Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Import dat ze služby ACM Digital Library do formátu XML

Plzeň, 2012 Jan Krupička

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 11. 5. 2012

Jan Krupička

Abstract

Import of data from ACM Digital Library into XML

The Association for Computing Machinery is the world's largest and most prestigious scientific and educational computing society. One of its services is a digital library ACM Digital Library which is available online via search form at *dl.acm.org*. ACM DL is the most comprehensive collection of full-text articles and bibligraphic records in existence today covering the fields of computing and information technology.

The aim of this work is to create an application with a grafical user interface (GUI), which allows the user to search this library. Then the application will be able to obtain desired information and save it into a designed XML format. This information will contain inter alia an unique identifier of found publication and a list of its citing publications so that these dependencies will form a graph.

Import dat ze služby ACM Digital Library do formátu XML

Společnost ACM (Association for Computing Machinery) je největší a nejvíce prestižní vědeckou a vzdělávací společností v oblasti výpočetní techniky. Jednou z jejích služeb je digitální knihovna ACM Digital Library, která je dostupná online pomocí vyhledávacího formuláře na webové adrese *dl.acm.org*. ACM DL je nejrozsáhlejší sbírkou full-textových článků a bibliografických záznamů pokrývajících oblasti výpočetní techniky a informačních technologií.

Cílem této práce je vytvořit aplikaci s grafickým uživatelským rozhraním (GUI), která uživateli poskytne možnost v této knihovně vyhledávat. Z nalezených výsledků poté dokáže získat požadované informace a ty pak uložit do navrženého formátu XML. Tyto informace budou mimo jiné obsahovat jednoznačný identifikátor nalezené publikace a seznam jejích citujících publikací tak, aby tyto závislosti utvářely graf.

Obsah

1 Úvod

Důležitou činností soudobého člověka je vědecký výzkum. Dnešní věda produkuje velké množství publikací, jako jsou např. články, časopisy nebo sborníky z konferencí a to ze všech možných odvětví. K rychlejšímu rozvoji je potřeba uchovávání těchto výsledků, možnost jejich snadného vyhledávání a poskytování zájemcům. Pro tento účel existuje mnoho bibliografických služeb, jako je např. SciVerse Scopus, Google Scholar nebo právě ACM Digital Library, které platícím uživatelům poskytují databáze plné informací o publikacích a jejich autorech.

Společnost ACM byla založena roku 1947 v New Yorku a dnes pomocí služby ACM DL poskytuje nejobsáhlejší kolekci full-textových článků a bibliografických záznamů, které pokrývají oblast výpočetní techniky a informačních technologií. Tato služba je přístupná na webové adrese [7].

Cílem této práce je seznámit se s prostředím služby a poté analyzovat, navrhnout a implementovat aplikaci s grafickým uživatelským rozhraním (GUI), která uživateli poskytne možnost v této knihovně vyhledávat. Z nalezených výsledků poté dokáže získat požadované informace a ty pak uložit do navrženého formátu XML.

Nejdůležitější ukládanou informací bude jednoznačný identifikátor nalezené publikace a seznam jejích citujících publikací. Tyto informace budou základním kamenem pro navazující práci, jejímž cílem bude vytváření citačních sítí a pro kterou tato bakalářská práce připravuje data.

Důležitou vlastností vytvořené aplikace bude implementovaná nastavitelná prodleva mezi dvěma následujícími připojeními k internetu. Důležitá je proto, aby nedocházelo k nadměrnému zatížení serveru služby ACM DL na úkor ostatních uživatelů.

2 Služba ACM DL

Služba má hlavní stránku na adrese [7]. Její horní část je vidět na obr. A.1. Obsahuje různé informace jako oznámení nebo nedávno přidaná vydání. Číslem 1 je na obrázku zvýrazněn základní vyhledávací formulář. Ten však nenabízí žádné možnosti pokročilého vyhledávání.

K tomu je určen odkaz "Advanced Search" [8], který je na obrázku zvýrazněn číslem 2. Odkaz vede na stránku s již značně rozsáhlým formulářem pro podrobnou specifikaci vyhledávacího dotazu.

2.1 Pokročilé vyhledávání

Formulář pokročilého vyhledávání je zobrazen na obr. A.2. Jak je vidět, obsahuje spoustu kriterií (od slov v abstraktu, jmen autorů, jejich afilace, vydavatele, roku vydání, přes název konference až po identifikační čísla), pomocí kterých lze dotaz zkonkretizovat.

Většina kriterií má navíc možnost volby boolovského spojení při víceslovném vyhledávání. Pokud chceme vyhledat frázi nebo např. celé jméno, musíme je uzavřít do uvozovek. Po vyplnění formuláře požadovanými informacemi dotaz odešleme tradičně stiskem tlačítka ENTER nebo Search. Dostaneme se tak na stránku s výsledky.

2.2 Výsledky vyhledávání

Zde je několik důležitých prvků, kterých je pro tuto práci vhodné si povšimnout. Všechny jsou vyznačeny na obr. A.3.

2.2.1 Přepínání mezi ACM a GUIDE

V základu se informace z formuláře hledají v knihovně ACM DL. Odkaz zvýrazněný číslem 1 rozšíří výsledky i na knihovnu GUIDE. Ta obsahuje i další záznamy, jejichž informace jsou však neúplné. Jsou jimi především jen citace a abstrakty.

2.2.2 Řazení výsledků

Kombo box označený číslem 2 slouží k nastavení řazení výsledků, podle různých kriterií. Řadit lze např. podle názvu, relevance výsledků nebo počtu citací. V této práci ponechávám řazení podle relevance a nedávám možnost toto programově měnit, nicméně je dobré o této možnosti vědět.

2.2.3 Způsob zobrazení výsledků

Číslem 3 je označen užitečný kombo box, který má dvě volby. Jednou je volba "expanded form", díky které se na jednu stránku výsledků zobrazí dvacet záznamů s rozšířenými informacemi. Druhou volbou je "condensed form", která na stránku s výsledky pošle padesát záznamů s pár základními údaji.

