MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

Rozpoznanie a archivácia českého webu mimo národnú doménu

BAKALÁRSKA PRÁCA

Ivan Vlček

Brno, 2008

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Vedúci práce: Ing. Peter Žabička

Poďakovanie

Ďakujem vedúcemu mojej práce, Ing. Petrovi Žabičkovi za ochotu, usmerňovanie a pripomienky a rovnako celému týmu WebArchiv.

Zhrnutie

Cieľom práce bolo navrhnúť a zrealizovať systém pre rozpoznávanie a archiváciu webových informačných zdrojov ako súčasť archivačného systému Heritrix. Systém by mal tieto zdroje čo najpresnejšie a najefektívnejšie automaticky identifikovať a archivovať pre využitie v projekte WebArchiv Národnej knižnice ČR. Programové riešenie systému je implementované v jazyku Java. Program využíva funkcie viacerých projektov. Zvolenou platformou bol Linux s vývojovým prostredím NetBeans.

Kľúčové slová

WebAnalyzer, Heritrix, WebArchiv, archivácia webu, Rozpoznanie webu, Sklizeň

Obsah

Kapitola 1

Úvod

Internet patrí k najúspešnejším objavom ľudstva. Mnohí z nás si v súčasnosti život bez neho nevieme ani predstaviť. Používame ho každodenne v práci, v škole aj doma, pretože ponúka široký sortiment možností pre takmer všetky skupiny ľudí.

Internet sa začal masovo zavádzať a používať v nespočetnom množstve inštitúcií, firiem a iných organizácií, či už verejnoprávnych alebo súkromných. Jeho veľký vplyv sa postupne premieta aj do škôl. Mení sa spôsob a forma výuky, ktorá sa s Internetom stáva interaktívnejšia, kvalitnejšia a zábavnejšia.

Ďalšou možnosťou, ktorú Internet ponúka je umiestňovanie materiálov prostredníctvom webových stránok. Mnoho užívateľov túto službu pokladá za výhodnú a rýchlu, čím sa na Internete každodenne objavujú nové a nové informácie a dokumenty. V posledných rokoch je zaznamenaný obrovský nárast elektronických online zdrojov prístupných iba na Internete. Mnohé z týchto zdrojov sú považované za veľmi hodnotné. Existuje však aj riziko, že tieto informácie budú zmenené alebo dokonca zmazané a navždy stratené. Preto sa musíme pokúsiť takéto informácie zachrániť, aby boli zachované kultúrne umelecké a historické hodnoty, ktoré tieto dokumenty nesú a aby boli prístupné aj pre budúce generácie. Ak sa na Internet pozeráme z tohto hľadiska, tak ho môžeme označiť za studňu ľudskej múdrosti, o ktorú sa musíme starať a nemôžeme dovoliť aby vyschla.

Otázkou uchovávania obsahu Internetu sa zaoberali inštitúcie už pred mnohými rokmi. Medzi najznámejšie patrí neprofitujúca spoločnosť Internet Archive, ktorá bola založená v roku 1996 za účelom vybudovania Internetovej knižnice a poskytnutia permanentného prístupu pre výskumných pracovníkov, historikov, školákov a všeobecnú verejnosť k historickým kolekciam, ktoré existujú v digitálnej podobe. V súčasnosti Internet Archive zahrňuje vo svojich kolekciách texty, audio záznamy, animované obrázky, softwér a webové stránky. Prostredníctvom Internet Archive môžete nájsť webové stránky, ktoré v súčasnosti neexistujú alebo sa môžete pozrieť ako vyzerali vaše obľúbené stránky pred desiatimi rokmi a sledovať ich vývoj až do súčasnosti.

Internet Archive (IA) obsahuje snímky webových stránok už od roku 1996 až po dnes. Aj keď je priemerná doba životnosti stránky 100 dní, Internet Archive archivuje stránky pravidelne každé dva mesiace. Pre posilnenie stability a trvanlivosti archívu je IA zrkadlovaný do Bibliotheca Alexandrina v Egypte. Podľa údajov z 20. februára IA v Bibliotheca Alexandrina zahŕňa webovú kolekciu od roku 1996 po rok 2006, 2000 hodín Egyptského a US vysielania a 1000 archívnych filmov. Toto všetko predstavuje 1.5 PB dát uložených na 880 počítačoch. Archív je plne funkčný a prístupný cez webové stránky projektu IA.

Sprístupnenie archívu je uskutočňované pomocou programu Wayback Machine. Wayback Machine je služba, ktorá Vám umožňuje navštíviť archivované verzie webových stránok. Všetko čo potrebujete spraviť je napísať URL, ktorú požadujete a Wayback Machine Vám vráti zoznam všetkých archivovaných verzií zadanej stránky.

V Českej Republike existuje projekt nazvaný WebArchiv, ktorý sa zaoberá archiváciou a zprístupnením českého webu. Jediné rozhodovacie kritérium pre identifikovanie českého webu funguje na základe domény. Tým je zaručené, že všetky URL s českou doménou budú archivované. Existujú však stránky, ktoré sa nachádzajú na iných doménach ako .cz a sú jednoznačne považované za české. WebArchiv má záujem o tieto stránky a preto je jeden z novších cieľov WebArchivu archivácia českého webu mimo doménu .cz.

Táto práca popisuje systém na rozpoznanie a archiváciu českého webu mimo národnú doménu. Systém na rozpoznanie českého webu je integrovaný do systému Heritrix, ktorý okrem iného tieto identifikované stránky archivuje. Pomocou systému je možné identifikovať bohemikálny(český) zdroj na základe viacerých kritérií, ktoré sú v systéme zakomponované. Za bohemikálny zdroj systém považuje všetky stránky, ktoré spĺňajú určité podmienky. Tieto podmienky môže užívateľ upraviť pred spustením samotného systému, tak aby vyhovovali jeho požiadavkam pre definíciu bohemikálneho zdroja. Podmienky sú vyhodnocované na základe kritérií implementovaných v systéme. Medzi základné patria identifikácia jazyka, identifikácia českých IP adries, identifikácia e-mailov a telefónnych čísel. Každá spracovaná stránka, ktorá je systémom na rozpoznanie identifikovaná ako česká je následne archivovaná systémom Heritrix.

Identifikáciou a archiváciou webu mimo národnú doménu sa zaoberajú aj iné krajiny. Kritéria na identifikáciu národného webu sú vo všetkých prípadoch veľmi podobné.

Portugalský web je definovaný ako množina dokumentov, ktoré sú umiestnené na doméne .PT alebo sú umiestnené na doménach .COM, .NET, .ORG, .TV, napísané v portugalskom jazyku a sú odkazované aspoň z jednej stránky s doménou .PT. Ďalšie kritérium, ktoré používajú je *WHOIS*[12] databáza, podľa ktorej zisťujú z ktorej krajiny pochádza majiteľ webu[13].

Austrálsky web je definovaný ako množina dokumentov umiestnených na doméne .AU. Počas sklizne sa stránky odkazujúce do iných domén kontrolujú pomocou GeoIP databáze, ktorá zistí, či sa stránka fyzicky nachádza v Austrálii[14].

V Dánsku používajú automatický a manuálny prístup na identifikáciu národných stránok. Automatický prístup pracuje na základe GeoIP databázy rovnako ako v Austrálii. Manuálny prístup znamená vyhľadávanie stránok cez Google, ktoré obsahujú dánske názvy, ale nie sú umiestnené na .DK doméne[15].

V Thajsku definujú nasledujúce kritéria pre národný web. Stránka je umiestnená na .TH doméne alebo IP adresa serveru je fyzicky umiestnená v Thajsku (GeoIP) alebo je stránka napísaná v thajskom jazyku.

V Grécku považujú za národný web stránky, ktoré sú napísané v gréckom jazyku alebo sú umiestnené na .GR doméne alebo obsahujú informácie, ktoré sa viažu s Gréckom. Napríklad sa vyhľadávajú slová ako sú Hellas, Greek a Greece[16].

Následná kapitola bližšie popisuje spôsob archivácie v projekte WebArchiv.

Kapitola 2

Projekt WebArchiv

WebArchiv je digitálny archív „českých“ webových zdrojov, ktoré sú tu zhromažď'ované za účelom ich dlhodobého uchovávania. Ochranu a uchovávanie týchto dokumentov zaisťuje od roku 2000 Národná knižnica ČR v spoluprácií s Moravskou zemskou knižnicou a Ústavom výpočtovej techniky Masarykovej univerzity. Pri archivácií a sprístupňovaní webu sa používajú softwérové nástroje vyvinuté organizáciou Internet Archive[1].

2.1 Archivácie WebArchivu

Ideálnym cieľom projektu by bolo archivovať všetky české zdroje, ktoré kedy boli na Internete umiestnené. Z technických dôvodov to však nie je možné a preto WebArchiv používa osvedčené techniky, ktoré sa k tomuto ideálnemu cieľu približujú. V súčasnosti WebArchiv používa tri základné spôsoby archivácie.

Plošné sklizne Cieľom tohoto prístupu je archivácia čo najväčšieho počtu domácich webových zdrojov na základe parametrov definovaných v aplikovanom SW. Hlavným parametrom pre archiváciu českého webu je národná doména[2].

Výberový prístup Pravidlá výberu dokumentov registrovaných v národnej bibliografii zahrňujú:

- *Územie* - všetky dokumenty (zdroje) publikované na území Českej republiky.
- *Jazyk* - všetky zdroje v čestine (bez ohľadu na miesto vydania)
- *Autorstvo* - všetky zdroje českých autorov (bez ohľadu na miesto vydania)
- *Predmet/obsah* - všetky zdroje, ktorých obsah sa týka Českej republiky alebo českého národa (bez ohľadu na miesto vydania)[2].

Tematické zbierky Tematické zbierky sú monotematické súbory webových dokumentov. V rámci tematických sklizní sledujeme predovšetkým také deje, ktoré doprevádza celospoločenská debata a je u nich teda predpoklad, že zaujmú významejšie miesto z hľadiska dejín Českej republiky (napr. voľby, povodne)[2].

2.2 Plošná sklizeň národnej domény

WebArchiv vykonáva plošnú sklizeň národnej domény pomocou softvérových nástrojov, webových pavúkov (crawler - angl.), ktoré sú na túto činnosť stavané. Webové pavúky dokážu automaticky prechádzať celý Web, stránku po stránke. Medzi najpopulárnejšie patrí systém Heritrix, Httrack a Nedlib Harvester.

Webové pavúky začínajú parsovaním štartovacej webovej stránky, ktoré sú v tejto problematike známe pod názvom semienka. Semienka sú zdrojové URI (unified resource identifier), z ktorých celý proces začína. Pri parsovaní webový pavúk hľadá hypertextové odkazy, ktoré sú na tejto stránke umiestené a odkazujú na ďalšie stránky. Pavúk následne parsuje nájdené stránky, aby našiel nové odkazy. Celý proces sa rekurzívne opakuje, pokiaľ už nebudú žiadne odkazy, ktoré by sa spracovali. Webový pavúk posiela HTTP požiadavky na dokumenty umiestnené na serveroch. Postup je rovnaký ako keď užívateľ kliká na odkazy vo svojom webovom prehliadači. Webový pavúk však dokáže tento proces automatizovať.

2.2.1 Úvod do Heritrixu

WebArchiv používa pri plošnej sklizni softvér Heritrix, ktorého hlavnou výhodou je modulárnosť a rozšíriteľnosť. Heritrix sa dá jednoducho nastaviť tak, aby používal modul, ktorý pri spracovaní webových stránok akceptuje len stránky s českou doménou. Všetky ostatné URI iných domén sú ignorované.

Heritrix je pred spustením sklizne nakonfigurovaný. Na vstupe je množina semienok, ktoré predstavujú adresy rozsiahlych českých portálov. Ideou je zvoliť za semienka také stránky, ktoré obsahujú čo najviac odkazov na iné české zdroje a stránky. Ideálny kandidát pre semienko je napríklad portál *seznam.cz*, ktorý je výborným rozcestníkom do českého webu.

Proces nazvaný *sklizeň_cz_2007*, ktorého hlavnou úlohou bola celoplošná sklizeň domácich webových zdrojov, trval 3 týždne na serverovom počítači. Výsledkom sklizne vykonanej na systéme Heritrix je kolekcia archívnych súborov, ktoré sú umiestnené v adresárovej štruktúre na disku. Celá kolekcia za rok 2007 obsahuje 81 300 000 dokumentov o celkovej veľkosti 3.6 TB.

2.2.1.1 Archívny súbor - ARC

ARC je súbor vytvorený pre archívne účely a je hlavným stavebným kameňom arhívu. Obvyklá veľkosť ARCu je 100 MB. Skladá sa z hlavičky, ktorá jednoznačne identifikuje každý ARC a jednotolivých URL záznamov, ktoré obsahujú samotné dáta. Nástupcom súboru ARC je WARC, ktorý sa bude tradične používať na ukládanie dát získaných zo sklizní Internetového obsahu. WARC je navrhnutý ako norma ISO 28500.

2.3 Zprístupnenie archívu

ARC súbory, ktoré sme získali z Heritrixu sa musia zaindexovať, tak aby bolo možné v nich jednotlivé záznamy rýchlo vyhľadávať . Problém indexovania a vyhľadávania záznamov v ARC súboroch rieši systém ARCWayback, ktorý si WebArchív vyvinul presne na svoje potreby. Sprístupnenie týchto záznamov poskytuje ďalší systém Wayback Machine. Systém ARCWayback, ktorý používa WebArchiv je popísaný v práci *Accessing the Czech Web Archive*[10].

2.3.1 Wayback Machine

Systém vyvinutý spoločnosťou Internet Archive, ktorý užívateľom umožňuje prehliadať archivované verzie webových stránok z rôznych časov. Wayback Machine poskytuje prístup k archivovaným dátam prostredníctvom URL (unified resource locator). Jeho nevýhoda je, že nepodporuje fulltextové prehľadávanie, takže užívateľ musím poznať URL adresu webovej stránky aby mohol pristúpiť k archívu. Po zadaní URL adresy Wayback Machine vráti zoznam dátumov, kedy bola stránka archivovaná. Potom stačí kliknúť na jeden zo záznamov a prezerať archivovanú stránku. Odkazy, ktoré sa na tejto stránke nachádzajú odkazujú na iné stránky z archívu. Pri klikaní na odkazy sa Wayback Machine vždy snaží vrátiť stránky, ktoré sú archivované v čo najbližšom čase ako zvolená stránka. To znamená že môžete surfovať po webe tak, ako vyzeral v danom čase v minulosti.

2.4 Nástup sklizne mimo národnú doménu

Sklizeň mimo doménu .cz by mala naviazať na celoplošnú sklizeň českých webových zdrojov. Pri sklizni národnej domény sa archivujú všetky české stránky, ktoré prejdú procesom Heritrixu. Všetky odkazy, ktoré sa na spracovávaných stránkach nájdu sa filtrujú na základe domény. Odkazy s českou doménou sú ďalej zahrnuté do kolekcie URL adries čakajúcich na spracovanie, zatiaľ čo ostané odkazy vedúce do iných domén sú odhodené.

Tieto odhodené odkazy mimo doménu .cz však môžu byť hodnotným materiálom pri nástupe sklizne mimo .cz. Existuje možnosť, že odkazy mimo doménu .cz, ktoré sa nachádzajú na českej stránke môžu odkazovať na iné relevanté, české stránky, ktorých doména už nie je česká. Preto je výhodné skúmať tieto odhodené odkazy zo stránok s českou doménou ako prvé. Heritrix sa dá nakonfigurovať tak, aby tieto odhodené odkazy z českých stránok logoval do osobitného súboru. Tento súbor bude po skončení sklizne domény .cz plný hodnotných odkazov, ktoré môžeme predať Heritrixu na vstupe vo forme semienok pri následnej sklizni mimo doménu .cz.

Po skončení *sklizne_cz_2007* Heritrix vygeneroval súbor *outScope.log*, ktorého veľkosť je 16 GB. Na každom riadku sa nachádzajú logovacie informácie a URL, ktorá sa použije ako semienko pre sklizeň mimo doménu .cz. Z technických príčin nie je možné predhodiť Heritrixu 16 GB dlhý zoznam semienok a preto je nutné tento zoznam rozdeliť na viac kolekcií, ktoré sa postupne spracujú.

Kapitola 3

Heritrix

Heritrix je webový pavúk a zároveň archivačný nástroj v jednom. Je to open-source nástroj napísaný v Jave, vyvíjaný spoločnosťou Internet Archive. Poskytuje užívateľsky príjemne webové rozhranie a okrem toho umožňuje spustiť a inicializovať sklizeň cez príkazový riadok, čím ušetrí čas pri spúšťaní opakovaných úloch. Užívateľský manuál sa nachádza na stránkach Heritrixu *http://crawler.archive.org*

Pred spustením sklizne je treba v Heritrixe vytvoriť takzvaný *crawl job*, v ktorom sa ďalej vyberajú a konfigurujú moduly, ktoré chcete pri sklizni použiť. Prostredníctvom webového rozhrania je konfigurácia intuitívna a jednoduchá. Pri vytváraní *crawl job* je nutné zadať jeho názov, stručný popis a hlavne zoznam semienok, ktoré sa majú použiť ako iniciálne URI pri štarte sklizne.

Defaultné moduly sú ihneď nastavené, ale ak chcete robiť konkrétnu sklizeň, tak si môžete zvoliť tie moduly, ktoré aktuálne potrebujete. Pri každom module je jeho stručný popis, aby bolo jasné na akú úlohu daný modul slúži.

