. Mazání proběhlo úspěšně.

Kapitola 10

Závěr

Při vypracování semestrálního projektu jsem nastudoval knihovny pro hodnocení změn v textovém řetězci a práci s modelem DOM v jazyce Java. Prostudoval jsem dostupný server pro správu anotací vyvíjený v rámci evropského projektu Decipher a zejména části tohoto serveru, které se zabývají aktualizací anotací. Navrhl jsem vylepšení aktualizací pozic anotovaných fragmentů po změně anotovaného dokumentu a detekci míry ovlivnění anotací těmito změnami. Součástí návrhu bylo určení vhodných prahů, kdy již fragment nelze správně nalézt a kdy je anotace změnou zneplatněna. Prahy byly statisticky určeny podle dokumentů dostupných v rámci projektu Decipher a podle znalostní báze nástroje pro rozpoznání pojmenovaných entit využitého v tomto projektu. V neposlední řadě jsem navrhl také algoritmus pro určení, zda je vliv na anotace daného dokumentu natolik závažný, že by aktualizace dokumentu neměla být provedena bez interakce s uživatelem. V rámci diplomové práce jsem navržené řešení implementoval a otestoval.

Vyhodnocení testů výsledného řešení prokázalo, že konstanty jsou zvoleny tak, aby většina anotací vytvořených s využitím znalostní báze byla po pravděpodobných změnách fragmentů nalezena znovu. Testy vyhodnocení míry ovlivnění anotací v dokumentu ukázaly, že automatická oprava anotací nalezne většinu změněných anotací a teprve po změnách více jak poloviny dokumentu je vyžadována interakce uživatele. Testy však také potvrdily důležitost optimalizace konstant pro konkrétní projekt a uživatele.

Diplomovou práci jsem rozšířil o návrh vylepšení a opravu ukládání anotací, které v analýze vykázalo nedostatky. Při vývoji projektu Decipher vznikla potřeba ukládat nabídky anotací, proto byl součástí rozšíření také návrh ukládání nabídek anotací. Navržené rozšíření jsem implementoval a otestoval. Testy prokázaly, že nedostatky byly v implementovaném řešení odstraněny a ukládání nabídek anotací bylo implementováno korektně.

Výsledné řešení aktualizace fragmentů, detekce míry ovlivnění anotací, ukládání anotací a ukládání nabídek anotací je zahrnuto v konečném produktu Evropského projektu Decipher.

Další vývoj by se mohl zaměřit na rozšíření nastavení a otestování konstant pro optimalizování serveru na jiné oblasti použití, než je projekt Decipher. Případně by se mohla aktualizace fragmentů rozšířit o napojení přímo na znalostní bázi, ve které by vyhledávala potenciální změny názvů a komplexně by je porovnávala se změněnými fragmenty.

Literatura

[1] 4A Framework. [Online; navštíveno 29.12.2013].
URL http://knot.fit.vutbr.cz/annotations/

[2] 4A Server Architecture. [Online; navštíveno 10.12.2013].
URL
http://knot.fit.vutbr.cz/decipher/integration/4A_architecture_v4.png

[3] Aloisi, A. J.: Homophones. [Online; navštíveno 31.12.2013].
URL http://www.homophone.com/index.php#Whatisahomophone

[4] Apparao, V.; Byrne, S.; Champion, M.; aj.: Document Object Model (DOM) Level 1 Specification. [Online; poslední modifikace: 01.10.1998; navštíveno 01.03.2014].
URL http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/

[5] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.: Extensible Markup Language (XML) 1.0. [Online; poslední modifikace: 10.02.1998; navštíveno 01.01.2014].
URL http://www.w3.org/TR/1998/REC-xml-19980210

[6] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; aj.: Extensible Markup Language (XML) 1.1 (Second Edition). [Online; poslední modifikace: 29.09.2006; navštíveno 01.01.2014].
URL http://www.w3.org/TR/xml11/

[7] Clark, J.; DeRose, S.: XML Path Language (XPath) Version 1.0. [Online; poslední modifikace: 16.11.1999; navštíveno 01.01.2014].
URL http://www.w3.org/TR/xpath/

[8] DB-Engines Ranking. [Online; poslední modifikace: 01.2014; navštíveno 01.01.2014].
URL http://db-engines.com/en/ranking

[9] Decipher - Objectives. [Online; navštíveno 10.12.2013].
URL http://decipher-research.eu/objectives

[10] DeMichiel, L.; aj.: Java Community Process - Java Persistence 2.1. [Online; poslední modifikace: 05.03.2013; navštíveno 01.01.2014].
URL https://jcp.org/en/jsr/detail?id=338

[11] Dublin Institute of Technology. [Online; poslední modifikace: 13.1.2014; navštíveno 14.01.2014].
URL http://www.dit.ie

[12] Fischer, H.: Alternative to Soundex, Metaphone. [Online; navštíveno 08.01.2014].
URL http://www.sound-ex.com/alternative_zu_soundex

[13] Gosling, J.; Joy, B.; Jr., G. L. S.; aj.: *The Java Language Specification, Third Edition*. Addison-Wesley Professional, 2005, ISBN 0-321-24678-0.