Druhá volba tedy aplikaci umožní snížit počet připojení k serveru při zpracovávání výsledků na začátku procesu. Stránka s výsledky s tímto nastavením je zobrazena na obr. A.4.

2.2.4 Vyhledané záznamy

Posledním zajímavým prvkem jsou samotné jednotlivé výsledky. Každý vyhledaný záznam má svůj název, který je zároveň také odkazem na další webovou stránku s detailem tohoto záznamu.

2.3 Detail publikace

Detail publikace obsahuje množství více či méně důležitých informací a je zobrazen na obr. A.5. Pokud je dostupný plný text publikace, odkaz na něj je hned pod jejím názvem. Všechny prvky důležité pro tuto práci jsou opět na obrázku zvýrazněné.

2.3.1 Autoři

Seznam autorů je umístěn v poli s hlavními informacemi a na obrázku je označen číslem 1. Každý autor může mít u svého jména uvedenu svou afilaci, pokud nějakou v době vydání publikace měl.

Tyto informace mohou být a většinou také jsou odkazem na stránku s profilem autora, resp. instituce. Lokální umístění obou informací ve zdrojovém textu stránky (tedy v tom, který bude zpracováván) závisí právě na tom, zda jsou dostupné odkazy na stránky s detaily.

2.3.2 Formáty pro export

Dalším důležitým prvkem je nabídka exportu informací, na obrázku je zvýrazněna číslem 2. ACM umožňuje informace exportovat ve třech formátech. Jsou jimi BibTeX, EndNote a ACM Ref.

Každý z nich má jinou strukturu a obsahuje různé množství různých informací. Neobsahují tedy vše stejné, ale stejná je jen určitá podmnožina. Nejvíce informací je obsaženo v BibTeXu (obr. A.10). Je tedy nejlepším kandidátem na zdroj informací.

2.3.3 Přepínač stylu stránky

Toto nastavení se týká oblasti pod hlavními informacemi. V základním stavu je styl stránky moderní s lištou z interaktivních tlačítek, která pomocí technologie AJAX (Asynchronous JavaScript and XML) naplňují informacemi spodní část stránky.

Po přepnutí stylu zmíněným přepínačem, na obrázku je označen číslem 3, na styl jednoduché stránky se všechny informace vypíší sériově pod sebe. Tento styl (viz obr. A.6) je tudíž vhodnější pro strojové zpracování než simulace AJAXu a znamená také další snížení počtu připojení k serveru.

2.3.4 Informace o citacích

V zadání požadované logické spojení publikací pomocí informací o citacích je možné provést dvěma směry. Lze to provést jednak využitím citací, které obsahuje daná publikace (na stránce to je oddíl *REFERENCES*), nebo pomocí informací o publikacích, které citují naší danou publikaci (oddíl *CITED BY*), označeno číslem 4.

Oddíl *REFERENCES* je zobrazen na obr. A.7. Jak je z obrázku vidět, ne všechny záznamy jsou zároveň odkazy na detail publikace, kterou představují. Naproti tomu oddíl *CITED BY* (viz obr. A.8) obsahuje vždy jen záznamy, které jsou zároveň odkazy na detail publikace.

3 Technologie

V této části budou popsány znalosti a technologie, které jsou potřebné pro úspěšné vypracování této práce a splnění jejího zadání.

Jelikož je hlavním objektem zájmu služba ACM DL, která je běžně dostupná pouze přes webové rozhranní, budu se věnovat stručnému popisu příslušných webových technologií a možnostem získání odpovědi serveru z internetu. Dále také výběrem vhodného programovacího jazyka k implementaci aplikace, možnostem zpracování získaných informací a způsoby jejich ukládání do souboru.

3.1 Webové technologie

3.1.1 HTML

HyperText Markup Language [1][10] je značkovací jazyk vyvinutý pro snadnou tvorbu webových dokumentů a se snadným a velmi užitečným odkazováním na jiné zdroje v internetu. HTML je aplikací univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language), který umožňuje definovat jiné značkovací jazyky coby své podmnožiny.

Dnes existuje již ve verzi 5 a je velice oblíbený pro svou jednoduchost a, v kombinaci s moderními prohlížeči, pro svou vlastnost odpouštět autorům webových stránek různé chyby ve své syntaxi. Toto odpouštění je sice přívětivé pro tvůrce stránek, na druhou stranu to však znamená složitější zpracovávání takovýchto dokumentů.

I když jazyk obsahuje prostředky pro základní vizuální popis dokumentu, v dnešní praxi se jazyk používá pro ryze sémantický popis dokumentu. Pro vizuální popis slouží kaskádové styly (CSS), které poskytují velmi rozsáhlé možnosti stylizace. Jelikož se však v této práci nejedná o získávání vizuálních informací, nebudu se dále CSS zabývat.

3.1.2 XML

Extensible Markup Language [2] je obecný značkovací jazyk. Je to zjednodušená verze jazyka SGML. XML je používán zejména pro serializaci dat a jejich výměnu mezi různými aplikacemi (např. databáze). K popisu jeho konkrétního formátu slouží např. soubory DTD nebo XSD, které se používají k validaci dokumentu.

Tento jazyk je podporován celou řadou aplikací a programovacích jazyků. I z tohoto důvodu bude použit pro ukládání informací získaných ze služby ACM DL. K tomu bude potřeba navrhnout strukturu a názvy pro prvky ve výsledném dokumentu.

3.1.3 XHTML

Extensible HyperText Markup Language [1] je značkovací jazyk, který byl vyvinut tak, aby vyhovoval striktní syntaxi XML, ale zároveň zůstal kompatibilní s HTML. To s sebou nese větší požadavky na autory webových stránek, ale také následné snadnější zpracování.

K webovým stránkám ACM je připojen validační DTD soubor s popisem tohoto jazyka. I přes to je v hlavičce stránek jako jejich obsah uvedeno HTML ("text/html"). Jeden z možných důvodů bude uveden níže v analýze zpracování informací (úsek 4.3).