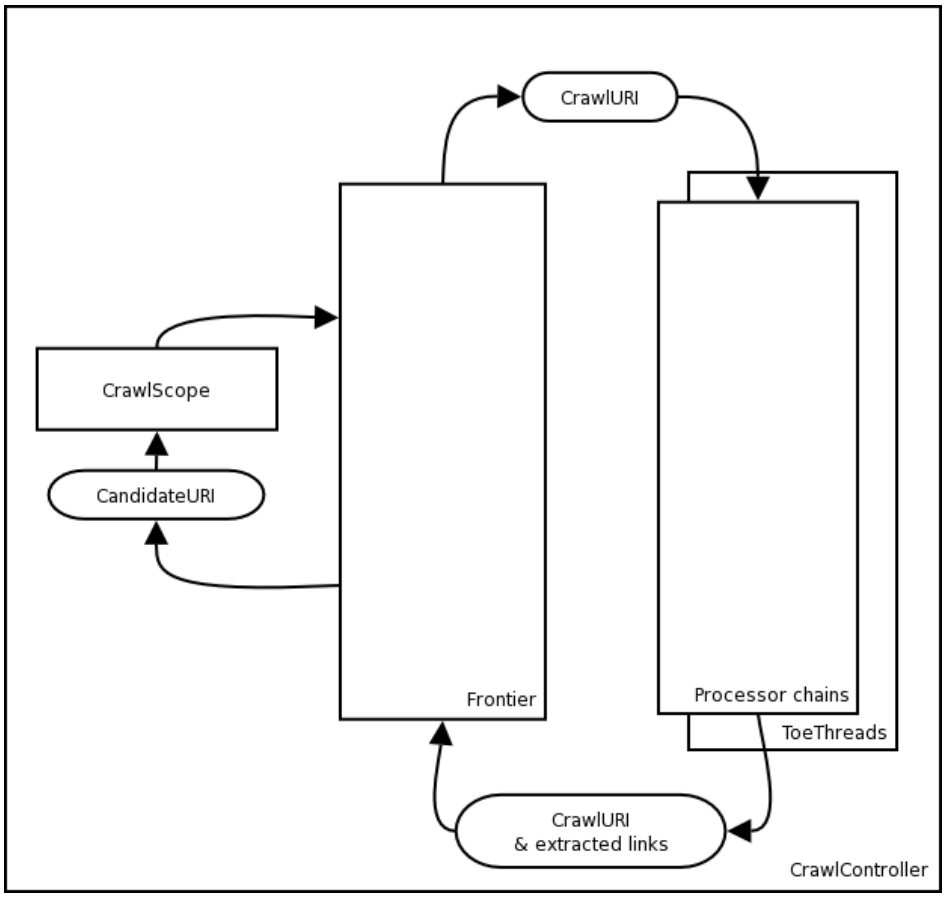
Po konfigurácií modulov ostáva posledný krok. Vytvorený *crawl job* stačí už len spustiť. Webové rozhranie ukazuje aktuálne informácie o bežiacom *crawl job*, medzi ktoré patrí počet aktuálne pracujúcich vláken, spotrebovaná pamäť, počet a veľkosť archivovaných dokumentov.

3.1 Prehľad Heritrixu

Heritrix je navrhnutý modulárne. Ktoré moduly sa majú použiť môže byť nastavené za behu z užívateľského rozhrania. Ak chcete aby sa webový pavúk správal inak ako je defaultne nastavený, môžete naprogramovať vlastný modul, ktorý pridáte do Heritrixu k ostatným alebo ním nahradíte nietkorý zo stávajúcich modulov.

Pred programovaním vlastného modulu je nevyhnutné vedieť akým spôsobom Heritrix pracuje a keďže program je vytvorený v Jave, je nutné vedieť sa orientovať v tomto programovacom jazyku.

Heritrix pozostáva z jadra a prídavných modulov. Triedy jadra sa dajú konfigurovať, ale nemôžu byť odstránené, zatiaľ čo triedy prídavných modulov môžu byť nahradené. [3].



Obrázok 3.1: Prehľad Heritrixu

3.2 Objekt CrawlController

CrawlController obsahuje všetky triedy, ktoré spolupracujú pri vykonávaní plazenia po stránkach. Poskytuje rozhranie pre činnosť plazenia a spúšťa hlavné vlákno *master thread*, ktoré prideľuje URI adresy z objektu Frontier jednotlivým vláknam ToeThread. CrawlController predstavuje globálny kontext celého procesu plazenia a všetky subkomponenty k sebe zvyčajne pristupujú a komunikujú cez CrawlController. [3].

3.3 Objekt Frontier

Objetk Frontier je zodpovedný za prideľovanie následovnej URI, ktorá má byť spracovaná. Má zapezpečovať zdvorilosť, ktorá zaisťuje, že žiadny webový server nebude príliš zaťažovaný sťahovaním stránok, ktoré sú na ňom umiestnené. Po tom ako je URI spracovaná je predaná opäť objektu Frontier spolu so všetkými jej objavenými odkazmi, ktoré by mal Frontier naplánovať na ďalšie spracovanie[3].

Frontier je objekt, ktorý udržuje stav procesu prechádzania stránok. Aktuálny stav zahŕňa hlavne tieto položky:

- *URI adresy, ktoré boli objavené*
- *URI adresy, ktoré sa aktuálne spracúvajú*
- *URI adresy, ktoré boli spracované*

Trieda Frontier implementuje rozhranie Frontier a môže byť nahradená hocijakou inou triedou Frontier, ktorá bude toto rozhranie implementovať. Je nutné spomenúť, že naprogramovanie triedy Frontier nie je triviálna záležitosť[3].

Objekt Frontier sa spolieha na správanie nasledújucich externých procesorov: PreconditionEnforcer, LinksScoper a FrontierScheduler.

Procesor PreconditionEnforcer zabezpečuje, že *DNS* a *robots.txt* sú skontrolované pred tým, než sa daná stránka stiahne zo serveru. LinksScoper testuje či je daná URI v oblasti záujmu a ak áno, tak aký výsoký je záujem o danú URI. Podľa toho LinksScoper nastaví prioritu pre danú URI, na základe ktorej sa vykonáva zoradenie URI adries čakajúcich na spracovanie v objekte Frontier. Toto zoradenie sa už vykonáva pomocou objektu FrontierScheduler[3].

3.4 Objekt ToeThread

Heritrix je multivláknová aplikácia. Každá URI je spracovávaná svojím vlastným vláknom nazvaným ToeThread. Objekt ToeThread žiada objekt Frontier o novú URI a následne ju posiela cez všetky procesory. Po spracovaní URI posledným procesorom si ToeThread pýta ďalšiu URI čakajúcu na spracovanie[3].

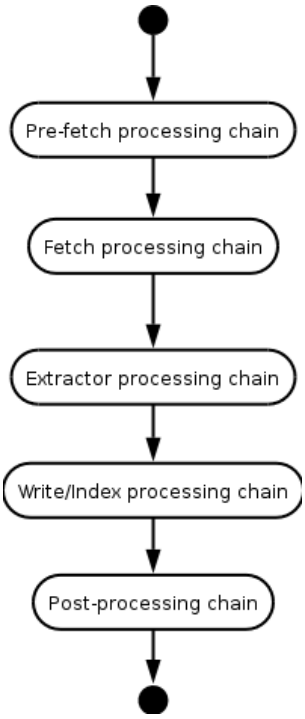
3.5 Objekt Processor

Procesory sú zoskupené do procesorových reťazcov. Každá reťaz vykonáva s URI nejaký proces. Keď objekt Processor spracuje URI, ToeThread pošle URI do nasledujúceho procesoru, pokiaľ nie je spracovaná všetkými procesormi. Procesor má možnosť prikázať URI aby preskočila do konkrétnej reťaze procesorov. Rovnako ak Processor vyhodí fatálnu chybu, tak URI automaticky skočí do poslednej reťaze nazvanej *Post-processing chain*. Postup spracovania URI jednotlivými procesorovými reťazami:

Postup spracovania URI jednotlivými procesorovými reťazcami:

3.5.1 Pre-fetch processing chain

Prvá reťaz procesorov je zodpovedná za vyšetrovanie, či URI môže byť spracovaná. Je tu zahrnutá kontrola, či sú všetky predpoklady splnené (DNS-vyhľadávanie, kontrola súboru robots.txt, autentifikácia). Je tiež možné úplne zablokovať spracovanie URI adries, ktoré neprešli kontrolou[3].



Obrázok 3.2: Reťazec procesorov

- Vlastníci webových stránok používajú textový súbor nazvaný robots.txt, aby prostredníctvom neho predali inštrukcie o ich stránkach webovým robotom (The Robots Exclusion Protocol). Vždy keď chce webový robot navštíviť URL webovej stránky, najprv hľadá tento súbor robots.txt. Pri návšteve URL http://www.seznam.cz sa robot pokúša získať súbor z URL http://www.seznam.cz/robots.txt. V tomto súbore môže byť zapísané, že by robot nemal navštevovať žiadne stránky z danej domény. Existuje však mnoho robotov, ktoré tento súbor a jeho inštrukcie ignorujú [4].

V reťazi Pre-fetch processing chain by mali byť zahrnuté tieto procesory alebo náhradné moduly, ktoré vykonávajú podobné operácie:

- *procesor Preselector* - Posledná kontrola, či by URI mala byť skutočne spracovaná. Môže napríklad znovu skontrolovať, či URI patrí do oblasti záujmu. Je to užitočné ak sa zmenia pravidlá oblasti záujmu v priebehu procesu samotného plazenia po webových stránkach. Kontrola či URI patrí do oblasti záujmu je zvyčajne vykonávaná objektom LinksScoper, predtým než sú nové URI pridané do objektu Frotnier [3].
- *procesor PreconditionEnforcer* - Zabezpečuje, že všetky predpoklady pre spracovanie URI sú splnené. Konkrétne zahrňuje overenie, že *DNS* a *robots.txt* boli získané pre danú URI [3].

3.5.2 Fetch processing chain

Procesory v tejto reťazi sú zodpovedné za získanie dát zo vzdialeného serveru. Mal by tu byť jeden procesor pre každý protokol, ktorý Heritrix podporuje, ako napríklad FetchHTTP [3].

3.5.3 Extractor processing chain

V tomto bode je obsah dokumentu odkazovaného prostredníctvom URI prístupný a niekoľko procesorov sa bude snažiť získať z tohoto obsahu nové odkazy[3].

3.5.4 Write/Index processing chain

Táto reťaz je zodpovedná za zapisovanie dát do archívnych súborov. Heritrix obsahuje procesor ARCWriterProcessor, ktorý zapisuje dáta do formátu ARC. Nové procesory by mohli byť naprogramované tak, aby podpodorvali aj iné formáty[3].

3.5.5 Post-processing chain

URI adresa vždy prechádza cez túto reťaz procesorov aj keď bolo predtým rozhodnuté iným procesorom, že táto URI sa nebude spracovávať. Reťaz Post-processing chain musí obsahovať nasledujúce procesory alebo obdobné moduly vykonávajúce podobné operácie[3].

- *procesor CrawlStateUpdater* - Aktualizuje informácie o každom hoste, ktoré sa môžu uplatniť pri ďalšom spojení na server. Konkrétne *robots.txt* a informácie o IP adresách[3].
- *procesor LinksScoper* - Kontroluje všetky odkazy extraktované z aktuálneho získaného obsahu. Všetky odkazy, ktoré nepatria do oblasti záujmu (oblasť záujmu môže byť napr. doména .cz), ktorú sme si zvolili sú vyradené[3].
- *procesor FrontierScheduler* - Rozplánuje všetky odkazy nájdene v aktuálne spracovávanej URI do objektu Frontier tak aby sa postupne spracovali podľa priority[3].

3.6 Triedy URI v Heritrixe

URI adresy sú v Heritrixe reprezentované objektom UURI, ktorý rozširuje objekt URI. Ak je UURI adresa úspešne vytvorená, tak to znamená že pôvodná URI spĺňa všetky podmienky a nejedná sa o žiadnu podvrhnutú URI, ktorá by mohla neskôr v procese spracovania vytvárať konflikty. Objekt UURI taktiež poskytuje mnohé metody na prístup k rôznym častiam URI[5].

Dve triedy, ktoré obaľujú triedu UURI v Heritrixe sú:

- *CandidateURI* - Je to URI, ktorá je objavená počas behu procesu a môže byť zahrnutá do kolekcie URI čakajúcich na spracovanie. CandidateURI sa vytvorí z objektu Link, ktorý predstavuje odkaz objavený na spracovávanej CrawlURI. Vytváranie objektuCandidateURI z objektu Link má na starosti procesor LinksScoper. CandidateURI môže byť zahrnutá do kolekcie čakajúcich URI, ak spĺňa definované požiadavky. Pri zavolaní objektom ToeThread sa CandidateURI zmení na objekt CrawlURI[5].

- *CrawlURI* - CrawlURI predstavuje URI, ktorá sa práve spracováva procesormi. Uchováva stav, ktorý sa mení počas spracovania. CrawlURI je potomok triedy CandidateURI. Samotná instancia je vytvorená v momente, keď sa vyberie z objektu Frontier a predá sa procesorom na spracovanie[5].

3.6.1 Zoznam atribútov objektu CrawlURI

Objekt CrawlURI poskytuje zoznam atribútov, ktorý je použitý na uchovanie ľubovoľných informácií o URI počas procesu jej spracovnia. Pri písaní nových procesorov, určite využijete tento zoznam atribútov. Je to vlastne zoznam párov kľúč a hodnota[5].

3.6.2 The recorder streams

Heritrix by nebol moc užitočný, keby neposkytoval prístup k HTTP požiadavku a odpovedi. Pre tento účel má objekt CrawlURI metodu nazvanú getHTTPRecorder(). Rekordér je referencovaný v objekte CrawlURI a tým prístupný pre všetky procesory, ktoré s CrawlURI pracujú[5].

3.7 Požiadakvy pre každý konfigurovateľný modul

Ak chcete napísať vlastný modul, mali by ste dediť z triedy ModuleType, ktorá je potomkom triedy ComplexType. Trieda ModuleType bola zamýšľaná na to, aby bola rodičom každého modulu v Heririxe[6].

3.7.1 Definícia modulu

Heritrix vie ako pracovať s komplexným typom a ako získať potrebné informácie zadané prostredníctvom užívateľského rozhrania. Aby to Heritrix vedel spracovať, nový modul musí splňovať pár pravidel[6]:

- Modul by mal vždy implementovať konštruktor, očakávajúci práve jeden argument *name*.

- Všetky konfigurovateľné atribúty by mali byť definované v konštruktore daného modulu.

3.7.1.1 Povinný jedno argumentový konštruktor

Všetky moduly musia mať konštruktor prijímajúci práve jeden argument typu reťazec. Reťazec je použitý na identifikovanie modulu. V prípade, že nový modul je rovnakého typu ako existujúci modul, ktorý sa môže vyskytovať v Heritrixe práve jeden krát, musí sa meno nového modulu zhodovať s menom existujúceho modulu. To je presne prípad modulu Frontier, ktorý je v procese plazenia po webových stránkach použitý práve jeden krát[6].

3.7.1.2 Definovanie atribútov

Atribúty v module, ktoré majú byť konfigurovateľné musia byť definované v konštruktore modulu. Pre tento účel existuje metóda *addElementToDefinition(Type type)*. [6]:

3.7.2 Zostavenie jednoduchého modulu

Zo získaných znalostí už môžme zostaviť jednoduchý modul, ktorý síce nerobí nič užitočné, ale popisuje základné postupy. Postup naprogramovania modulu môžete nájsť v príručke Heritrixu pre vývojarov[6].

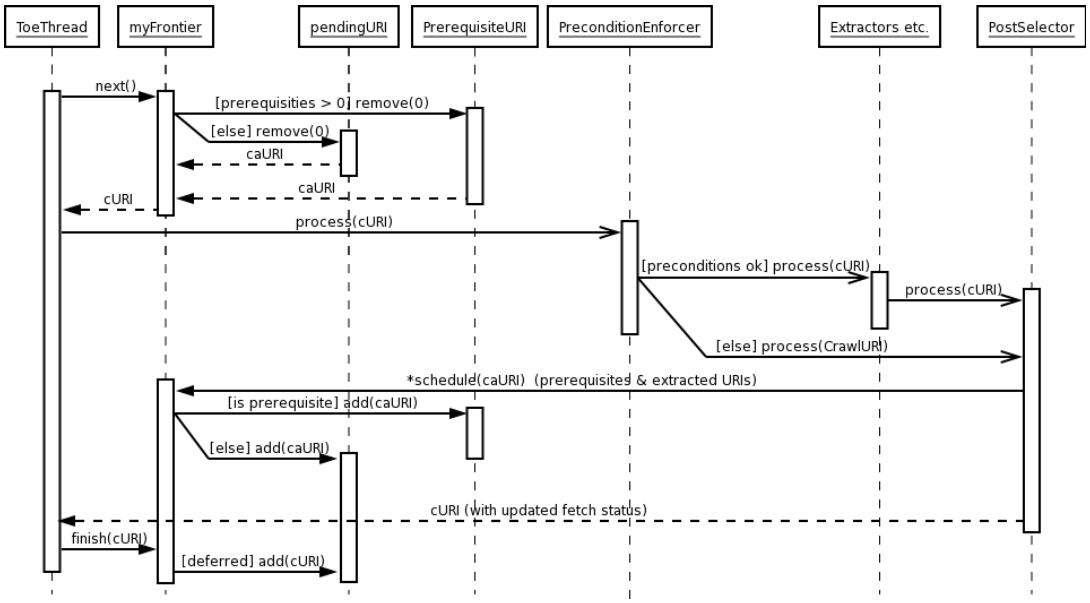
3.7.2.1 Poznámky k pridávaniu modulu do Heritrixu

Aby bol nový modul rozpoznaný Heritrixom, je nutné sa o ňom zmieniť vo vhodnom konfiguračnom súbore (cesta *src/conf/modules*). Ak je nový pridávaný modul Processor, musíte sa o ňom zmieniť v súbore *Processors.opions*. Tento súbor sa pri zostavovaní aplikácke pridá do knižnice Heritrixu *Heritrix.jar* [6].

Ak pri pridávaní nového modulu používate nejakú knižnicu, je nevyhnutné aby bola zmienená v konfiguračných súboroch Heritrixu, ktoré používa zostavovací nástroj *Maven 1*. Rovnako ak celý nový modul zabalíte do knižnice *jar*, stačí ju pridať do adresára *HERITRIX_HOME/lib* a zapísať nový modul do konkrétneho konfiguračného súboru v adresári */src/conf/modules*.