[14] Gusfield, D.: *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, 1997, ISBN 0-521-58519-8.

[15] Hors, A. L.; Hégaret, P. L.; Wood, L.; aj.: Document Object Model (DOM) Level 3 Core Specification. [Online; poslední modifikace: 07.04.2004; navštíveno 01.03.2014].
URL http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/

[16] Irish Museum of Modern Art. [Online; poslední modifikace: 2013 navštíveno 10.12.2013].
URL http://www.imma.ie/en/index.htm

[17] Jurafsky, D.: Minimum Edit Distance. [Online; poslední modifikace: 21.01.2012; navštíveno 08.01.2013].
URL http://www.stanford.edu/class/cs124/lec/med.pdf

[18] LLC, A. S.: Metaphone 3. [Online; navštíveno 28.04.2014].
URL http://www.amorphics.com/index.html

[19] National Gallery of Ireland. [Online; poslední modifikace: 2013; navštíveno 10.12.2013].
URL http://www.nationalgallery.ie

[20] Oracle: *GlassFish Server Open Source Edition Release Notes, Release 4.0*. [online]; poslední modifikace: 05.2013; [cit. 08.01.2014].
URL https://glassfish.java.net/docs/4.0/release-notes.pdf

[21] Oracle: Java EE 6 Standards Support. [Online; poslední modifikace: 2011; navštíveno 08.01.2014].
URL http://docs.oracle.com/cd/E18930_01/html/821-2434/gipkz.html

[22] Oracle: Java Enterprise Edition. [Online; navštíveno 29.12.2013].
URL http://www.oracle.com/technetwork/java/javaee/overview/index.html

[23] Oracle: Java Persistence API. [Online; navštíveno 01.01.2014].
URL http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[24] Oracle: Java SE Update Release Notes. [Online; poslední modifikace: 15.10.2013; navštíveno 08.01.2014].
URL
http://www.oracle.com/technetwork/java/javase/7u-relnotes-515228.html

[25] Oracle: MySQL. [Online; poslední modifikace: 2014; navštíveno 14.01.2014].
URL http://www.mysql.com/

[26] Oracle: MySQL Community Server. [Online; poslední modifikace: 03.12.2013; navštíveno 01.01.2014].
URL http://dev.mysql.com/downloads/mysql/

[27] Pecinovský, R.: *Myslíme objektově v jazyku Java*. Grada Publishing, 2004, ISBN 80-247-0941-4.

[28] Robie, J.; Chamberlin, D.; Dyck, M.; aj.: XML Path Language (XPath) 3.0. [Online; poslední modifikace: 22.10.2013; navštíveno 01.01.2014].
URL http://www.w3.org/TR/xpath-30/

[29] Schildt, H.: *Java 7 Výukový kurz*. Computer Press, 2012, ISBN 978-80-251-3748-2.

[30] Soundex. [Online; poslední modifikace: 30.05.2007; navštíveno 31.12.2013].
URL http://www.archives.gov/research/census/soundex.html

[31] Stapleton, M.; Knoth, P.; Kilfeather, E.; aj.: *Decipher-D-WP5-SSL-D5.1.1-System_Architecture-v1.3-RE*. [online]; poslední modifikace: 07.2012; [cit. 09.12.2013].
URL http://decipher-research.eu/sites/decipherdrupal/files/Decipher-D5.1.1-WP5-SSL-System%20Architecture-PU.pdf

[32] System Simulation. [Online; navštíveno 10.12.2013].
URL http://www.ssl.co.uk/

[33] The Open University. [Online; poslední modifikace: 2013; navštíveno 10.12.2013].
URL http://www.open.ac.uk

[34] Tillman, K.; Lobo, R.: Oracle Buys Sun. [Online; poslední modifikace: 20.04.2009; navštíveno 08.01.2014].
URL http://www.oracle.com/us/corporate/press/018363

[35] VUT v Brně. [Online; poslední modifikace: 2013; navštíveno 10.12.2013].
URL http://www.vutbr.cz