3.1.4 HTTP

HyperText Transfer Protocol je protokol pro přenos hypertextových dokumentů v internetu. Komunikace pomocí protokolu probíhá způsobem dotaz-odpověď. Protokol je to bezestavový, tzn. neudržuje žádný stav komunikace a každý nový dotaz je nezávislý na předchozím. K udržení informace o komunikaci klienta se serverem (stavu komunikace) se v hlavičce protokolu posílají tzv. cookies (viz dále).

3.1.5 Cookies

Cookies jsou malá data, která se posílají v hlavičce protokolu HTTP, a slouží pro udržení informace o stavu komunikace klienta se serverem. Vytváří je server, ale uchovávají se na straně klienta, který je při dalším dotazu posílá zpátky serveru. Cookies obsahují informaci o svém názvu, hodnotě, datu vypršení platnosti a jméně domény, pro kterou platí.

3.1.6 URL

Uniform Resource Locator [15] je řetězec znaků, který slouží ke specifikaci umístění různých zdrojů či služeb v internetu. URL má strukturu rozčleněnou na několik částí. První z nich je schéma, na němž závisí i formát zbytku URL. Jako schéma pro internetové zdroje, používané v této práci, bude vždy "http". To znamená, že zdroj je webová stránka přebíraná pomocí protokolu HTTP.

Další části jsou následující:

<host>:<port>/<path>?<search part> ,

kde "host" je plně kvalifikované jméno domény, "port" je číslo portu, na kterém je zdroj či služba dostupná, "path" je relativní cesta ze vstupního adresáře domény k požadovanému zdroji či službě a konečně "search part" je část vyhrazená pro vyhledávací kontext. Ta je složená ze znakem '=' oddělených dvojic, které jsou samy odděleny znakem '&'.

3.1.7 OpenURL

OpenURL je standardizovaný formát URL, který má uživatelům umožnit snazší nalezení zdroje v internetu. OpenURL je používáno především digitálními knihovnami, protože tento standard umožňuje odkazovat se na zdroje (např. údaje v databázích) přes služby knihovny.

Služba knihovny může být vyhledávání v databázi. Na serveru pracuje tzv. link resolver, který zpracuje jednotlivé části OpenURL a na základě tohoto zpracování dokáže uživateli zprostředkovat požadované informace.

3.1.8 BibTeX

BibTeX [14] je nástroj, který je určen pro formátování bibliografických informací (ty se nacházejí v souboru s určitým formátem) pro systém LaTeX. Jeho formát je ukázán na obr. A.10.

3.2 Programovací jazyky

Zde se nachází výčet několika programovacích jazyků, které připadají v úvahu při tvorbě zadané aplikace.

3.2.1 C/C++

Tyto nyní již starší jazyky [5][6] jsou i dnes velmi používané a to zejména pro rychlost výsledného programu. Jsou to nízkoúrovňové, kompilované jazyky, kdy se zejména C používá pro psaní ovladačů zařízení nebo operačních systémů. Jsou často využívány také pro psaní interpreterů vysokoúrovňových jazyků (např. Java a C#). Zdrojové kódy dodržující standard jsou přenositelné na platformy, pro které existuje překladač.

Z důvodu omezené standardní knihovny je nutné si veškerou vyšší funkčnost naprogramovat svépomocí nebo použít externí knihovnu, pokud ovšem existuje.

3.2.2 C#

Je to moderní vysokoúrovňový programovací jazyk firmy Microsoft. Podporuje objektově orientované programování a je interpretovaný platformou .NET. Tato platforma je dostupná především na operačních systémech Microsoft Windows, ale objevují se i implementace standardizovaných částí platformy i na další systémy.

Výhodou tohoto jazyka [16] je jeho robustnost, automatická správa paměti a také rozsáhlá standardní knihovna. I proto získává stále více na popularitě.

3.2.3 Java

Java [4][9] je jeden z nejpopulárnějších programovacích jazyků dneška [13], což nejspíše ještě více upevní oblíbená mobilní platforma Google Android. Java je moderní vysokoúrovňový interpretovaný jazyk původem od firmy Sun Microsystems, dnes vyvíjený společností Oracle. Mezi její vlastnosti patří mj. jednoduchost, robustnost, automatická správa paměti, bezpečnost a objektově orientovaný přístup.

Java je přenositelná na každou platformu, která má implementovaný interpreter JVM (Java Virtual Machine). Dále obsahuje rozsáhlou standardní knihovnu,

která programátorovi dovoluje zaměřit se na funkčnost aplikace, místo programování infrastruktury, která tuto funkčnost umožní.

3.3 Způsoby získání odpovědi serveru

Aby bylo možné pracovat s daty ze služby ACM DL, musí být nejprve lokálně k dispozici. Aplikace tedy musí být schopna získat odpověď serveru (HTML stránku) pomocí HTTP protokolu (tzn. komunikace typu dotaz-odpověď).

3.3.1 Prostředky jazyka

Nejlepší možností je využití prostředků zvoleného programovacího jazyka, pokud tuto funkčnost poskytuje. To připadá v úvahu u moderních jazyků s rozsáhlou standardní knihovnou.

U jazyka C# tuto funkčnost zajišťují třídy WebRequest a WebResponse, resp. HttpWebRequest a HttpWebResponse. U jazyka Java je to třída URL, resp. HttpURLConnection.

3.3.2 Vlastní implementace HTTP komunikace

Tato možnost zbývá u nízkoúrovňových jazyků, které tuto funkčnost nemají implementovanou ve standardní knihovně, nebo není k nalezení vyhovující externí knihovna.

3.4 Zpracování získaných informací

V této části popíši technologie, jež souvisí se zpracováním informací ve formě XHTML/XML souboru, který budeme mít k dispozici po odeslání dotazu a přijetí odpovědi serveru.

3.4.1 XSLT transformace

XSLT transformace [12] je technologie sloužící pro dotazování nad a transformaci XML dat. Samotná transformace je popsána v XML souboru zvaném "stylesheet". Ten definuje, která data ze zdrojového XML (toho, co bude transformováno) se na kterých místech a jakým způsobem použijí ve výstupním XML souboru.