3.8 Napísanie modulu Frontier

Ako bolo spomenuté, Frontier je prídavný modul, ktorý je zodpovedný za rozhodnutie, ktorá URI bude spracovaná ako nasledujúca a kedy. Frontier je taktiež zodpovedný za urdžovanie celkového vnútorného stavu procesu plazenia po webových stránkach a je vhodným miestom pre vytváranie reportov a logovanie aktuálneho stavu. Frontier je jeden z najzložitejších modulov. Preto je lepšie pred jeho modifikovaním premyslieť všetky ostatné alternatívy, ktoré by sa vyhli úpravam v module Frontier[7].



Najdôležitejšie metódy modulu Frontier sú:

- *next(int timeout)* - Metóda next je volaná objektom ToeThread, predstavujúci vlákno, ktoré má vždy pridelenú práve jednu URI s ktorou pracuje. Pri volaní sa predávajú URI podľa pridelenej priority. To umožňuje, že sa najprv spracujú *DNS* záznamy a súbory *robots.txt* predtým než sa z konkrétneho hostu získajú nejaké dáta. Následne sa vybraná URI reprezentovaná objektom CandidateURI premení na objekt CrawlURI, ktorý je vhodný na spracovanie procesormi[7].

- *schedule(CandidateURI caURI)* - Keď je URI spracovaná reťazou procesorov, tak končí v objekte LinksScoper. Všetky URI by tu mali skončiť, aj keď nespĺňali predpoklady na to aby boli spracované. LinksScoper iteruje cez všetky nové URI nájdené v kontexte danej CrawlURI a ak vyhovujú požiadavkom skúmanej oblasti, tak ich prekonvertuje z objektov Link na objekty CandidateURI. Neskôr ich v reťazi *Postprocessor chain* objekt FrontierScheduler pridá do objetku Frontier volaním metody *schedule(CandidateURI)*. Táto metóda zabezpečuje, aby bola každá URI pridaná maximálne raz [7].

- *finished(CrawlURI cURI)* - Až všetky procesory skončia, tak vlákno ToeThread volá metodu *finished(CrawlURI)* objektu Frontier, s parametrom CrawlURI, ktorá práve prešla celým procesom. Ak sa URI úspešne spracovala, tak sa aktulizujú štatistiky a volá sa metóda *stripToMinimal()*, ktorá odstráni všetky nepotrebné dáta zozbierané pri spracovávaní danej CrawlURI[7].

3.9 Napísanie modulu Filter

Filtre sú objekty, ktoré vezmú CrawlURI a určia, či spĺňa definované kritéria. Ak áno vrátia logickú hodnotu *true* inak *false*[8].

Filter môže byť použitý na viacerých miestach v procese plazenia webových stránok. Najväčšie využitie filterov je v procesoroch na odfiltrovanie nehodiacich sa URI. To znamená, že hocijaká URI, ktorá neprejde filtrom v danom procesore tento procesor preskočí. To môže byť užitočné pri zabránení extrahovaní odkazov z dokumentov, ktoré pochádzajú zo špecifickej oblasti webových stránok (napr. vulgárne, pornografické stránky)[8].

3.10 Napísanie modulu Scope

Instancia objektu CrawlScope definuje, ktoré URI sú v rámci nášho záujmu. Je to v podstate filter, ktorý sa pozerá na maximálne množstvo informácií o danej URI, aby určil, či má byť naplánovaná na spracovanie alebo nie. Informácia, ktorá určuje z ktorej URI bola aktuálna URI objavená, by mala byť braná do úvahy, zatiaľ čo dočasné informácie ako *robots.txt* alebo história pokusov spracovať rovnakú URI by sa nemali uvažovať. Tieto časovo náročné rozhodnutia by sa mali uskutočňovať v inom mieste výpočtu[9].

3.11 Napísanie modulu Processor

Všetky procesory rozširujú triedu *org.archive.framework.Processor*. Je to vlastne kompletná trieda a môže byť použitá ako validný procesor, ale v skutočnosti nič nerobí.

Rozširujúce triedy musia prepísať metódu *innerProcess(CrawlURI)*, aby pridali potrebnú funkcionalitu. Táto metóda je zavolaná nad každou URI ktorá sa spracováva.

Metóda *innerProcess* používa objekt CrawlURI, ktorý je predaný ako parameter a objekt HttpRecorder, ktorý je zabalený v objekte CrawlURI.

Objekty Fetcher (FetchHTTP, FetchFTP, ...) získajú relevantný dokument a zapíšu jeho obsah do objektu HttpRecorder. Objekty Extractor (HTMLExtractor, CSSExtractor, ...) čítajú objekt HttpRecorder a pridávajú objevené odkazy do zoznamu odkazov nájdených na danej CrawlURI [10].

Niekoľko iných metód môže byť prepísaných na špeciálne účely:

- *initialTasks()* - Táto metóda bude volaná až bude nastavený celý proces plazenia, ale predtým než bude nejaká URI spracovaná. Je to vlasnte inicializačný kód, ktorý musí byť spustený práve jeden krát pri štarte plazenia webu[10].
- *finalTasks()* - Táto metóda bude zavolaná po tom ako bude spracovaná posledná URI. Je to na samom konci plazenia webu a znamená to, že proces je normálne ukončený. V podstate je to miesto, kde sa môže napísať finalizačný kód[10].
- *report()* - Táto metóda vracia reťazec, ktorý obsahuje ľudsky čitateľnú správu o stave alebo progrese procesu plazenia webu. Táto správa je prístupná cez webové uži-

vateľské rozhranie. Procesor by mohol informovať o počte CrawlURI, ktoré spracoval alebo počte odkazov, ktoré objavil [10].

3.11.1 Pristupovanie a aktualizovanie CrawlURI

Objekt CrawlURI obsahuje nasledujúce metódy:[10]

- *getAlist()* - Táto metóda vracia objekt *AList* danej CrawlURI. *AList* je v podstate mapa do ktorej si môžeme uložiť hocijaké informácie, ktoré potrebujeme použiť pri spracovávaní.
- *setHttpRecorder(HttpRecorder)* - Nastavuje HttpRecorder, ktorý obsahuje stiahnutý obsah zo serveru. Metóda je zvyčajne vykonaná nejakým procesorom z reťaze *Fetch processing chain*.
- *getHttpRecorder()* - Metóda, ktorá získa stiahnutý obsah.
- *getContentSize()* - Ak bol dokument úspešne stiahnutý, táto metóda vráti jeho dĺžku v bytoch.
- *getContentType()* - Mime type stiahnutého obsahu.

3.11.2 HttpRecorder

HttpRecorder je pripojený ku každej CrawlURI, ktorá bola úspešne stiahnutá pomocou FetchHTTP procesoru. Procesory, ktoré sa zaujímajú o textový obsah stiahnutej CrawlURI k nemu môžu pristupovať pomocou metódy *getReplayCharSequence()*, vracajúcej objekt *java.lang.CharSequence*.

3.11.3 Poznámky k písaniu modulu Processor

Pre každý procesor je vytvorená iba jedna instancia počas celého procesu definovanej úlohy *crawl job*. Keďže je Heritrix multivláknová aplikácia, musí byť procesor napísaný správne aby nedochádzalo ku konfliktom[10].

Kapitola 4

Návrh a Integrácia

4.1 Integrácia

Podľa popisu funkčnosti Heritrixu je nutné vhodne integrovať systém WebAnalyzer tak, aby sme dosiahli požadovaných výsledkov. Modul WebAnalyzer musí každopádne spracovávať každú URI, ktorá prejde procesom plazenia webu. Preto je potrebné integrovať WebAnalyzer do niektorého z modulov. Vhodná možnosť je navrhnúť a naimplementovať modul WebAnalyzer, vo forme knižnice *.jar*, ktorá bude poskytovať definované rozhranie, s ktorým bude niektorý z modulov Heritrixu spolupracovať a používať jeho služby. Systém WebAnalyzer by mal vziať na vstupe URI, o ktorej na základe svojich funkcií a definovaných požiadavkov určí, či je česká alebo nie. Teraz ostávalo zistiť, ktorý modul bude ten najvhodnejší pre integráciu s WebAnalyzerom.

WebAnalyzer by mal pri spracovávaní URI dostať na vstup konkrétnu URI a čo najviac informácií, ktoré Heritrix pri jej spracovávaní získal.

Integrácia s modulom Frontier neprichádza do úvahy, pretože Frontier má prístup k URI, buď na samom začiatku spracovania alebo na úplnom konci. Ani jedna z ponúkaných možností nie je vyhovujúca. Pred samotným spracovaním URI Heritrix pozná minimum informácií, medzi ktoré patrí názov samotnej URI, názov URI na ktorej bola aktuálna URI objavená a nejaké informácie o servery, na ktorom sa URI nachádza. V tomto stave nemáme prístupný dokonca ani textový obsah, ktorý získajú až procesory FetchHTTP z reťaze *Fetch processing chain*. Naopak na úplnom konci spracovania Heritrix zistil o URI maximálne množstvo informácií, ktoré by mohli byť pre WebAnalyzer hodnotné. Keby sme nasadili WebAnalyzer v tomto bode a identifikovali URI, tak by sme prišli o možnosť archivovania českých URI. Archivácia sa totiž vykonáva procesorom ARCWriterProcessor, ktorý vstupuje do činnosti skôr ako skončí URI v objekte Frontier.

Z predchádzajúcich úvah vyplýva, že ak chceme využiť funkcionalitu archivovania, ktorú Heritrix ponúka, musíme WebAnalyzer umiestniť niekde pred procesorom ARCWriterProcessor, ktorý archivuje spracovávané URI.

Následne sa ponúka možnosť integrovať WebAnalyzer do modulu Filter, ktorý pri spracovávaní každej URI určuje, či spĺňa definované kritéria alebo nie. Filter môže byť použitý na viacerých miestach procesu spracovania URI, takže by nebol problém umiestniť ho tesne pred modul archivovania a pomocou WebAnalyzeru určit, či je aktuálna URI česká alebo nie. Podľa tohoto rozhodnutia by ju potom mohol následujúci procesor archivovať. Pokus však ukázal, že ani filter nie je vhodným modulom. WebAnalyzer pri analyzovaní použí-

va viacero projektov a knižníc, s ktorými spolupracuje. Funkcionalita týchto podsystémov systému WebAnalyzer vyžaduje, aby bol WebAnalyzer pred samotným spustením nainicializovaný. Tým pádom je nutné vždy pred jeho používaním zavolať jeho metodu *initializeWebAnalyzer()*, ktorá zabezpečí, že sa otvoria a nainicializujú všetky spojenia k externým súborom, databázam a indexom, ktoré WebAnalyzer používa. Taktiež je nutné pri skončení používania zavolať metodu *closeWebAnalyzer()*, ktorá tieto spojenia opäť uzavrie. To znamená, že v čase keď začna filter pracovať, WebAnalyzer už musí byť nainicializovaný. Bola tu možnosť inicializovať WebAnalyzer v čase vytvárania objektu Frontier, ale nakoniec sa tento postup ukázal zbytočne zložitý.

Ďalšia možnosť bola integrácia s modulom Scope. Modul Scope rozhoduje, ktorá URI je v oblasti záujmu a ktorá nie. Je to v podstate obdoba modulu Filter. Scope sám o sebe poskytoval metódy, *initialize()* a *finalize()*, ktoré sa dali preťažiť a mohol by sa v nich inicializovať a zatvárať objekt WebAnalyzer. Taktiež umiestnenie modulu pred procesorom na archiváciu vyhovovalo. Modul Scope sa zdal vyhovujúcim, ale na základe vyjadrenia Internet Archive, ktorý mal v pláne modul Scope úplne preorganizovať a zrefaktorovať v nadchádzajúcej verzii Heritrixu, sme skúsili ešte inú možnosť.

Poslednou a konečnou možnosťou bola integrácia do modulu Processor, ktorá sa ukázala byť najviac vyhovujúcou možnosťou. Procesor sa vyskytuje v niekoľkých vartiantách ako sme spomenuli v sekcii Processor. Heritrix definuje niekoľko reťazí procesorov. Pre systém WebAnalyzer je najvýhodnejšie umiestniť procesor do reťaze *Extractor processing chain*. Prvé dve reťaze *Pre-fetch processing chain* a *Fetch processing chain* neboli vyhovujúce pretože v tom čase ešte nemáme prístupný textový obsah URI, ktorý môže WebAnalyzer použiž na analýzu textu. Textový obsah URI je dostupný až po skončení reťaze *Fetch-processing chain*. Nasledujúca reťaz *Extractor processing chain* obsahuje procesory nazvané Extractor, ktorých hlavná úloha je extraktovať zo získaného textu URI všetky odkazy. Nachádzajú sa tam extractory typu ExtractorHTML, ExtractorCSS a podobné. Keďže kolekcia objavených odkazov mohla byť ďalšou hodnotnou informáciou pre WebAnalyzer, bolo nutné umiestniť WebAnalyzer do procesu, ktorý by sa nachádzal na samom konci tejto reťaze, keď už budú objavené všetky odkazy.

Nasledujúca reťaz po *Extractor-processing chain* je *Write/Index-processing chain*, ktorej hlavnou úlohou je archivovať URI. Úplne posledná reťaz procesorov je *Post-processing chain*, ktorá spracováva objavené odkazy a podľa definovaných kritérií ich ďalej zaraďuje do procesu alebo vyhadzuje.

Processor Extractor rovnako ponúka metódy *initialize()* a *finalize()*, ktoré sa volali vždy práve jeden krát počas celého procesu plazenia webu. Tieto instancie procesorov sú potom používané jednotlivými vláknami ToeThread. V týchto metódach je možné jednoducho zavolať inicializačnú aj finalizčnú metódu WebAnalyzeru, čím sa nám zaistí, že WebAnalyzer sa správne nainicializuje aj uzavrie spolu s procesorom, v ktorom bude integrovaný. Hlavná metóda procesoru *innerProcess(CrawlURI)* bola ideálnym kandidátom na volanie služieb poskytovaných systémom WebAnalyzer.

4.1.1 ExtractorWebAnalyzer

Integračný procesor sme umiestnili do reťaze *Extractor processing chain*. Extractor sa nazýva ExtractorWebAnalyzer a je to potomok triedy Extractor, ktorá je zas potomkom triedy Processor. ExtractorWebAnalyzer bude jediný modul Heritrixu, ktorý bude priamo pracovať so systémom WebAnalyzer. Systém WebAnalyzér ponúka v podstate 4 metódy, ktoré používa ExtractorWebAnalyzer.

- *initialize()* - Metóda, ktorá inicializuje instanciu WebAnalyzer. Pre celý proces plazenia webu sa použije práve jedna instancia WebAnalyzeru.
- *isContetTypeText(curiName)* - Metóda, ktora zistí Mime Type predanej URI.
- *run(urlName, urlContent, urlOutlinks, contentType)* - Metóda, ktorá spustí analýzu konkrénej URI. WebAnalyzer využíva pri analýze názov URI, textový obsah URI, objavené odkazy a Mime Type.
- *close()* - Finalizačná metóda, ktorá uzavrie všetky spojenia používané systémom WebAnalyzer.

Pri testovaní tejto verzie sme dosiahli, že každá jedna URI bola analyzovaná našim WebAnalyzerom. Na základe rozhodnutia WebAnalyzeru sme v procesore ExtractorWebAnalyzer nastavili, ktorá URI sa má archivovať a ktorá má naopak procesor na archivovanie preskočiť. Tým sme dokázali archivovať všetky URI, ktoré WebAnalyzer identifikoval ako české.

4.1.2 Nedostatky prvej verzie integrácie WebAnalyzeru

Počas testovania prvej verzie sa objavilo mnoho nedostatkov, ktoré si vyžadovali mnoho práce na integrácií.

- *archivácia obrázkov* - Pri archivácií českej stránky sa nearchivovali obrázky a css štýly, čím bola stránka veľmi ochudobnená. Každý obrázok a hociktorý iný zdroj na stránke je samostatnou URI adresou spracovávanou nezávisle na ostatných URI adresách. Preto je nutné aby si systém pamätal, ktoré URI sú obrázky z už archivovaných českých stránok a archivoval ich tiež.
- *archivácia odkazov* - Podobný problém ako s obrázkami vznikol aj pri odkazoch. Pri identifikácií vznikla požiadavka aby systém okrem samotnej českej stránky archivoval automaticky aj všetky odkazy, ktoré sa na tejto stránky vyskytujú a to do určitej úrovne. To znamená, že bolo nutné definovať do akej hĺbky od českej stránky sa majú referencované stránky automaticky archivovať. Tieto stránky by totiž mohli byť hodnotným zdrojom odkazovaným z českej stránky. Opäť bolo nutné aby si systém pamätal, z ktorej rodičovskej URI bola aktuálna URI objavená a či tá rodičovská URI bola identifikovaná ako česká.

- *analyzovanie binárneho obsahu* - Počas testovania záznamy z logu ukazovali, že WebAnalyzer analyzuje aj binárny obsah, čo je nežiadúce, pretože textová analýza binárnych dokumentov nedáva zmysel. Napriek tomu, že v procesore ExtractorWebAnalyzer bolo jasne definované, aby WebAnalyzer analyzoval iba dokumenty s textovým obsahom, sa v logu objavili analýzy dokumentov rôzneho Mime Type (audio, video, image, application). Bolo teda nutné precízne identifikovať typ dokumentu aby sa predišlo nezmyslených analýzam, ktoré boli dôvodom mnohých ďalších chýb.

- *výnimka OutOfMemory* - Počas testovania Heritixu sa mnoho krát objavila výnimka *java.lang.OutOfMemory*. Z logu sa dalo vyčítať že dôvodom bola práve analýza textového obsahu rôznych dokumentov typu audio a video, ktorých veľkosť bola v stovkách MB. Pri analýze sa vytvárali objety *java.lang.String* ktoré zaberali mnoho pamäte a spôsobovali výnimku *java.lang.OutOfMemory*. To bol ďalší dôvod aby bola riadne zabezpečená identifikácia typu dokumentu.