Pro výběr dat se používá dotazovací jazyk XPath. Výsledkem dotazu v tomto jazyce je ve většině případů množina uzlů, se kterými se dále pracuje. Nad daty je možné provádět jednoduché výpočty nebo jiné operace, pro které je dostupné více než 150 funkcí.

Transformace jsou prováděny pomocí XSLT procesorů. Příkladem může být Saxon [17] nebo Xalan [18]. XSLT procesory jsou také integrované v internetových prohlížečích a jsou dostupné i jako knihovny programovacích jazyků (např. Xalan je součástí Javy od verze JDK 1.4)

3.4.2 Objektové stromy (DOM)

Pro manipulaci s dokumentem lze nejprve vytvořit stromovou strukturu charakterizující XML dokument v paměti a následně pak z této struktury vybírat potřebná data a dále s nimi pracovat.

3.4.3 Proudové parsery

Proudové parsery zpracovávají vstupní soubor postupně (v proudu). To znamená, že je potřeba méně paměti a lze zpracovávat i velmi rozsáhlé soubory, nebo soubory, které nejsou v daném čase dostupné v kompletní podobě.

Existují dva druhy proudových parserů. Tzv. push parsery zpracovávají soubor a generují při tom různé události. Události se generují např. při rozpoznání otevíracího tagu elementu, nového atributu atd. Druhým typem jsou tzv. pull parsery. Ty negenerují události, ale další prvky XML souboru vrací na vyžádání.

3.4.4 Předzpracování cleanerem

Jak bylo řečeno v části o HTML jazyku, moderní prohlížeče a samotný HTML jazyk odpouští mnoho syntaktických prohřešků. Tato skutečnost má za následek výskyt spousty nevalidních stránek, které je o to těžší následně zpracovat.

Naštěstí existují volně dostupné nástroje, které tento problém řeší předzpracováním zdrojových souborů webových stránek a opravou těchto syntaktických chyb na úroveň validního HTML nebo spíše až validního XHTML tudíž v podstatě XML. Těmto nástrojům se říká HTML cleanery.

3.5 Způsoby ukládání dat

Způsob ukládání dat do jisté míry záleží na tom, v jaké podobě máme data v aplikaci reprezentovaná.

3.5.1 Objektový strom (JAXB)

Pokud máme data reprezentovaná stromovou strukturou objektů, pro jejich uložení se v Javě nabízí technologie JAXB [3]. Ta využívá faktu, že je struktura výstupního XML souboru popsána externím souborem XSD. Z tohoto souboru se automaticky vygenerují třídy jazyka Java, které lze naplnit daty a následně nechat celou stromovou strukturu jednorázově vypsat do souboru.

3.5.2 Proudový zápis (StAX)

Proudový zápis se hodí především při proudovém zpracovávání vstupního souboru. Data se zapisují postupně, např. v době, kdy je již nějaký datový celek připraven k výpisu. Tento způsob [3] opět umožňuje použití menšího množství paměti.

4 Analýza

Zde uvádím porovnání a výběr z technologií a přístupů, které jsou popsány v teoretické části výše.

4.1 Programovací jazyk

Zadání nepřikládá žádné ultimátní požadavky na rychlost požadované aplikace a tudíž výkonným kompilovaným jazykům C a C++ odpadá jejich (v tomto případě si troufám říci i jediná) výhoda, kterou je právě ona rychlost.

Oba zbylé interpretované jazyky mají pro tento druh aplikace výkonnost více než dostatečnou. Nespornou velkou výhodou je u obou rivalů rozsáhlá standardní knihovna, která nabízí infrastrukturu potřebnou pro splnění cílů této práce již hotovou k dispozici.

Z mého pohledu jsou to pro tyto účely jazyky zcela rovnocenné a tak přicházejí na řadu osobní preference. S jazykem C# jsem se za svou stále velmi krátkou programátorskou kariéru setkal jen okrajově, tudíž z tohoto důvodu padá jednoznačná volba na jazyk Java. Ten je mým jazykem mateřským a používám jej po celé studium na vysoké škole.

4.2 Získání odpovědi serveru

Nejlepší možností by bylo využít programového API pro přímou komunikaci se službou ACM DL, nicméně žádné takové API bohužel neexistuje. Díky zvolení jazyka Java odpadá nutnost zařídit si komunikaci pomocí HTTP protokolu svépomocí.

Volba je tedy jasná. Pomocí možností jazyka Java odeslat dotaz na server a následně stáhnout odpověď v podobě webové stránky.

4.2.1 Povaha dotazu

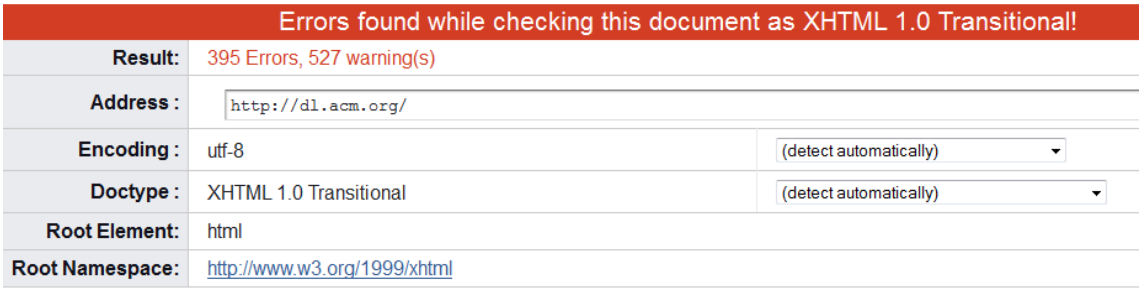
V internetovém prohlížeči se po vyplnění formuláře a stisknutí tlačítka Search dotaz odešle serveru. Díky tomu, že ACM používá standard OpenURL, nejsou data z formuláře odesílána metodou POST protokolu HTTP, ale jsou předána pomocí URL v podobě dvojic *název=hodnota*. Těchto dvojic se v dotazu ACM vyskytuje více než 40.

Tato povaha výrazně zjednoduší proces dotazování, jelikož se HTTP požadavek v Javě provádí pomocí již zmíněné třídy URL. Stačí tedy správně sestavit URL z vyhledávaných klíčových slov a vzápětí lze server požádat o vyhledání záznamů.

4.3 Zpracování informací

Získání dat z jednoho XML a jejich uložení do jiného XML je doména XSLT transformací. K tomu, aby bylo možné tuto transformaci využít je však potřeba validního XML nebo alespoň XHTML.

Pro prozkoumání této možnosti jsem pomocí validátoru W3C [11] nechal zkontrolovat úvodní stránku služby ACML DL. Výsledek dopadl celkem tragicky, jak je vidět na obrázku 4.1. Na této krátké webové stránce se našlo okolo čtyř set chyb a pěti set varování a to i přes to, že bylo použito schéma XHTML Transitional. Pro zajímavost uvedu, že tato čísla jsou možná vysoká, ale ještě tři měsíce před psaním tohoto textu byla dvojnásobná.



Obr. 4.1: Výsledek validace úvodní stránky služby ACM DL.

Je tedy jasné, že bude nevyhnutelné použít některý z volně dostupných HTML čističů [19]. Jedním z nich je HTML Parser [20], který si dokáže poradit s nevalidním HTML a rovnou webovou stránku naparsuje do stromové struktury podobné specifikaci DOM. To je také pro další práci zapotřebí, navíc je tato knihovna dobře otestovaná, při své činnosti rychlá a podporuje filtrování uzlů podle různých vlastností.

Proudové parsery nejsou pro tuto aplikaci příliš vhodné, protože zdrojový kód stránky je strojově generovaný a obsahuje jen velmi málo "záchytných bodů", tzn. uzlů pojmenovaných identifikátory nebo třídami či mající speciální tvar.

Výběr je tedy učiněn. Použije se stromová reprezentace dokumentu, ve které se budou hledat požadované informace.

4.4 Způsob ukládání dat

Podle výběru vnitřní reprezentace dat pomocí stromové struktury, by se mohlo zdát dobrou volbou způsobu ukládání informací bude technologie JAXB. Avšak stromová struktura držená v paměti v průběhu zpracování jedné publikace není ta samá jako struktura, která by byla vypisována pomocí JAXB. Ta totiž charakterizuje reprezentaci XML stromu výsledného souboru a obsahuje tak informace o všech zpracovaných publikacích. V průběhu času by se tedy s dalšími zpracovávanými publikacemi rychle zvětšovaly nároky na paměť, což není vůbec dobré.

Optimální bude použít proudového způsobu ukládání a tedy technologii StAX (také známou jako SJSXP neboli Sun Java Streaming XML Parser). Výhodou je, že se neukládá celý strom jednorázově a to po každé změně (nebo množině změn) znovu. Místo toho se data ukládají postupně a v době, kdy jsou poslány na výstup, se již nacházejí v souboru na disku a jsou tak chráněna před ztrátou např. kvůli neočekávanému pádu aplikace, nebo hostitelského počítače.

5 Návrh

Cílem praktické části této práce je naprogramovat aplikaci, která dokáže učinit dotaz na službu ACM DL a následně stáhnout její odpověď na něj. Dále musí umět tyto výsledky zpracovat a extrahovat požadovaná data, která budou ukládána do navrženého XML souboru. Aplikace musí obsahovat požadované vlastnosti uvedené níže.

5.1 Požadované vlastnosti

Následuje stručný popis vlastností, které jsou požadovány zadavatelem a které se týkají jednak obsahu výstupního XML a jednak funkčnosti samotné aplikace.

5.1.1 Požadavky na výstupní XML

Nejdůležitější informací je identifikátor nalezené publikace, pomocí něhož lze vytvořit graf citačních závislostí. To znamená, že tento identifikátor musí být stejného typu pro každý nalezený záznam a zároveň musí být u každého záznamu přítomen.

Ze všech nalezených identifikátorů tyto dvě vlastnosti splňuje pouze jeden a tím je interní ACM ID. Toto ID lze nalézt v odkazu na detail každé publikace (obecně entity, tj. např. i autor). ACM ID se u publikací skládá ze dvou částí oddělených tečkou. První je ID nadřazené publikace a až druhá část je ID samotné publikace. Pokud nadřazená publikace neexistuje (např. u sborníku z konference), první část včetně tečky se neuvádí. Také v druhém případě platí, že první část je pouze volitelná, neboť lze snadno ověřit, že obě podoby ID vedou ke stejné publikaci.

Samozřejmostí je uvedení názvu nalezené publikace. Tato informace se nachází na více místech, např. na stránce s výsledky, v detailu publikace nebo v sekci s citujícími publikacemi, přičemž na některých těchto místech se nachází název zkrácený. Do XML jsem pro jeho snadné získání vybral název z BibTeXu.

Další, pro tvoření grafu citačních závislostí důležitou, informací je seznam identifikátorů publikací, které citují aktuálně nalezenou publikaci.

Následuje seznam autorů, kteří se podíleli na tvorbě nalezené publikace. Položka v tomto seznamu obsahuje kromě jména autora také jeho ID a afilaci.

V XML je také požadována sada informací týkajících se vydání publikace. Je jimi svazek, číslo periodika, rok vydání, rozsah stránek, vydavatel nebo název nadřazené publikace. Pokud některá z těchto informací není dostupná (např. název nadřazené publikace pro sborník z konference), je tento atribut vyplněn hodnotou "none".

Posledním požadovaným atributem je číslo generace ve vyhledávacím procesu, ve které byla publikace nalezena.

5.1.2 Funkčnost aplikace

Nejdůležitější funkcionalitou aplikace je implementace nastavitelné prodlevy mezi připojeními k serveru. Jinými slovy je tato prodleva garantovaný minimální čas od posledního připojení k serveru ACM DL, po který nebudou vykonávána jakákoli další připojení.

Tato prodleva je nutná pro to, aby nedocházelo k nepřiměřenému zatížení serveru ACM DL a tedy aby si program nenárokoval velkou část výpočetní kapacity serveru na úkor ostatních uživatelů služby.