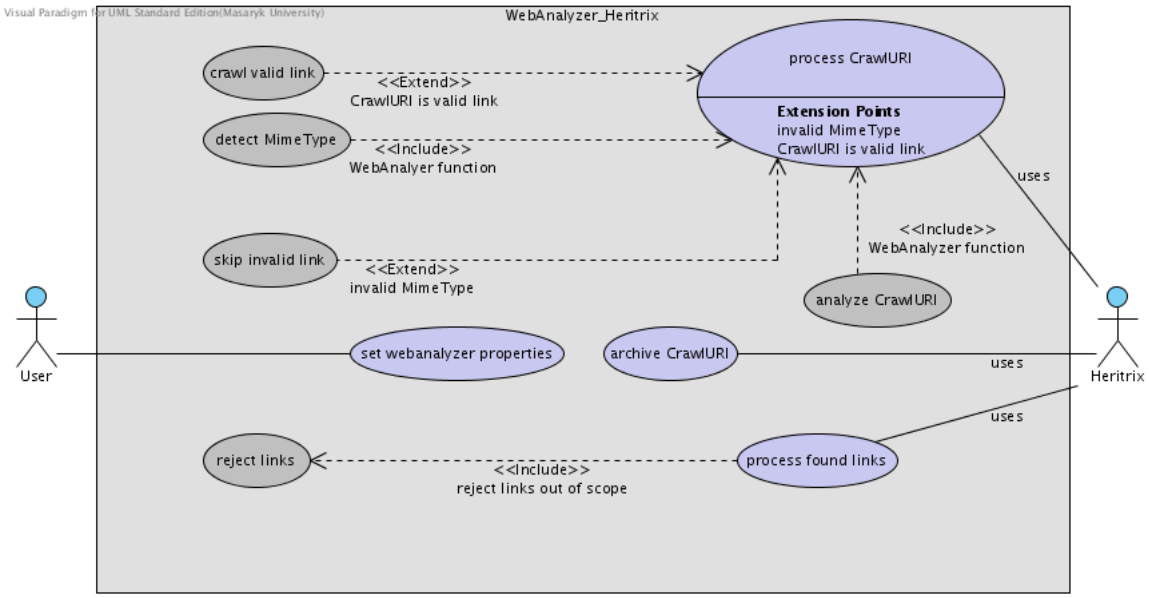
- *archivovanie DNS záznamov* - Predtým než sa nejaká stránka stiahne zo serveru, tak sa spracuje odpvedajúci *DNS* záznam. Tento *DNS* záznam Heritrix automatický archivuje, vďaka čomu sme mali v ARC súboroch obrovské množstvo zbytočných DNS záznamov. My však chceme archivovať len tie *DNS* záznamy, ktoré sa spájajú s českými stránkami. Tohoto stavu je ale ťažké dosiahnuť, pretože v čase keď prebieha analýza stránky je už odpvedajúci *DNS* záznam archivovaný. Riešením by bola dodatočná archivácia *DNS* záznamov. Dočasné riešenie je nearchivovať žiadne *DNS* záznamy, pretože nemajú vpliv na sprístupnenie archivovaných stránok.

4.1.3 Integrácia pomocou viacerých modulov

Na základe nedostatkov z prvej otestovanej verzie, vznikli ďalšie požiadavky, ktoré by mal systém spĺňovať. Na základe nových požiadavkov vznikol use case diagram Obrázok 4.1, ktorý popisuje integráciu modulu WebAnalyzer so systémom Heritrix. Integrácia je realizovaná pomocou modulov ExtractorWebAnalyzer, LinksScoperWebAnalyzer a ARCWriterProcessorWebAnayzer, ktoré sú súčasťou Heritrixu. Procesor ExtractorWebAnalyzer je jediný, ktorý používa rozhranie modulu WebAnalyzer.

4.1.3.1 Archivovanie DNS záznamov

Pred spracovaním ľubovoľnej stránky z konkrétneho hostu sa najprv získa *DNS* záznam. Ten je spracovaný procesorom PreconditionEnforcer, ktorý overuje či sú splnené všetky požiadavky pre nasledujúce spracovanie danej URI (napr. *DNS, robots.txt*). ARCWriterProcessor tieto *DNS* záznamy automaticky archivuje. Pre nás je však zbytočné archivovať všetky DNS záznamy. Tento problém zatiaľ nemá vysokú prioritu a preto použijeme vlastný procesor ARCWriterProcessorWebAnalyzer, ktorý jednoducho nebude archivovať DNS záznamy.



Obrázok 4.1: Use case diagram integrácie systému Heritrix a WebAnalyzer

4.1.3.2 Výnimka OutOfMemory a analyzovanie binárneho obsahu

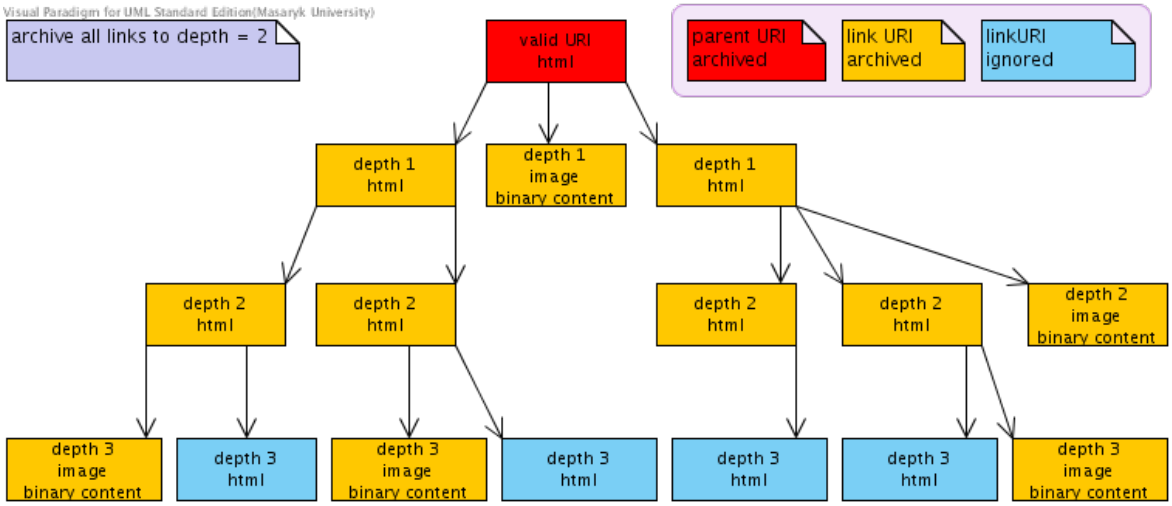
Pri spracovaní dochádzalo pomerne často k výnimke *java.lang.OutOfMemory* spôsobenej v dôsledku analyzovania binárnych dokumentov.

Staré typy serverov, ktoré nevedeli rozoznať typ dokumentu automaticky nastavovali pre tieto dokumenty *contentType = text/plain*. V skutočnosti to však mohol byť dokument s MimeType typu video, audio alebo application. Pri textovej analýze takýchto falošných „textových“ dokumentov, modulom WebAnalyzer, sa vytvárajú enormne veľké objekty *java.lang.String*, ktoré spôsobujú výnimku *java.lang.OutOfMemory*.

ExtractorWebAnalyzer kontroluje MimeType všetkých dokumentov pred tým než začne analýza modulom WebAnalyzer. Funkčnosť Heritrixu, ktorú používa ExtractorWebAnalyzer na určenie MimeType dokumentu evidentne nie je dostatočná a preto bolo nutné vytvoriť podsystém MimeTypeDetector, ktorý identifikuje správny MimeType dokumentu. MimeTypeDetector je nutné umiestniť do procesoru ExtractorWebAnalyzer pred spustením modulu WebAnalyzer, aby sa predišlo analýzam binárnych dokumentov.

4.1.3.3 Archivácia odkazov a obrázkov

Predchádzajúca verzia systém archivuje len tie URI, ktoré WebAnalyzer identifikoval ako české. Všetky ostatné URI (URI obrázkov, css súborov a iných odkazov českej stránky) sa nearchivujú. My však potrebujeme archivovať nielen zdroje českej stránky (obrázky, css) ale aj odkazy vedúce z tejto stránky a to do určitej úrovne. Odkazy vedúce z českej stránky pravdepodobne nesú hodnotné informácie asociované s odpovedajúcou českou stránkou.



Obrázok 4.2: Archivovanie odkazov do hĺbky 2

Obrázok ukazuje strom, ktorého koreň je česká stránka a jej listy sú odkazy z tejto stránky. Pri definovanej hĺbke 2 by sa mali všetky odkazy, obrázky a iné zdroje vedúce z českej stránky automaticky archivovať do tejto hĺbky. Pri archivovaní odkazov 2. úrovne však budú tieto URI opäť bez obrázkov, css a iných zdrojov. Preto musíme archivovať aj 3. úroveň, ale okrem html dokumentov. Tým zabezpečíme, že každá archivovaná stránka bude mať v ARC súboroch aj svoje obrázky, css štýly, audio, video nahrávky a iné zdroje. Definovanie hĺbky môže nastaviť užívateľ pred spustením celého procesu.

Archivácia odkazov z českých stránok do určitej hĺbky sa dala zrealizovať novým modulom LinksScoperWebAnalyzer a odpovedajúcimi zmenami v procesore ExtractorWebAnalyzer tak, aby mohli moduly vzájomne spolupracovať. Veľkou pomôckou boli objekty CrawlURI a CandidateURI, ktoré obsahujú objekt AList, čo je zoznam objektov pridružených k danej URI. Do tohto zoznamu je možné pridať hocijaký príznak (objekt), ku ktorému môžeme pristupovať v ľubovoľnom module. Na základe týchto príznakov si Heritrix môže zapamätať, ktoré URI sú české, ktoré URI sú validnými odkazmi z českých stránok alebo URI, ktoré predstavujú obrázky, css štýly či iné zdroje asociované s českými stránkami. Týmto spôsobom môžeme všetky URI odkazované z českej stránky archivovať a to do určitej úrovne, ktorú si môže užívateľ sám zvoliť.

Modul LinksScoperWebAnalyzer je potomkom modulu LinksScoper, ale dopĺňa funkcionalitu na spracovanie odkazov z českých stránok.

Postup archivovania odkazov z českých stránok: Obrázok 4.3

- *1. Frontier next(int Timeout)* - Každá CrawlURI vstupuje do procesu z objektu Frontier. Zavolaním jeho metody next() si vlákno ToeThread vezme nasledujúcu CrawlURI, ktorá potom putuje procesormi.

- *2. ExtractorWebAnalyzer innerProcess(CrawlURI curi)* - Keď CrawlURI dosiahne procesor ExtractorWebAnalyzer, tak sa zisťuje či má nastavený nejaký príznak, podľa ktorého sa odvíja ďalšie spracovanie.

  - *2.1 CrawlURI NO FLAG* - CrawlURI nemá nastavený žiadny príznak. Bude následne identifikovaný jej MimeType pomocou modulu WebAnalyzer.
  - *2.2 CrawlURI FLAG=valid_link, DEPTH=x* - CrawlURI má nastavený príznak *valid_link*, ktorý definuje, že táto URI bola odkazom z českej stránky a má sa okamžite archivovať. Nie je nutné robiť jej analýzu. Putuje rovno do procesoru ARCWriterProcesorWebAnalyzer, ktorý ju archivuje. Príznak *DEPTH=x*, definuje do akej hĺbky sa majú odkazy nájdené z tejto URI automaticky archivovať. Táto hĺbka sa znižuje s každou archivovanou úrovňou.
  - *2.3 CrawlURI FLAG=archive_bin_link, DEPTH=0* - CrawlURI s príznakom *archive_bin_link* definuje, že táto URI je odkazom z českej stránky, ale v stromovej štruktúre predstavuje list (Obrázok 4.2). To znamená, že sa vykonala archivácia všetkých uzlov vedúcich z českej stránky (koreňa) do definovanej hĺbky. V listoch však môžu existovať odkazy na obrázky, css štýly a iné zdroje, ktorý by chýbali v už archivovaných stránakch. Preto je nutné z týchto listov archivovať všetky dokumenty okrem tých, ktorých *content type* je *text/html*. Týmto dosiahneme archiváciu obrázkov a css štýlov.
    - *2.3.1 CrawlURI Content type=text/css, DEPTH=0* - Ak je CrawlURI s príznakom *archive_bin_link* dokument typu css, tak sa automaticky archivuje. Okrem toho sa pre túto CrawlURI nastaví príznak *valid_link* a *depth=0*. Tým sa zabezpečí že budú archivované aj obrázky nachádzajúce sa v css dokumente, ktoré by inak v ARC súboroch chýbali.
    - *2.3.2 CrawlURI Content type=other* - CrawlURI s nastaveným príznakom *archive_bin_link*, ktorých MimeType je iný ako *text/html* sa taktiež automaticky archivujú. Tým dosiahneme archiváciu obrázkov a iných multimediálnych dokumentov, ktoré sa nachádzajú na už archivovaných stránkach.
    - *2.3.2 CrawlURI Content type=other* - CrawlURI s nastaveným príznakom *archive_bin_link*, ktorých MimeType je *text/html* sa už archivovať nebudú a rovnako sa nebudú nastavovať žiadne príznaky pre odkazy na tejto stránke. To by už znamenalo presiahnutie definovanej hĺbky zadanej užívateľom.

- *3. MimeTypeDetector isContentTypeText(curi.getName())* - CrawlURI, ktorá nemala nastavený žiadny príznak je identifikovaná modulom MimeTypeDetector, ktorý je súčasťou modulu WebAnalyzer. Na základe získaného MimeType sa CrawlURI vydá dvomi rôznymi cestami.

  - *3.1 CrawlURI Content type=other* - CrawlURI, ktorej identifikovaný MimeType bol iný ako *text* sa nebude analyzovať. Pravdepodobne sa jedná o URI s binárnym obsahom, ktorej analýza nemá zmysel. CrawlURI postupuje do reťazca *Post-processing chain*.
  - *3.2 CrawlURI Content type=text* - CrawlURI, ktorej identifikovaný MimeType bol *text* bude spracovaná modulom WebAnalyzer

- *4. WebAnalyzer run(curiName, outlinks, contetType, content)* - CrawlURI, ktoré sa dostali do tejto vetvy sú spracované WebAnalyzerom. Ten na základe parametrov o danej CrawlURI identifikuje či sa jedná o českú URI alebo nie.

  - *4.1 CrawlURI NO FLAG* - CrawlURI, ktorá nebola WebAnalyzerom identifikovaná ako česká postupuje do reťaze *Post-processing chain*. Nenastavuje sa jej žiadny príznak.
  - *4.2 CrawlURI FLAG=valid_URI, DEPTH=x* - CrawlURI, identifikovaná ako česká bude následne archivovaná. Predtým sa jej však nastaví príznak *valid_URI* a *depth=x*. Hodnotu pre hĺbku získa WebAnalyzer z externého súboru, ktorý nastavuje užívateľ pred spustením. Tým zabezpečíme aby si systém pamätal, do akej hĺbky má rekurzívne nájdené odkazy automaticky archivovať.

- *5. ARCWriterProcessorWebAnalyzer innerProcess(CrawlURI curi)* - Procesor ARCWriterProcessorWebAnalyzer archivuje každú CrawlURI, ktorá sa mu predá ako parameter s výnimkou CrawlURI typu *DNS*.

- *6. LinksScoperWebAnalyzer innerProcess(CrawlURI curi)* - Procesor LinksScoperWebAnalyzer spracuje všetky odkazy nájdené na danej CrawlURI predanej ako parameter. Podľa príznaku CrawlURI sa vytvoria ekvivalentné CandidateURI z objavených odkazov a nastavia sa im príslušné príznaky.