Další důležitou dovedností programu je schopnost nalezená a zpracovaná data ukládat průběžně. To je nutné z toho důvodu, aby se při vyskytnutí nějakého problému (např. pád aplikace, odpojení elektřiny atd.) neztratil veškerý dosavadní postup, tj. data dosud nalezených publikací.

Jedním z dalších požadavků je také dvojice nastavení, které se týkají zpracovávaných generací. Prvním požadovaným nastavením je šířka první (tzn. nulté) generace. Tento údaj charakterizuje počet prvků ve výchozí generaci. Z těch se poté při vyhledávání pokračuje na jejich citující publikace.

Druhým nastavením je číslo poslední generace, která se má zpracovat. Aplikace tedy bude zpracovávat generace 0 až toto číslo včetně.

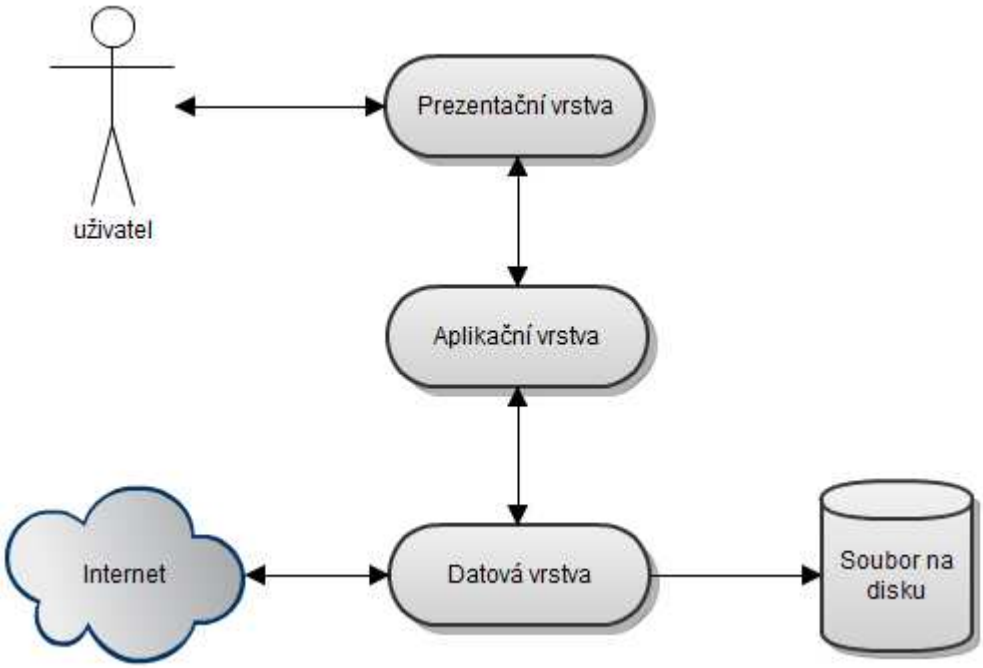
Již méně esenciálním požadavkem je zapracování stavového řádku do grafického rozhranní aplikace. Tento řádek zobrazuje stav, ve kterém se aplikace aktuálně nachází. V tomto případě bude zobrazovat jednoduchou průběžnou statistiku.

Souvisejícím grafickým prvkem a zároveň dalším požadavkem je tzv. progres bar. Ten v případě provádění nějaké činnosti zobrazuje, jaká část práce byla touto činností vykonána. V případě nedeterministických činností (jako je tato) jen jednoduše indikuje, že se nějaká činnost provádí.

Posledním požadavkem je, aby aplikace nabídla možnost volby umístění výstupního XML souboru s výsledky.

5.2 Architektura

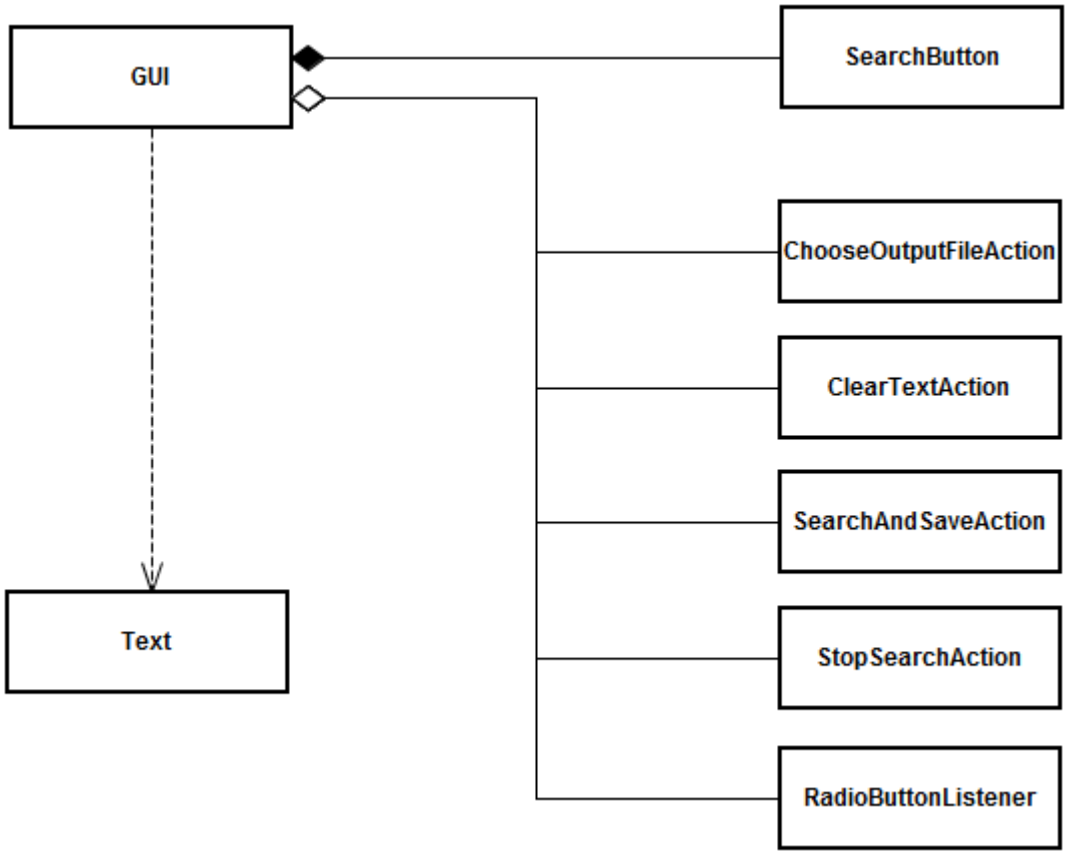
Aplikace bude rozdělena na tři části a to podle klasického třívrstvého návrhu architektury. Jsou jimi prezentační vrstva, aplikační vrstva a vrstva datová, které jsou popsány dále. Obecná struktura a základní interakce jsou znázorněny grafem na následujícím obr. 5.1.