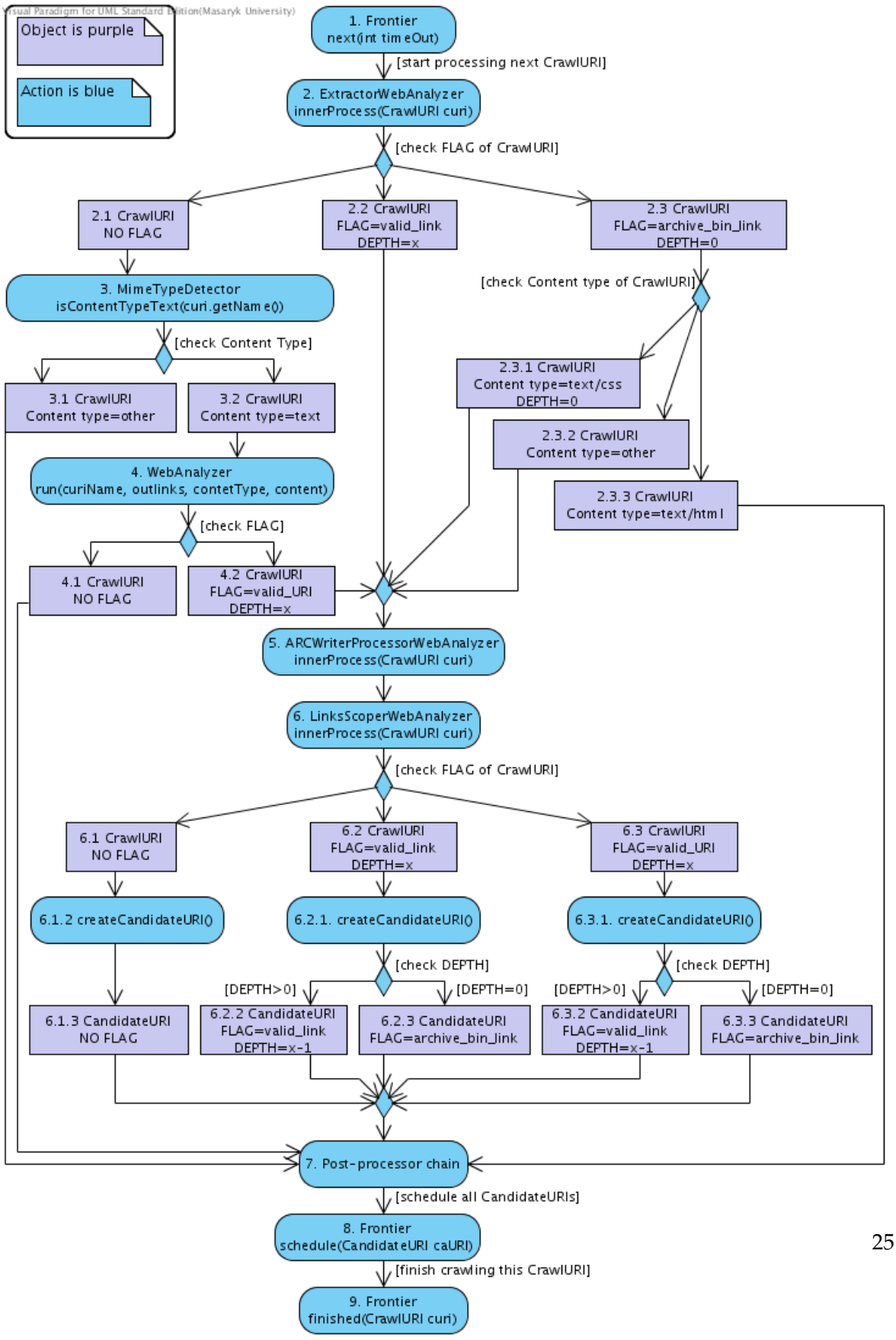
  - *6.1 CrawlURI NO FLAG* - CrawlURI, ktorá nemá nastavený žiadny príznak sa spracuje klasickým spôsobom
    * *6.1.2 createCandidateURI()* - Vytvorí sa objekt CandidateURI.
    * *6.1.3 CandidateURI NO FLAG* - CandidateURI, ktorej rodičovská stránka CrawlURI nemá žiadny príznak pokračuje klasickým spôsobom v spracovaní reťazou *Post-processing chain*.
  - *6.2 CrawlURI FLAG=valid_link, DEPTH=x* - CrawlURI, ktorá ma nastavený príznak *valid_link* sa ďalej spracuje podľa toho aká je hodnota príznaku *depth*.
    * *6.2.1. createCandidateURI()* - Vytvorí sa objekt CandidateURI.
    * *6.2.2 CandidateURI FLAG=valid_link, DEPTH=x-1* - CrawlURI, ktorej príznak *depth=x* má hodnotu x > 0. Objektom CandidateURI vytvoreným

z odkazov sa nastaví príznak *valid_link* a hodnota príznaku *depth* bude *depth=x-1*. Tým sa zabezpečí, že sa archivujú validné odkazy do definovanej hĺbky. Po znížení hĺbky na x=0 sa objektom CandidateURI nastaví príznak *archive_bin_link*, ktorý zaistí, že sa z týchto odkazov archivujú len obrázky, css štýly a iné zdroje umiestené na predtým archivovanej stránke.

    - *6.2.3 CandidateURI FLAG=archive_bin_link* - CrawlURI s nastaveným príznakom*depth=0*. Objektom CandidateURI vytvoreným z takýchto odkazov sa nastaví príznak *archive_bin_link*.

- *6.3 CrawlURI FLAG=valid_URI, DEPTH=x* - CrawlURI, ktorá ma nastavený príznak *valid_URI* sa ďalej spracuje podľa toho aká je hodnota príznaku *depth*.
    - *6.3.1. createCandidateURI()* - Vytvorí sa objekt CandidateURI.
    - *6.3.2 CandidateURI FLAG=valid_link, DEPTH=x-1* - CrawlURI, ktorej príznak *depth=x* má hodnotu x > 0. Objektom CandidateURI vytvoreným z odkazov sa nastaví príznak *valid_link* a hodnota príznaku *depth* bude *depth=x-1*.
    - *6.3.3 CandidateURI FLAG=archive_bin_link* - CrawlURI s nastaveným príznakom *depth=0*. Objektom CandidateURI vytvoreným z takýchto odkazov sa nastaví príznak *archive_bin_link*.

- *7. Post-processor chain* - Každá CrawlURI musí prejsť touto reťazou procesorov.
- *8. Frontier schedule(CandidateURI caURI)* - Zaradí každú CandidateURI do kolekcie URI čakajúcich na spracovanie podľa priority.
- *9. Frontier finished(CrawlURI curi)* - Potom ako všetky procesory skončia, ToeThread volá metodu *finish(CrawlURI curi)* objektu Frontier.



Obrázok 4.3: Diagram popisujúci spracovanie URI pomocou integrácie viacerých modulov

Kapitola 5

WebAnalyzer - návrh a implementácia

V kapitole o integrácií modulu WebAnalyzer so systémom Heritrix, sme spomínali, že systémWebAnalyzer bude klasická knižcnica, ktorá bude poskytovať rozhranie, používané v procesore ExtractorWebAnalyzer. Zbytok funcionality celého systému na identifikovanie a rozpoznanie webu mimo národnej domény je zahrnutý v integračných moduloch. Teraz sa zameriame na návrh tried systému WebAnalyzer.

WebAnalyzer sme navrhli tak, aby mohol byť použitý ako samostatná komponenta, ktorá bude identifikovať URI podľa nastavených požiadavkov. Súčasťou je súbor *webanalyzer.properties*, v ktorom môže užívateľ nastaviť aké rozhodovacie kritéria sa majú použiť a za akých podmienok sa má stránka identidikovať ako validná.

WebAnalyzer je navrhnutý tak, aby bolo jednoduché pridať nové rozhodovacie kritérium do kódu. Poskytuje rozhrania, ktoré musí nové kritérium implementovať.

5.1 Návrh tried modulu WebAnalyzer

Návrh tried modulu WebAnalyzer je zobrazený na obrázku Class Diagram modulu WebAnalyzer. Triedy sú zabalené do jednotlivých balíkov, ktorých triedy vykonávajú podobnú funkcionalitu, alebo sú vzájomne tesne spjaté. Obrázok A.5

- *cz.webarchiv.webanalyzer.multithread* - Tento balík obsahuje triedy, ktoré sa starajú o načítavanie vlasntostí definovaných v externom súbore a triedu, ktorá poskytuje rozhranie pre prácu s celým systémom.
  - **WebAnalyzer** - Trieda poskytuje rozhranie pre prácu so systémom. Počas celého procesu je vytorená iba jedna instancia tejto triedy.
  - **PropertiesReader** - Trieda, ktorá načíta a zvaliduje hodnoty parametrov z externého súboru *webanalyzer.properties*.
  - **WebAnalyzerProperties** - Trieda slúži na uchovanie informácií načítaných z externého súboru. Ostatné triedy majú takto prístup ku všetkým informáciam definovaným užívateľom.
- *cz.webarchiv.webanalyzer.multithread.analyzer.util* - Tento balík obsahuje triedy, ktoré obsahujú často používané funkcie WebAnalyzeru. Okrem toho obsahuje triedu, kde sú definované konštanty pre jednoduchú identifikáciu rozhodovacích kritérií.

  - **FilterText** - Trieda poskytuje funkcie na prácu s textom.
  - **PropertiesReader** - Trieda, ktorá obsahuje konštanty na lepšiu identifikáciu rozhodovacích kritérií.

- *cz.webarchiv.webanalyzer.multithread.analyzer* - Tento balík obsahuje triedy, ktoré uchovávajú celý stav procesu a poskytujú hlavný komunikačný kanál pre ostatné triedy.

  - **UrlAnalyzer** - Trieda, ktorá inicializuje všetky požadované rozhodovacie kritéria a spúšťa analýzu. Okrem toho počíta štatistiky a rozhoduje či je stránka validná.
  - **PointsCounter** - Trieda, ktorá uchováva počet dosiahnutých bodov pre analyzovanú stránku. Pri nájdení hľadanej informácie vyhľadávače zvyšujú počet bodov. Na konci sa porovná dosiahnutý počet bodov s požadovanou minimálnou hranicou a rozhodne sa či je stránka validná alebo nie.
  - **ProcessedCrawlURI** - Trieda, ktorá uchováva informácie o spracovávanej URI. Jej názov, typ dokumentu, textový obsah a kolekciu odkazov nájdených v textovom obsahu.

- *cz.webarchiv.webanalyzer.multithread.mime* - Tento balík obsahuje triedy, ktoré identifikujú MimeType dokumentu.

  - **Content** - Trieda, ktorá poskytuje rozhranie na volanie tejto komponenty. Pri vytvorení instancie tohto objektu sa vytvoria ostatné objekty a načítajú zoznam elementov MimeType, ktorý je uložený v externom xml súbore. Na základe informácií o dokumente sa indetifikuje jeho MimeType porovnávaním so zoznamom všetkých MimeTypov (napr. podľa koncovky).
  - **MimeType** - Trieda, ktorá uchováva informácie o danom MimeType. Názov koncovky, popis, MagicMimeType a iné.
  - **MimeTypes** - Táto trieda poskytuje úlohu repozitára všetkých MimeType, prostredníctvom ktorých dokáže identifikovať MimeType dokumentu na základe poskytnutých informácií.
  - **MimeTypesReader** - Trieda jednoducho načíta zoznam MimeType z externého xml súboru a naplní ním objekt MimeTypes.
  - **MimeTypesException** - Výnimka pri identifkovaní MimeType.

- *cz.webarchiv.webanalyzer.multithread.managers* - Tento balík obsahuje triedy, ktoré sú nevyhnutné pre prácu niektorých vyhľadávačov. Inicializujú databázové spojenia, otvárajú súbory a iné zdroje používané vyhľadávačmi. Musia sa inicializovať pred spustením analýzy a musia poskytovať synchronizované metódy, aby sa dali použiť viacerými vláknami.

  - **WebAnalyzerManager** - Hlavná trieda, ktorá inicializuje potrebné objekty IManager, tak ako je definované užívateľom. Po skončení procesu volá metodu close() každého vytvoreného objektu IManager, aby sa uzavreli všetky otvorené spojenia.
  - **IManager** - IManager je rozhranie, ktoré musia implementovať manažéri. Manažér obsahuje metódy požadované rozhraním a synchronizovanú metódu používanú odpovedajúcim vyhľadávačom. (napr. GeoIPSearcher používa GeoIPManager na získanie informácií o danej IP z databáze)

- *cz.webarchiv.webanalyzer.multithread.criteria* - Tento balík obsahuje triedy, ktoré predstaujú vyhľadávače fungujúce na základe konkrétnych kritérií. Všetky vyhľadávače musia implementovať rozhranie ISearcher.

  - **ISearcher** - Rozhranie ISearcher požaduje metódu *search(ProcessedCrawlURI curi)*, ktorá vyhľadáva konkrétnu informáciu a metódu *toSring()*, ktorá poskytuje štatistiky získané počas spracovania danej ProcessedCrawlURI. Realizácia napr. EmailSearcher, GeoIPSearcher, PhoneSearcher...

5.2 Workflow modulu WebAnalyzer

Postup identifikovania URI sme rozdelili na 4 časti. Každá časť predstavuje jednu metódu volanú zo systému Heritrix (konkrétne z modulu ExtractorWebAnalyzer).

5.2.1 Inicializácia

WebAnalyzer sa musí pred použitím nainicializovať. Inicializácia WebaAnalyzeru sa uskutočňuje zároveň s inicializáciu prosesoru ExtractorWebAnalyzer, ktorý je súčasťou Heritrixu. Postup inicializovania: Obrázok A.1

- **1.initialize()** - Systém Heritrix volá prostredníctvom procesoru ExtractorWebAnalyzer, metódu *inicialize()*.
- **2.create PropertiesReader** - Pri inicializácií je nutné načítať nastavené vlastnosti v externom súbore *webanalyzer.properties*, ktoré definujú požiadavky na identifikáciu.
- **3. loadPropertiesReader** - Metóda načíta hodnoty pre konkrétne vlastnosti a zvaliduje ich. Ak je hodnota nevalidná, systém vyhodí výnimku.
- **4. create WebAnalyzerProperties** - PropertiesReader vytvorí objekt WebAnalyzerProperties a naplní mu všetky hodnoty jednotlivých vlastností. WebAnalyzerProperties bude uchovávať tieto informácie aby k nim mohli ostatné triedy pristupovať.

- **5. create WebAnalyzerManager** - Vytvára sa objekt WebAnalyzerManager, ktorý spravuje ostatných potrebných manažérov .
- **6. initializeManagers()** - Každý manažér reprezentovaný rozhraním IManager sa musí pred použitím nainicializovať.
- **7. getSearchersToUse()** - WebAnalyzerManager potrebuje zistiť, ktorých manažérov má vytvoriť.
- **8. return searchers** - WebAnalyzerProperties vráti zoznam vyhľadávačov, ktoré požadoval uživateľ pri konfigurácií. Na základe vyhľadávačov WebAnalyzerManager spozná, ktorých manažérov má vytvoriť.
- **9. create IManager** - Vytvárajú sa manažéri. Každý manažér bude predstavovať práve jednu instanciu, ktorá sa bude používať viacerými vláknami.
- **10. init()** - Každý manažér sa musí inicializovať, aby otvoril potrebné spojenia do databáze, súboru alebo iných zdrojov.

5.2.2 Identifikácia MimeType dokumentu

Pred samotnou analýzou dokumentu je nutné overiť jeho typ, aby sme sa vyhli textovej analýze binárneho dokumentu. Užívateľ však môže chcieť identifikovať stránky iba podľa IP adresy alebo názvu URL. V tomto prípade by mal užívateľ nakonfigurovať WebAnalyzer tak, aby používal iba odpovedajúce kritéria pri identifikácií. Moduly na vyhľadávanie e-mailov, telefónych čísel alebo slov v slovníku by mali byť vypnuté. Obrázok A.2

- **1. isContentTypeText()** - Heritrix volá metódu isContentTypeText(), ktorá spúšťa identifikáciu MimeType dokumentu z danej URI.
- **2. create Content** - WebAnalyzer vytvára objekt Content, ktorý nainicializuje potrebné objekty a poskytuje metódu na identifikovanie MimeType. Instancia objektu Content sa vytvára pri každom volaní metódy *isContentTypeText()*. Každé vlákno vlastní svoju instanciu tohto objektu.
- **3. init()** - Metóda *init()* inicializuje ostatné potrebné objekty. Objekty MimeTypes a MimeTypesReader sú konštantné, ich instancia sa vytvára len raz, pri prvom volaní tejto komponenty.
- **4. addMimeTypes()** - Objekt MimeTypes volá túto metódu, aby naplnil svoju kolekciu všetkých MimeType, ktoré sa neskôr použijú pri identifikácii neznámeho dokumentu.
- **5. readAllMimeTypes()** - Metóda vytvára spojenie s externým XML súborom a číta z neho jednotlivé elementy MimeType.

- **6. create MimeType** - Z každého elementu MimeType zapísaného v XML dokumente sa vytvorí odpovedajúci objet MimeType a pridá sa do kolecie všetkých MimeType záznamov.
- **7. return MimeTypes** - Naplní sa kolekcia MimeType záznamov v objekte MimeTypes. Tým sa dokončí potrebná inicializácia, ktorá sa už viac krát nebude opakovať. Pri ostatných volaniach sa zavolá priamo metóda *getContentType()*.
- **8. getContentType** - Metóda, ktorá hľadá MimeType skúmaného dokumentu.
- **9. getMimeTypes()** - Podľa informácií získaných zo skúmaného dokumentu sa hľadajú odpovedajúce MimeType.
- **10. getMimeTypes** - Objekt sa snaží zistiť MimeType skúmaného dokumentu podľa názvu URI alebo bytov reprezentujúcich skúmaný dokument.
- **11. return mimeTypes** - Vráti sa zoznam možných odpovedajúcich MimeType.
- **12. return MimeType** - Zo zoznamu odpvedajúcich MimeType sa jeden vyberie a ten sa vráti ako identifikovaný MimeType.
- **13. return true if text** - WebAnalyzer vráti systému Heritrix logickú hodnotu true, iba ak je identifikovaný MimeType typu *text*.

5.2.3 Analýza

Po nainicializovaní modulu WebAnalyzer môžeme začať používať jeho metódu na analyzovanie. Obrázok A.3

- **1. run()** - Heritrix volá metódu *run()*, ktorá spúšťa analýzu. Metóde run sú predané parametre, ktoré predstavujú informácie získané z Heritrixu. Medzi významné patria názov URI, ktorá sa analyzuje, jej textový obsah a množina odkazov objavených na tejto URI.
- **2. create UrlAnalyzer** - Pri každom volaní metódy *run()* sa vytvorí nová instancia objektu UrlAnalyzer, ktorý si pamätá stav pri spracovaní konkrétnej URI. Tým dosiahneme, že sa bude uskutočňovať analýza paralelne vo viacerých vláknach. Samotný Heritrix pri defaultnom nastavení spúšťa 50 vláken, ktoré potom môžu súčasne používať modul WebAnalyzer.
- **3. create PointsCounter** - Pre každú instanciu UrlAnalyzer sa vytvorí instancia PointsCounter, ktorá si pamätá počet nazbieraných bodov pre analyzovanú URI.
- **4. getSearchersToUse()** - Z už vytvoreného objektu WebAnalyzerProperties si získame zoznam vyhľadávačov, ktoré sa majú pri analýze použiť.

- **5. return searcherToUse** - Metóda vráti zoznam vyhľadávačov, definovaný užívateľom.
- **6. create ISearcher** - Pre každý záznam zo zoznamu vyhľadávačov sa vytvorí odvedajúci objekt.
- **7. setPointsCounter()** - Každá instancia vyhľadávača si priradí už vytvorený objekt PointsCounter, ktorý bude používať. Každý vyhľadávač pri nájdení hľadanej informácie pripočíta alebo odpočíta počet bodov analyzovanej stránke.
- **8. getProperties()** - Počet bodov s ktorými vyhľadávač pracuje je taktiež definovaný v externom súbore a je prístupný cez objekt WebAnalyzerProperties, z ktorého si vyhľadávače všetky hodntoy získajú.
- **9. return props** - WebAnalyzerProperties vráti hodntoty vlastností, ktoré používa konkrétny vyhľadávač.
- **10. setIManager()** - Niektoré vyhľadávače pristupujú k databázam alebo externým súborom, a preto používajú manažerov, ktoré poskytujú synchronizované metody volané odpovedajúcim vyhľadávačom. Pri vytvorení si musí každý vyhľadávač asociovať svoj už vytvorený manažér. Tým sa končí vytváranie vhľadávačov, ktoré sa môžu použiť pri analýze.
- **11. analyze()** - Metóda analyze() už má pripravené všetky podklady pre spustenie analýzy, tak aby sa použili tie vyhľadávače, ktoré boli nadefinované.
- **12. create ProcessedCrawlURI** - Pri analýze URI sa vytvorí objet ProcessedCrawlURI, ktorý zabaľuje informácie o URI predanej z Heritrixu (názov, typ, obsah, odkazy).
- **13. return curi** - Vytvorená instancia sa vráti objektu UrlAnalyzer, aby ju mohol použiť ako parameter pre každý vyhľadávač.
- **14. search(curi)** - V cykle sa spustí každý nadefinovaný vyhľadávač jeho spúšťacou metódou *search(ProcessedCrawlURI curi)*. Každý vyhľadávač pracuje nad konkrétnými atribútmi objektu ProcessedCrawlURI.
- **15. uses IManager** - Pri vyhľadávaní používajú niektoré vyhľadávače svojich manažérov.
- **16. increment()** - Pri nájdení hľadanej informácie sa inkrementuje počet bodov objektu PointsCounter. PointsCounter poskytuje synchronizované metódy, aby bol zachovaný konzistentý stav v multivláknovom prostredí.
- **17. getStatistics(curi)** - Po spracovaní skúmanej URI všetkými vyhľadávačmi WebAnalyzer vypíše získané štatistiky. Vypisovanie je nastavené do viacerých logovacích súborov.

- **18. isValid()** - Na základe analýzy sa vyhodnotí či je stránka validná alebo nie.
- **19. getPoints()** - UrlAnalyzer požaduje počet dosiahnutých bodov z objektu PointsCounter.
- **20. return points** - Vráti sa počet dosiahnutých bodov, ktoré sa použijú na výsledné vyhodnotenie spracovávanej URI.
- **21. valid = (points >= minPointsToValid)** - Vyhodnotenie validity funguje na základe porovnania získaných bodov a bodov predstavujúcich hranicu, ktorú treba prekonať aby bola stránka identifikovaná ako validná. Táto hranica je definovaná užívateľom v externom súbore.
- **22. return valid** - UrlAnalyzer vracia logickú hodnotu.
- **23. return valid** - WebAnalyzer predáva Heritrixu informáciu, či je analyzovaná URI identifikovaná ako validná.

5.2.4 Uzavrenie

Keď skončí *crawl job* v Heritrixe, tak sa následne zatvárajú všetky procesory, ktoré boli použité. Preto sme v integračnom procesore ExtractorWebAnalyzer pri jeho finalizovaní zavolali metódu na finalizáciu modulu WebAnalyzer. Tým zabezpečíme, že sa WebAnalyzer správne ukončí, aby nevznikali zbytočné problémy v spojení s Heritrixom. Obrázok A.4

- **1. close()** - Heritrix volá metódu *close()* objektu WebAnalyzer, ktorá ukončí všetky spojenia a iné relácie.
- **2. closeManagers()** - Zatvoria sa spojenia s DB, súbormi a inými zdrojmi používanými manažérmi.
- **3. close()** - Každý manažér volá svoju metódu *close()*.
- **4. clearProperties()** - Nakoniec sa vymažú nastavenia definované v objekte WebAnalyzerProperties.

5.3 Konfigurácia systému

Konfigurácia celého systému sa skladá z konfigurácie Heritrixu, prostredníctvom jeho webového rozhrania a z konfigurácie WebAnalyzeru cez externý súbor *webanalyzer.properties*. Heritrix musíme nastaviť tak aby používal naše integračné procesory (ExtractorWebAnalyzer, LinksScoper, ARCWriterProcessor).

5.3.1 Konfigurácia Heritrixu

Požiadavky na Heritrix.

- Na vstupe by mali byť semienka, ktoré predstavujú odhodené odkazy zo *sklizne_cz_2007*. Tieto semienka sú veľkí konkurenti na české stránky mimo národnej domény.
- Nastavenie väčšiny procesorov môže ostať, tak ako sú pôvodne nastavené pri vytvorení novej úlohy - *crawl job* Heritrixu. V odpovedajúcich reťaziach však musíme pridať naše integračné procesory.
  - *ExtractorWebAnalyzer* - do reťaze *Extractor processing chain*musíme pridať ExtractorWebAnalyzer, ktorý priamo používa modul WebAnalyzer. Procesor musíme pridať na samý koniec reťaze.
  - *ARCWriterProcessorWebAnalyzer* - do reťaze *Write/Index processing chain* musíme pridať tento procesor, ktorý nearchivuje DNS záznamy. Pôvodný procesor ARCWriterProcessor odstránime.
  - *LinksScoperWebAnalyzer* - Tento procesor pridáme do reťaze *Post-processing chain*. Procesor zabezpačuje správne spracovanie odkazov a nastavovanie príslušných príznakov pre spracovávané odkazy. Pôvodný procesor LinksScoper odstránime.
- Po nastavení procesorov musíme ďalej nastaviť moduly *Deciding rule* a to v nasledujúcom poradí. Tieto nastavenia sa nachádzajú v záložke *submodules*. Bližšie informácie sú v užívateľskej príručke Heritrixu.
  - *1. AcceptDecideRule*
  - *2. TooManyHopsDecideRule*
  - *3. TooManyPathSegmentsDecideRule*
  - *4. MatchesRegExpDecideRule*
- Posledné nastavenie v Heritrixe je v záložke *settings*, kde nastavíme konkrétne hodnoty pre moduly Deciding rule.
  - *MatchesRegExpDecideRule* - Nastavíme, aby sa každý objavený odkaz, ktorý odpovedá regulárnemu výrazu odhodil. Regulárny výraz na na zistenie českých URL *^.*\.cz[\W].*$*.
  - *Počet vláken ToeThread* - Môžeme nastaviť počet vláken, ktoré budú súčasne spracovávať URI adresy. Rozsah 1-200.
  - Ďalej sa nastavujú ostatné podrobnosti, ktoré sú popísané v užívateľskej príručke.

5.3.2 Konigurácia modulu WebAnalyzer

Konfigurácia modulu WebAnalyzer sa uskutočňuje pred spustením nadefinovanej úlohy *crawl job* Heritrixu. Všetko potrebné sa nachádza v jednom externom súbore *webanalyzer.properties* Dodatok B.

V súbore *webanalyzer.properties* môžeme nájsť vlastnosti a ich hodnoty, ktoré systém WebAnalyzer používa. Medzi dôležité patria hodnoty vlastností:

- **webanalyzer.urlanalyzer.min.valid.points=200** - táto vlastnosť definuje aký počet bodov musí skúmaný dokument pri analýze dosiahnuť, aby bol identifikovaný ako validný.
- **webanalyzer.urlanalyzer.depth.toarchive=3** - táto vlastnosť slúži pre Heritrix, ktorý ju používa na archivovanie odkazov z validnej stránky do určitej hĺbky. Ak je nastavená hodnota 0, tak sa archivuje len validná stránka, spolu s jej zdrojmi (obrázky, css štýly, audio a video dokumenty), ktoré sa na nej nachádzajú. Ostanté odkazy sú ignorované. Ak nastavíme hodntotu na 3, tak sa budú okrem validnej stránky archivovať všetky odkazy z nej vedúce a to až do hĺbky 3.

Ostatné vlastnosti, ktoré sa v súbore nachádzajú sú vlastnosti konkrétnych vyhľadávačov.

- **# properties for emailSearcher** -
- **webanalyzer.searcher.email.use=1** - pri analýze sa má použiť vyhľadávač emailSearcher. Hodnota 0 znamená, že komponenta sa nemá použiť.
- **webanalyzer.searcher.email.regexp=([a-z0-9._-]+@[a-z0-9.-]+\\.cz)** - EmailSearcher vyhľadáva z daného kontextu všetky emaily. Pomocou regulárneho výrazu si môžeme nastaviť, ktoré z nich nás zaujímajú. Zadaný regulárny výraz ukazuje, že zo všetkých e-mailov hľadáme iba tie s doménou .cz.
- **webanalyzer.searcher.email.point=1** - Vždy pri nájdení hľadaného e-mailu sa pripočíta analyzovanej stránke definovaný počet bodov.

Podobne sú definované ostatné vlastnosti vyhľadávačov.

5.3.3 Spustenie Heritrixu

Po tom ako sme nakonfigurovali oba systémy môžeme spustiť nadefinovanú úlohu Heritrixu. Heririx začne prehľadávať web a s pomocou modulu WebAnalyzer bude identifikovať české stránky, ktoré následne archivuje.

Priebeh celého procesu môžeme sledovať priamo cez webové rozhranie Heritrixu. Podrobné informácie sa zapisujú do viacerých logov. Niektoré logujú čisto priebeh Heritrixu, iné logujú štatisktiky , ktoré produkuje WebAnalyzer pri analyzovaní stránok. Štatisktiky ukazujú všetky informácie o spracovávanej URI a počet dosiahnutých bodov v konkrétnych vyhliadavačoch. Samozrejme je možné po úprave logovacej úrovne v súbore *log4j.xml* logovať aj konkrétne nájdene hodnoty.

5.4 Komponenty modulu WebAnalyzer

WebAnalyzer uskutočňuje analýzu pomocou jednotlivých vyhľadávačov, ktoré sú modulárne, tak aby sa dali nové kritéria jednoducho pridávať pomocou implementovania rozhrania. Každý nový vyhľadávač musí dediť abstraktnú triedu AStatisticsSearcher, ktorá slúži na počítanie štatistík. Štatiskity vyjadrujú koľko percent zo všetkých nájdených elementov je validných. Napríklad pri hľadaní českých e-mailov sa na stránke môžu vyskytovať 4 e-maily, ale len 2 z nich budú české. Štatistiky budú uvádzať hodnotu 50% pre vyhľadávač EmailSearcher.

5.4.1 EmailSearcher

Vyhľadávač e-mailov funguje na základe prehľadávania textového obsahu spracovávanej URI. Tento textový obsah zvyčajne predstavuje zdrojový kód stránky. EmailSearcher vyhľadáva všetky e-maily pomocou regulárneho výrazu. Každý nájdený e-mail je d'alej podrobený kontrole, či spĺňa požiadavky definované užívateľom. Užívateľ si takto môže jednoducho nakonfigurovať, či hľadá e-maily určitej domény, hostu alebo nejaký konkrétny e-mail. Podobne fungujú ostatné vyhľadávače : telefónne čísla, IP adresy, html atribút lang. Pri nájdení požadovaného e-mailu EmailSearcher zvýši počet bodov danej URI.

5.4.2 PhoneSearcher

Vyhľadávač telefónnych čísel funguje podobne ako EmailSearcher. Taktiež prehľadáva textový obsah spracovávanej URI pomocou regulárnych výrazov. Tento text je však pred vyhľadávaním upravený triedou FilterText, ktorá poskytuje funkcie na odfiltrovanie nepodstatných znakov, html tagov a iných skriptov zahrnutých v zdrojovom texte stránky. Užívateľ si môže v konfiguračnom súbore sám zvoliť aké telefónne čísla hľadá, pomocou regulárneho výrazu.

5.4.3 HtmlLangSearcher

Ďalším kritériom pri identifikovaní stránky môže byť hodnota html tagu *lang*. Veľká väčšina internetových stránok má hodnotu tohto tagu definovanú. Užívateľ môže nastaviť hľadanú hodnotu v konfiguračnom súbore (cs, sk, en, atd).

5.4.4 GeoIPSearcher

Zaujímavým kritériom pri analyzovaní je kód krajiny, z ktorej stránka pochádza a zároveň kódy krajín z ktorých pochádzajú odkazy objavené na tejto stránke. Modul používa funkcie projektu MaxMind.

Projekt sa zaoberá geologickou lokalizáciou danej IP adresy. Ponúka varianty *GeoLite Country* a *GeoLite City*, ktoré lokalizujú krajinu a mesto, odkiaľ pochádza IP adresa (je hostovaná na nejakom servery). Tieto varianty ponúkajú zdarma. My použijeme *GeoLite*

Country, pretože chceme zistiť krajinu danej IP adresy. *GeoLite Country* v podstate obsahuje databázu IP adries s ich informáciami. K tejto databáze dodávajú API, pomocou ktorého sa dajú informácie o IP adrese rýchlo vyhľadať. Náš vyhľadávač používa svojho manažéra, ktorý volá funkcie z tohoto API. Manažér musí pred používaním inicializovať databázu *GoeLite Country*. Pri vyhľadávaní nám API vráti kód krajiny danej IP adresy vo formáte cz, sk a podobne. Ak sa kód krajiny zhoduje s kódom zadaným užívateľom, tak sa analyzovanej stránke pripočítajú body. Keďže sa Internet stále rozrastá je nutné túto databázu občas aktualizovať.

5.4.5 DictionarySearcher

Táto komponenta slúži na vyhľadávanie slov v konkrétnom slovníku. Pomáha pri identifikácií českého jazyka na základe slov nachádzajúcich sa v kontexte stránky. Pred spracovaním textu je použitý filter, ktorý upraví text do vhodnej podoby. Odstránia sa skripty, kód a nečitateľné znaky. Prefiltrovaný text sa ďalej upraví na jeden reťazec, obsahujúci konečnú postupnosť slov a medzier. Vyhľadávač takto upravený text parsuje po jednom slove a volá metódu svojo manažéra, ktorý toto slovo vyhľadáva v slovníku. Ak sa slovo nájde, tak sa pripočíta počet bodov definovaný užívateľom.

5.4.5.1 Slovník slov

Slovník slov je vlastne zoznam slov. Pri vytváraní slovníka českých slov, som použil korpus českého jazyka ako základný zdroj. Slovník musí byť rýdzo český. To znamená, že nemôže obsahovať slová, ktoré sa vyskytujú aj iných jazykoch, a Ppreto bolo nutné tieto slová odstrániť. Napríklad slovo *auto* sa nachádza v českom, slovenskom aj anglickom jazyku. Najviac českých slov sa prekrýva so slovenským jazykom a preto bolo nutné tieto slová odstrániť pomocou korpusu slovenského jazyka. Výsledná fyzická podoba zoznamu rýdzo českých slov bol textový súbor, v ktorom bolo na každom riadku jedno slovo.

5.4.5.2 Vytvorenie indexu

Ďalšia otázka bola, ako sa bude v tomto slovníku vyhľadávať. Slovník mal okolo 14 MB, takže bol pomerne veľký. Prvá varianta bola pamätať si celý zoznam slov v nejakej štruktúre v pamäti. Druhá varianta bola spraviť k slovníku index a pomocou neho vyhľadávať.

Pred vytvorením indexu je nutné slová v slovníku zoradiť podľa abecedy. Následne sa vytvoril index Príklad je znázornený v tabuľke: Tabuľka 5.1

DictionarySearcher používa dvoch manažérov. Jeden sa stará o prístup k hodnotám indexu a druhý načítava slovo od konkrétneho byte až po koniec riadku (znak $\backslash n$). Potom stačilo naprogramovať klasické binárne vyhľadávanie, ktoré si ako pivota zvolí hodnotu n z indexu, vyjadrujúcu byte na ktorom začína odpovedajúce slovo. Manažér na načítavanie slov zo slovníka takto môže preskočiť daný počet bytov a načítať priamo odpovedajúce slovo. Toto slovo sa následne porovná so skúmaným slovom. Ak sa skúmané slovo nájde v slovníku tak sa pre analyzovanú stránku pripočítajú body.

slovo	odpovedajúci index
abandonovat	0
abatyše	12
abdikace	20
abdikovat	29

Tabuľka 5.1: Index k slovníku

5.4.5.3 Synchronizácia

Každé vlákno si pri vyhľadávaní slov vytvorí vlastnú instanciu objektu DictionarySearcher. Samotné čítanie zo súboru, ktoré poskytuje odpovedajúci manažér je prístupné cez synchronizovanú metódu.

5.4.5.4 Ďalšie možnosti

Takto vytvorená komponenta na vyhľadávanie sa dá pužiť na mnoho ďalších modulov. Napríklad vyhľadávanie v zozname českých miest alebo vyhľadávanie vulgárnych slov. Pri nájdení vulgárneho slova budeme naopak odpočítavať body analyzovanej stránke, pretože WebArchív nechce archivovať stránky s vulgárnym, pornografickým či rasistickým obsahom.

5.4.6 Ďalšie vyhľadávače ktoré chceme implementovať v budúcnosti

5.4.7 URLSearcher

Pri analyzovaní stránky sú všetky odkazy na nej objavené hodnotnými informáciami pre ďalšie kriréria. Ak identifikujeme český web, tak odkazy vedúce z analyzovanej stránky na stránky s českou doménou by mali byť taktiež zahrnuté do bodového hodnotenia analyzovanej stránky. Je totiž pravdepodovné, že takéto stránky s odkazmi do českej domény môžu byť pre český národ hodnotné a preto je dôvod ich archivovať. Vyhľadávač bude jednoducho prehľadávať objavené odkazy pomocou regulárneho výrazu a s každým nájdeným odkazom pripočítava stránke body. Užívateľ si môže nastaviť akú doménu požaduje.

5.4.8 ValidLinkSearcher

Táto komponenta bude v objavených odkazoch vyhľadávať odkazy *hodnotné* pre Český národ. Pod pojmom *hodnotný* odkaz máme na mysli odkaz vedúci na typicky českú stránku. Zoznam hodnotných odkazov môžeme získať zo známych českých portálov a rozcestníkov (seznam.cz a iné). Vyhľadávanie bude implementované s využitím projektu *Apache Lucene*, ktoré poskytuje na tento typ činnosti vhodné API.

5.4.9 ForbiddenLinkSearcher

Pri analyzovaní stránky je určite výhodné preskúmať všetky jej objavené odkazy. Ak medzi objavenými odkazmi existujú také, ktoré jasne referujú na neprijateľné weby s pornografickýcm obsahom, musíme ich identifikovať a značne znížiť body analyzovanej stránke. Najprv musíme získat zoznam takýchto stránok, aby sme v ňom mohli následne prehľadávať. Na prehľadávanie použijeme nástroj *Apache Lucene*.