Obr. 5.1: Vrstvená architektura aplikace.

5.2.1 Prezentační vrstva

Prezentační vrstva má za úkol zobrazovat uživateli informace o stavu aplikace a činnostech, které aplikace vykonává, a naopak přebírat informace od uživatele ke zpracování. Struktura vrstvy je znázorněna diagramem tříd na obr. 5.2.



Obr. 5.2: Zjednodušený diagram tříd prezentační vrstvy.

V aplikaci bude tato vrstva implementována jako grafické uživatelské rozhranní. Výhodou vybraného programovacího jazyka je, že tvorba grafického rozhranní je v jeho standardní knihovně podporována na vysoké úrovni, což nechává široké možnosti jeho tvorby a velmi usnadní to práci. Všechny třídy, které spadají do této vrstvy, se budou nacházet v balíku s názvem "gui".

Nejdůležitější částí GUI bude vstupní vyhledávací formulář. Jeho podoba bude mít předlohu ve formuláři dostupném na stránkách pokročilého vyhledávání služby ACM DL [8]. Jednotlivé prvky budou pojmenovány stejně a rozmístěny na stejných místech jako ve své předloze.

Pod formulářem se bude vyskytovat panel s nastavením vyhledávání a jeho ovládáním. Bude tedy obsahovat tři textová pole pro nastavení generací a prodlevy mezi připojeními, dále přepínač mezi knihovnou DL a GUIDE a tlačítko pro výběr názvu a umístění výstupního souboru. Posledním prvkem na tomto panelu bude tlačítko, které umožní spustit nebo zastavit vyhledávání.

Prostor pod právě popsaným panelem bude vyplňovat textová oblast, do které se budou v průběhu vyhledávání vypisovat URL adresy, ze kterých se aktuálně získávají informace, a také jednoduchá statistika poté, co vyhledávání skončí.

A konečně bude grafické rozhraní vespod uzavírat stavový řádek spolu s progres barem.

Hlavní třída s GUI bude dědit od třídy `JFrame` a bude tak charakterizovat okno aplikace. Při vytvoření instance třídy se pomocí početných metod sestaví výše popsané rozhranní. Dále bude třída poskytovat metody pro změnu indikace stavu celé aplikace. Bude tedy nabízet metody pro vypsání textu do textové oblasti, změnu textu ve status baru a metody pro práci s tlačítky, která zahrnuje jejich aktivaci/deaktivaci.

Asi nejobsáhlejší bude metoda pro sestavení vyhledávacího dotazu. Ze zhruba čtyřiceti prvků formuláře bude sestavovat vyhledávací část URL.

Dále bude vrstva obsahovat třídy reprezentující případné grafické prvky a další třídy implementující jejich akce a posluchače. Poslední třídou bude třída Text, která v sobě bude obsahovat všechny texty z celé aplikace. To bude uděláno proto, aby se texty snáze měnily a také aby byla ulehčena případná pozdější internacionalizace.

5.2.2 Aplikační vrstva

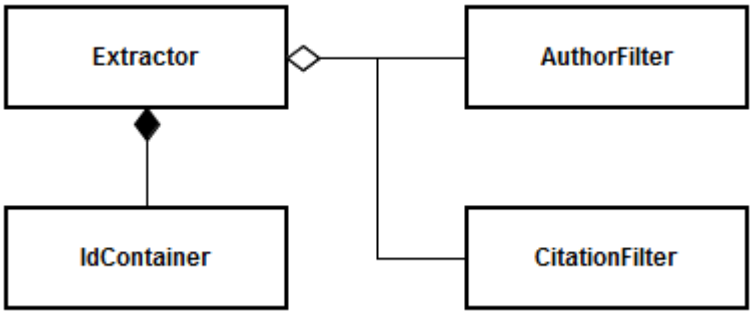
Úkolem aplikační vrstvy je převzít od datové vrstvy přijaté výsledky vyhledávání ze služby ACM DL, tato data zpracovat a poslat je opět datové vrstvě k uložení.

Do této vrstvy bude patřit jednak balík s názvem `program`, ve kterém se budou nacházet všechny třídy s výpočetní činností, ale také sem bude patřit celá externí knihovna *HTML Parser 2.0*. Struktura vrstvy je zachycena grafem na obr. 5.3.

HTML Parser se bude starat o předzpracování výsledků služby ACM DL, do kterého patří především dvě činnosti. Jednou je vyčištění zdrojových souborů,

tzn. nápravu chyb v syntaxi jazyka HTML. Druhou činností je vytvořit v paměti objektový strom, který charakterizuje dokument se staženými výsledky a dovoluje s nimi dále pracovat.

V balíku `program` se budou nacházet třídy, které se podílejí na zpracování zmíněného objektového stromu a extrakci požadovaných informací určených k uložení. V podbalíku `filters` budou k dispozici třídy pro filtrování množin uzlů, určené interním metodám knihovny *HTML Parser*.



Obr. 5.3: Zjednodušený diagram tříd aplikační vrstvy.