5.4.10 ForbiddenDictSearcher

Motiváciou WebArchivu je eliminovať stránky, ktoré obsahujú informácie zákazané zákonom. V tomto prípade môže byť slovník zakázaných slov jedným z kritérií. Pri nájdení nejakého slova z tohto slovníka odčítame analyzovanej stránke body, čím znížime možnosť archivovania stránok zakázaných zákonom. Tento modul zatiaľ nie je implementovaný, pretože čakáme na výsledky práce, ktorá sa zaoberá práve identifikovaním takýchto stránok. Tento systém sa pokúsime zakomponovať do nášho modulu.

5.4.11 TrigramSearcher

V blízkej budúcnosti plánujeme zahrnúť nový modul, ktorý bude identifikovať jazyk pomocou trigramov. Na základe odprúčania z laboratórie spracovania prirodzeného jazyka chceme tento modul implementovať a porovnať jeho výsledky s vyhľadávačom DictSearcher. Podľa skúseností z laboratórie funguje identifikácia pomocou trigramov spoľahlivo. Možno sa nám podarí prebrať existujúci systém z laboratórie, aby sme ho nemuseli implementovať. Potom ho stačí len zakomponovať do modulu WebAnalyzer a využívať jeho funcionalitu.

5.4.12 Iné vyhľadávače

Plánujeme vytvoriť vyhľadávač, ktorý bude používať *GoogleAPI* a projekt *Whois*. Pomocou *Whois* môžme zistiť adresu majitľa analyzovanej stránky). Ide o problém vyžadujúci samostatnú analýzu, čo už presahuje rámec tejto práce.

Kapitola 6

Testy

Systém v tejto podobe zatiaľ nebol spustený tak aby celý proces sklizne mimo národnú doménu dobehol do konca. Maximálna dĺžka behu systému bola okolo 24 hodín. Kedže sklizeň 2007 ešte nebola presunutá do Národnej knihovny nemal som k dispozícií niekoľko TB na diskovom poli, ktoré som potreboval na rozsiahlejší test. To som však nemohol ovplyvniť. Momentálne je už sklizeň presunutá a môžeme spustiť testy, aby sme získali nové presnejšie výsledky. Je možné že objavíme aj nové chyby, ktoré sa pri sklizni trvajúcej 1 deň neobjavili. Celý systém je ešte vo vývoji a je otestovaný iba v rámci aktuálnych možsnotí. Detailné testy bude nutné vykonať hneď ako bude viac priestoru.

Z testovania sem vybrali určitý výsek štatistík pre zobrazenie do grafickej podoby. Testovali sme na vzorke 450 adries (odhodených odkazov zo sklizne 2007). Systém ich analyzoval v priebehu 2 minút na osobnom notebooku s internetovým pripojením 100 Mb/s. V Heritrixe sme nastavili maximálne 5 vláken. Spracovanie na výkonnom serverovom počítači je oveľa rýchlejšie a je možné nastaviť Heritrixu maximálny počet 200 vláken. Výsledky sú zobrazené v grafoch, umiestnených v prílohe. Obrázok A.6, Obrázok A.7, Obrázok A.9, Obrázok A.8

Graf na obrázku popisuje modrou farbou koľko elementov sa našlo na analyzovanej stránke pomocou konkrétneho vyhľadávača. Napríklad pri vyhľadávači GeoIPSearcher považujeme za element každý odkaz, ktorý sa nachádza na analyzovanej stránke. Pri DictSearcher je elementom každé slovo a tak to platí pre ostatné vyhľadávače. Červená farba udáva, koľko z nájdených (modrých) elementov je validných (v našom prípade českých). Počet nájdených elementov je znázornený na osi Y. Každá hodnota na ose X predstavuje jednu analyzovanú stránku. V každom grafe je 450 analyzovaných stránok.

Kapitola 7

Záver

Na záver treba uviesť, že vývoj systému na rozpoznanie a archiváciu webu mimo národnú doménu bude ešte nejakú dobu pokračovať. Postupom času sa budú pridávať ďalšie kritéria a celý systém sa bude upravovať podľa nových požiadavkov, ktoré sa objavia po väčších skúsenostiach s týmto systémom. Počas ďalšieho vývoja sa budeme snažiť, aby bol systém čo najviac modulárny aby boli zmeny či pridanie novej funkcionality jednoduché. Zároveň sa snažíme vyvíjať systém tak, aby sa mohol použiť nielen na identifikovanie českého webu, ale aby bolo možné identifikovať aj weby iných národov. Jedinou úlohou pri identifikovaní webu iného národa, je zohnať odpvedajúce zdroje ako sú slovníky, zoznamy hodnotných odkazov alebo zoznamy miest. Tieto zdroje potom upravíme do požadovanej podoby aby s nimi WebAnalyzer dokázal pracovať.

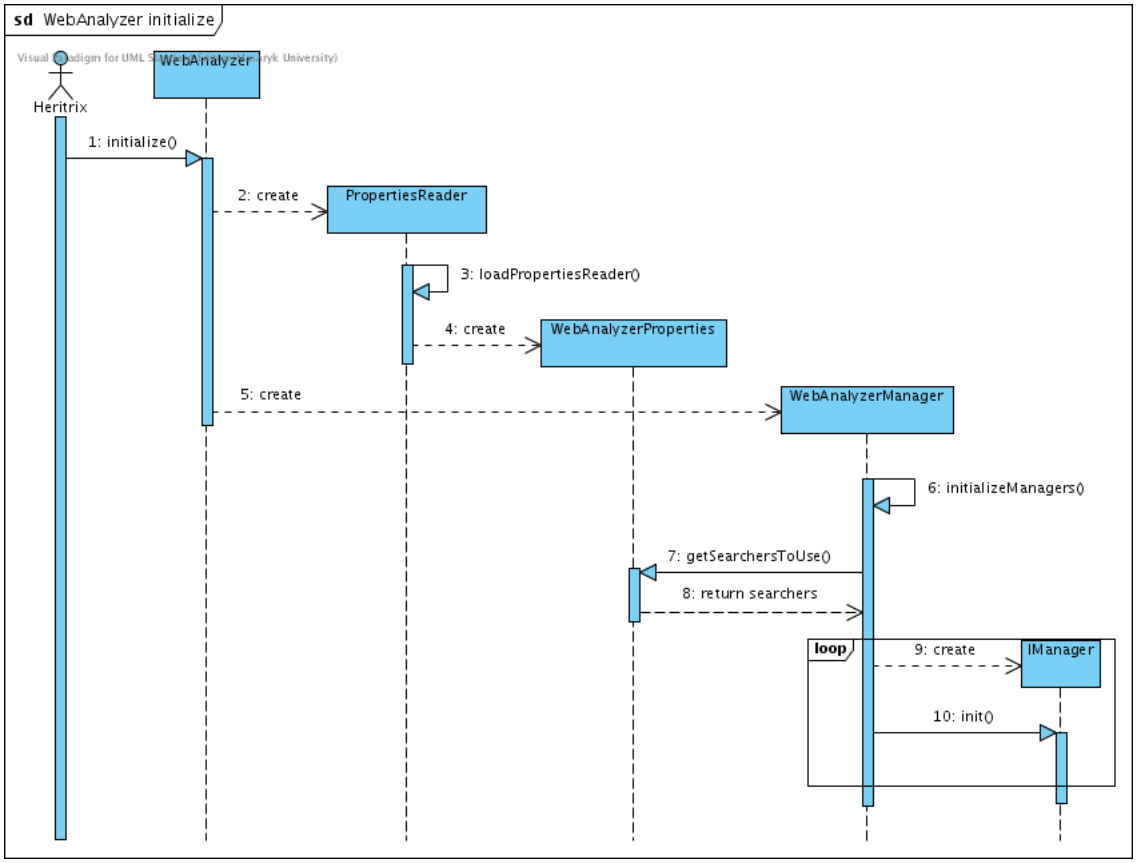
Ak bude systém pracovať spoľahlivo, tak sa ho pokúsime vydať pod licenciou GNU ako súčasť Heritrixu. Ďalším dôležitým mílnikom je nová verzia Heritrixu pod názvom *Heritrix 2*, ktorá vyšla nedávno a predstavuje veľký pokrok od predchodzích verzií. Budeme sa snažiť WebAnalyzer prospôsobiť novej verzii Heritrixu alebo dokonca vylepšiť celú integráciu s novými funkciami, ktoré *Heritrix 2* ponúka. Modul WebAnalyzer bude voľne dostupný na stránkach webarchívu a rovnako sprístupnime aj skompilovanú spustiteľnú verziu Heritrixu s integrovaným modulom WebAnalyzer.

Dúfame, že systém nájde svoje využitie a pomôže plniť ciele WebArchivu.

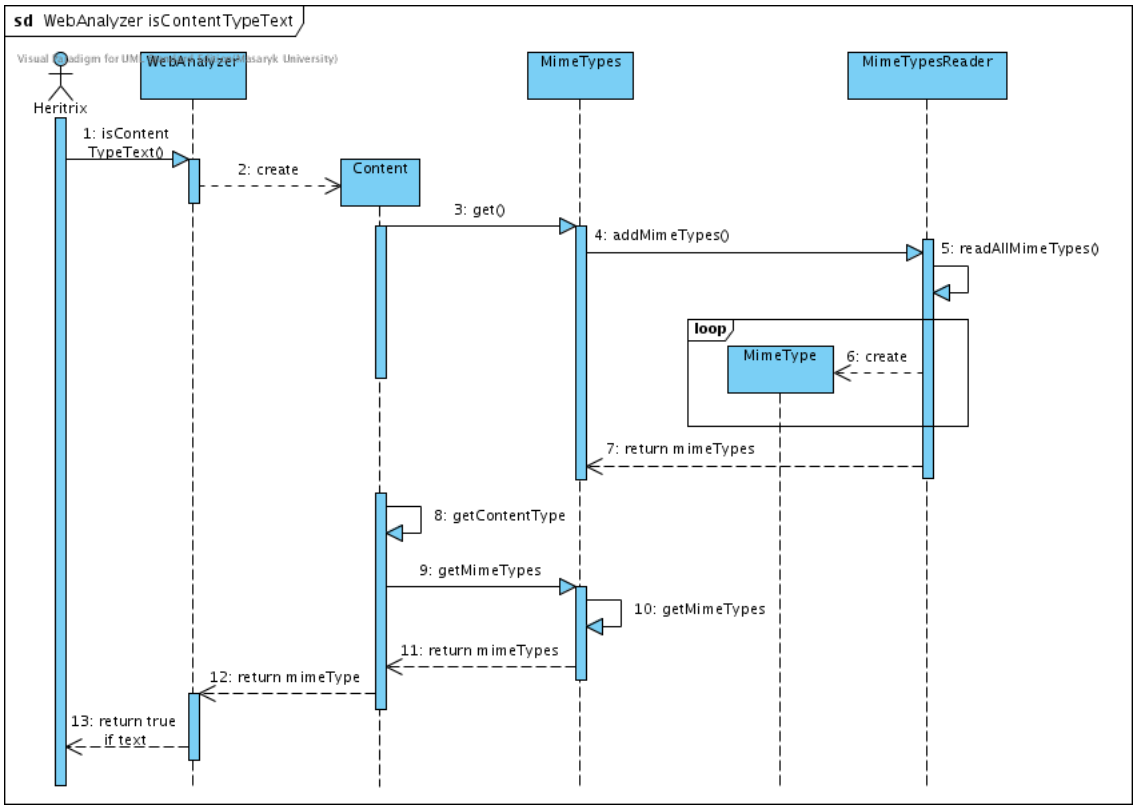
Dodatok A

A

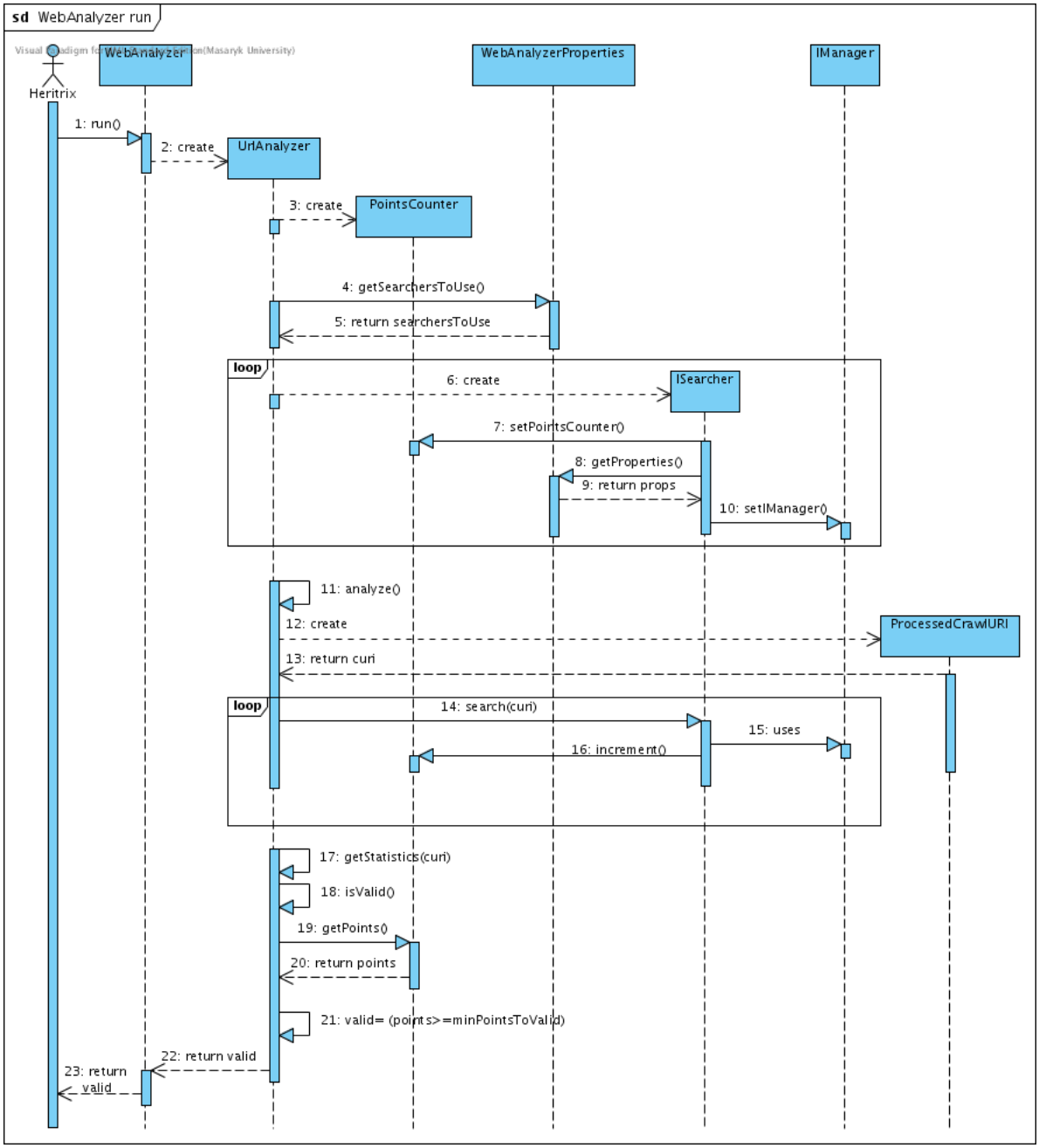
Sekvenčné diagramy popisujúce systém WebAnalyzer:



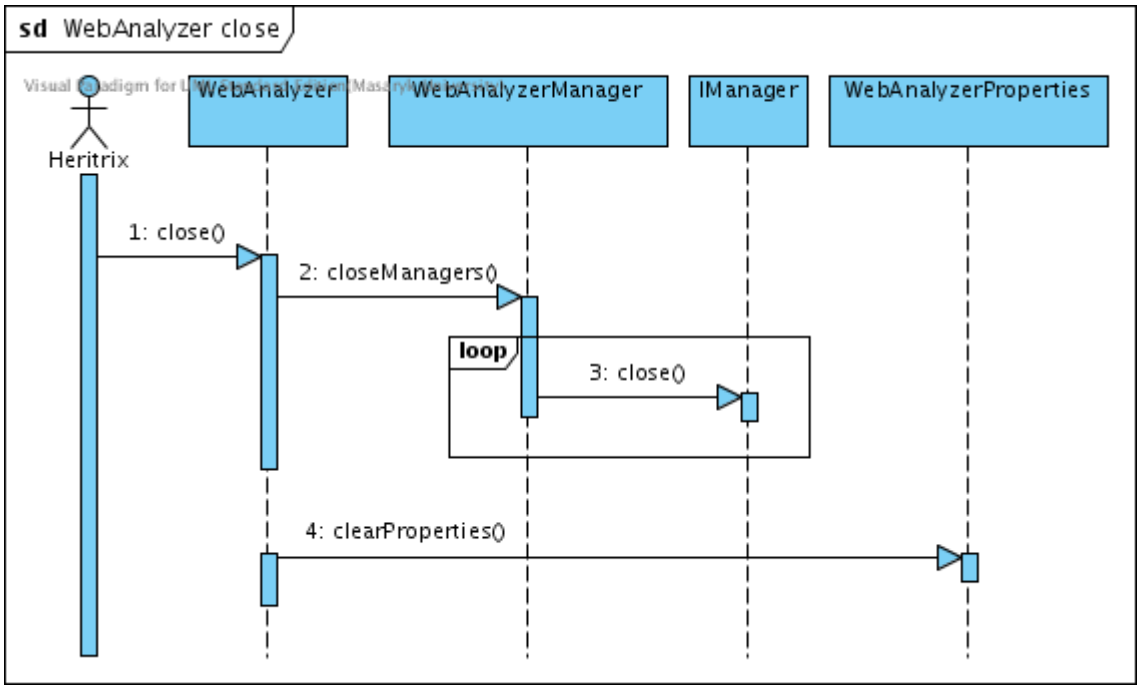
Obrázok A.1: Diagram popisujúci inicializáciu systému WebAnalyzer



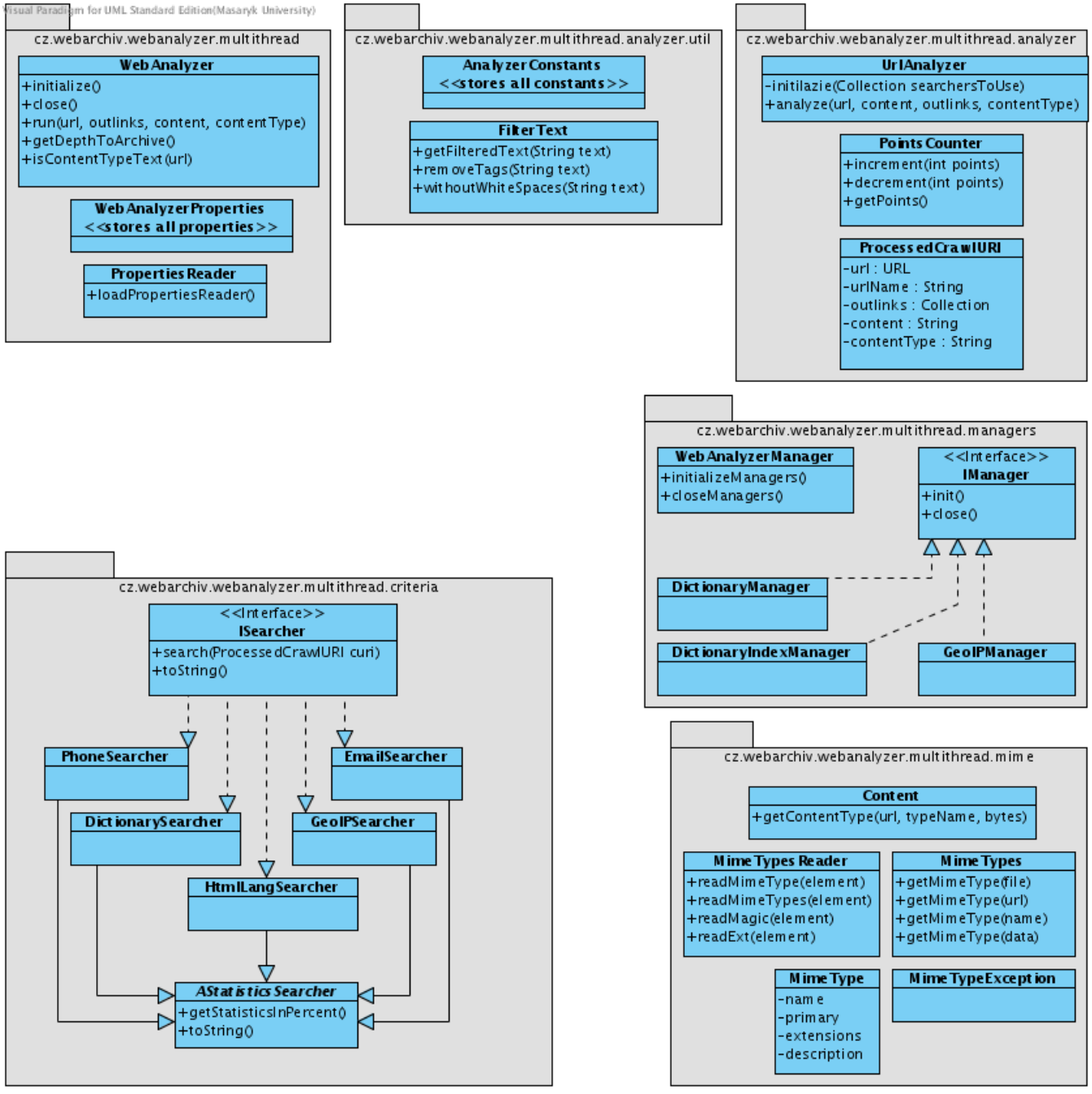
Obrázok A.2: Diagram popisujúci identifikovanie Mime Type dokumentu



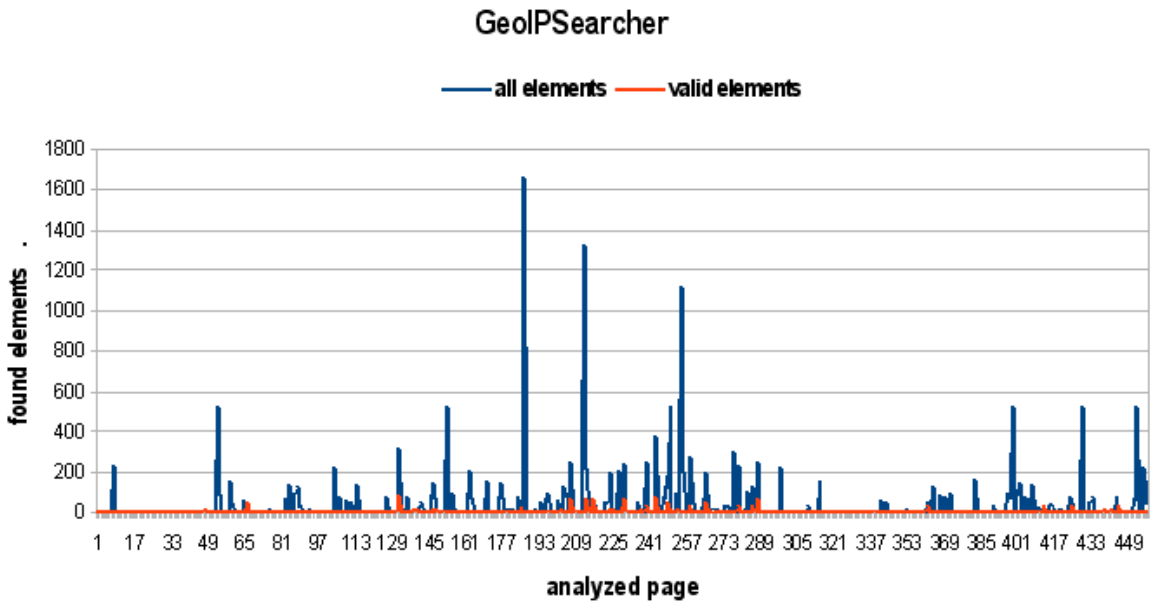
Obrázok A.3: Diagram popisujúci priebeh analýzy dokumentu



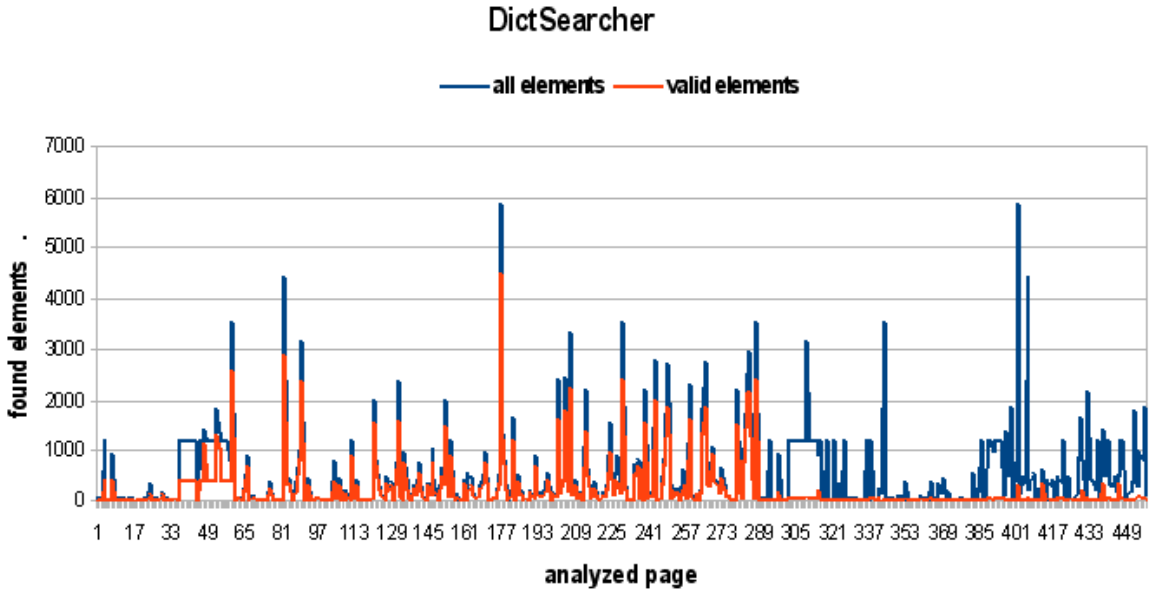
Obrázok A.4: Diagram popisujúci finalizáciu systému WebAnalyzer



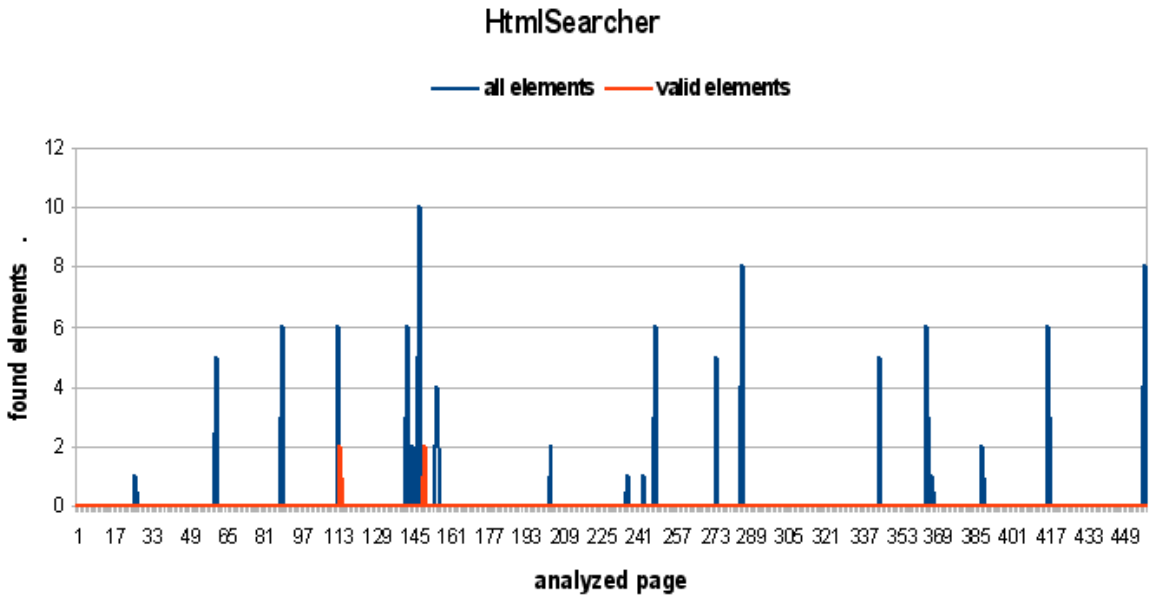
Obrázok A.5: Class Diagram modulu WebAnalyzer



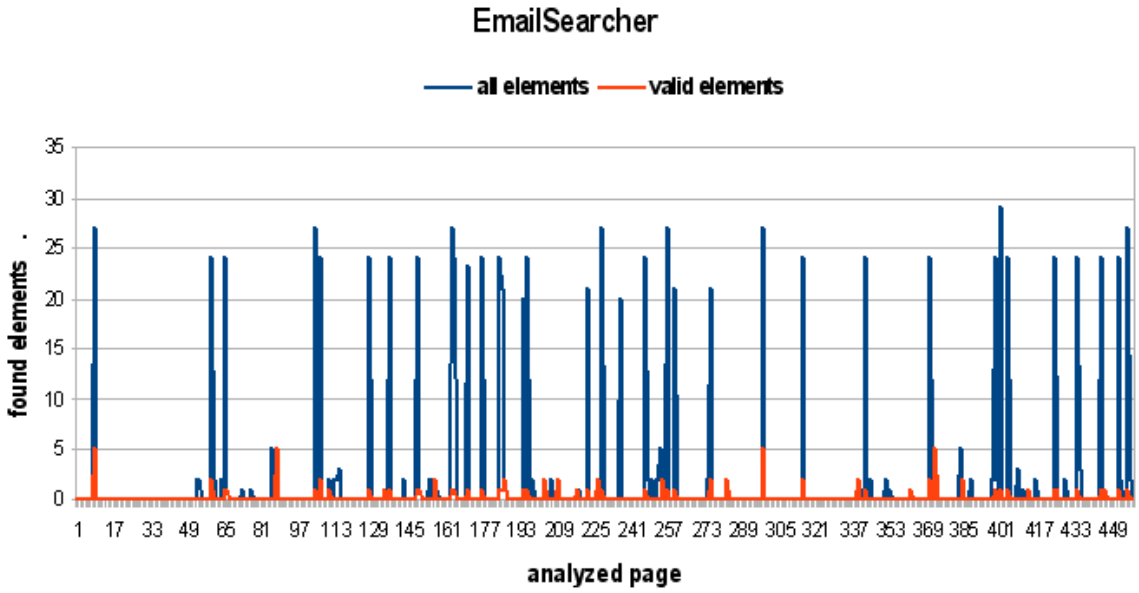
Obrázok A.6: GeoIPSearcher výsledky



Obrázok A.7: DictSearcher výsledky



Obrázok A.8: HtmlSearcher výsledky



Obrázok A.9: EmailSearcher výsledky

Dodatok B

B

Ukážka konfiguračného súboru *webanalyzer.properties*:

```
# Properties file for WebAnalyzer.
# As first read the readme.txt
# point properties must be in interval from 0 to 10

# properties for geoIpSearcher
WebAnalyzer.searcher.geoip.use=1
WebAnalyzer.searcher.geoip.point=1
WebAnalyzer.searcher.geoip.countrycode=cz

# properties for dictionarySearcher
WebAnalyzer.searcher.dictionary.use=1
WebAnalyzer.manager.dictionary.language=cz
WebAnalyzer.searcher.dictionary.point=1

# properties for emailSearcher
WebAnalyzer.searcher.email.use=1
WebAnalyzer.searcher.email.regexp=([a-z0-9._-]+@[a-z0-9.-]+\\.cz)
WebAnalyzer.searcher.email.point=1

# properties for phoneSearcher
WebAnalyzer.searcher.phone.use=1
WebAnalyzer.searcher.phone.regexp=\\+420 [0-9]{3} ?[0-9]{3} ?[0-9]{3}
WebAnalyzer.searcher.phone.point=1

# properties for htmlLangSearcher
WebAnalyzer.searcher.htmllang.use=1
WebAnalyzer.searcher.htmllang.regexp=lang ?= ?[\"|']? ?cs ?[\"|']?
WebAnalyzer.searcher.htmllang.point=1

# properties for urlAnalyzer, points je typu long
WebAnalyzer.urlanalyzer.min.valid.points=200
# min 0 - archives only first valid url with its images, bin links,
# max 1000000 - archives up to depth a milion of links refered from a valid url
WebAnalyzer.urlanalyzer.depth.toarchive=0
```

Dodatok C

C

Súčasťou práce je priložené CD, kde sú k dispozícií jednotlivé ukážky:

- zdrojový kód upraveného systému Heritrix
- zdrojový kód systému WebAnalyzer
- spustiteľná verzia systému Heritrix s integrovaným systémom WebAnalyzer
- tato práca vo formáte PDF
- tato práca vo formáte HTML

Referencie

[1] WebArchiv: *WebArchiv - archiv českého webu [online]*, WebArchiv, 09.04.2008, <http://www.webarchiv.cz>. 2

[2] WebArchiv: *Kritéria výběru webových zdrojů [online]*, WebArchiv, 09.04.2008, <http://www.webarchiv.cz/kriteria>. 2.1

[3] Internet Archive: *Overview of the crawler [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/overview.html>. 3.1, 3.2, 3.3, 3.4, 3.5.1, 3.5.2, 3.5.3, 3.5.4, 3.5.5

[4] robotstxt.org: *About /robotsrobots.txt [online]*, robotstxt.org, 01.04.2008, <http://www.robotstxt.org/robotstxt.html>. 3.5.1

[5] Internet Archive: *Some notes on the URI classes [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/uri.html>. 3.6, 3.6.1, 3.6.2

[6] Internet Archive: *Common needs for all configurable modules [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/chap_modules_common.html>. 3.7, 3.7.1, 3.7.1.1, 3.7.1.2, 3.7.2, 3.7.2.1

[7] Internet Archive: *Writing FrontierFrontier [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/frontier.html>. 3.8

[8] Internet Archive: *Writing Filter [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/writefilter.html>. 3.9

[9] Internet Archive: *Writing Scope [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/scope.html>. 3.10

[10] Internet Archive: *Writing Processor [online]*, Internet Archive, 26.03.2008, <http://crawler.archive.org/articles/developer_manual/processor.html>. 2.3, 3.11, 3.11.1, 3.11.3

[11] Lukáš Matejka: *Accessing the Czech Web Archive [online]*, WebArchiv, 09.04.2008, <http://en.webarchiv.cz/files/dokumenty/ostatni/thesis.pdf>.

[12] VERIO: *Whois.Net [online]*, Whois.Net, 18.5.2008, <http://www.whois.net/>. 1

[13] Gomes: *A Characterization of Portuguese Web [online]*, IWAW, 18.5.2008, <http://pandora.nla.gov.au/documents/domain_harvest_report_public.pdf>. 1

[14] Koerbin: *Report on the Crawl and Harvest of the Whole Australian Web Domain Undertaken during June and July 2005 [online]*, National Library of Australia, 10.10.2005, <http://bibnum.bnf.fr/ecdl/2003/proceedings.php?f=gomes>. 1

[15] Grethe Jacobsen: *Harvesting the Danish internet – the first two years [online]*, netarkivet.dk, 2.5.2007, <http://netarkivet.dk/publikationer/CollectingTheDanishInternet_2007.pdf>. 1

[16] Charalampos Lampos: *Archiving the Greek Web [online]*, Athens University of Economics and Business Department of Informatics, 18.5.2008, <http://www.iwaw.net/04/Lampos.pdf>. 1

Index