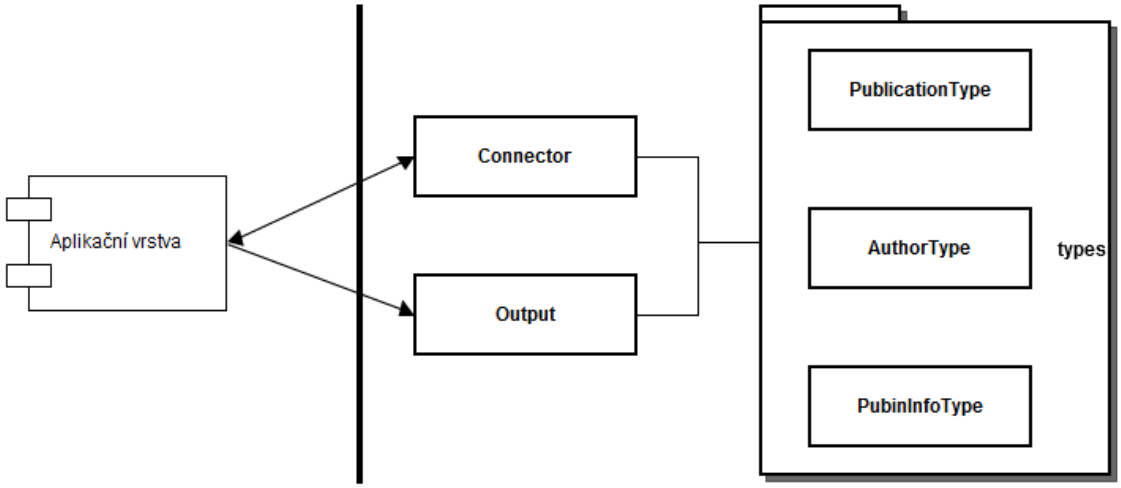
Nejdůležitější třídou bude třída `Extractor`, která bude řídit celý proces vyhledávání a zpracování informací. Bude obsahovat dva hlavní cykly. V prvním vyhledá identifikátory publikací z nulté generace. Tyto identifikátory bude ukládat do třídy `IdContainer`, která bude zároveň kontrolovat, zda se některá z publikací nezpracovává vícekrát (to by nastalo, např. kdyby dvě publikace byly citovány stejnou třetí).

V druhém cyklu se budou všechna nalezená ID zpracovávat. Budou se stahovat detaily publikací s těmito ID pomocí datové vrstvy, zpracovávat jejich informace a nakonec se tyto informace pošlou zpět do datové vrstvy k uložení. Během zpracování seznamu citujících publikací se jejich ID uschovají k pozdějšímu zpracování v další generaci. Z toho plyne, že tento cyklus zabere velké a nepředvídatelné množství času.

5.2.3 Datová vrstva

Datová vrstva má dva úkoly. Jedním z nich je připojit se k serveru služby ACM DL a získat jeho odpověď ke zpracování. V této fázi musí datová vrstva zajistit, aby byla

dodržena určená prodleva mezi jednotlivými připojeními. Druhým z úkolů je pak data zpracovaná aplikační vrstvou uložit na disk.



Obr. 5.4: Zjednodušený diagram tříd datové vrstvy.

Patří sem dvě třídy a jeden podbalík (viz obr. 5.4). Třída `Connector` se stará o získávání dat z internetu a práci s cookies. Bude obsahovat metody pro připojení k serveru, stažení jeho odpovědi a metodu pro garanci časové prodlevy.

Třída `Output`, která se postará o onen zápis dat na disk do XML souboru, jehož přesný formát bude popsán v části implementace. Data bude ukládat postupně proudově v průběhu vyhledávání.

V této vrstvě se ještě bude nacházet podbalík `types`, který bude obsahovat typové třídy charakterizující formát zpracovaných dat (informací o publikaci) určených k výpisu. O výpis tohoto souboru informací se pak naráz postará třída `Output` samostatně.

6 Implementace

V této části budu popisovat implementační detaily. Jedná se o popis přesného formátu výstupního XML souboru a ve zbytku textu pak o popis důležitých detailů jednotlivých tříd, rozdělených podle vrstev aplikace.

6.1 Výstupní XML

Ukázka jednoho konkrétního výstupního souboru je ukázána v příloze C. Následuje obecný slovní popis, ve kterém jsou zmíněny speciální případy a různé limity.

Kořenovým elementem výstupního XML je element s názvem `publications`. Tento element nemá žádné atributy. V něm se nachází seznam o žádném až N elementech typu `publication` (oproti kořenovému bez "s" na konci).

Element typu `publication` má tři atributy. Prvním a nejdůležitějším je atribut `id`, ve kterém se nachází ACM ID příslušné publikace mající formát (viz popis výše v části 5.1.1). Dalším je atribut `gen`, který obsahuje nezáporné celé číslo odpovídající pořadovému číslu generace, ve které byla publikace zpracována. Posledním atributem je atribut `no`, jenž také obsahuje nezáporné celé číslo, které tentokrát udává pořadové číslo zpracování nalezené publikace.

Dále element `publication` obsahuje čtyři podelementy. Jejich názvy jsou `title`, `citedBy`, `authors` a element `pubinInfo`. Jejich význam stejně tak jako obsah je popsán níže.

Element `title` obsahuje titul publikace. Tím může být například název konference (u jejího sborníku), název článku (např. článku v časopisu) nebo název kapitoly v knize. Tato informace je přebírána z BibTeXu k dané publikaci. Element neobsahuje žádné parametry.

Element `citedBy` reprezentuje seznam ACM ID publikací, které citují publikaci aktuální. Jeho atribut `count` udává počet těchto identifikátorů a jeho obsahem je tedy 0 až N elementů typu `id`. Samotný element `id` obsahuje textovou podobu ACM ID ve formátu popsaném výše v úseku 5.1.1.

Element `authors` je dalším seznamem. Opět má atribut `count` udávající počet vnořených elementů, ale tentokrát jimi jsou elementy typu `author`, jejichž formát je popsán v následujícím odstavci.

Element `author` má dva atributy a žádný vnořený element. Jeho obsahem je text, který informuje o jméně jednoho z autorů aktuální publikace. Prvním ze zmíněných atributů je `id` udávající autorovo ACM ID. Tento identifikátor je pouze jedním kladným číslem a neobsahuje tedy žádnou tečku, jako tomu je v případě publikací. Druhý atribut má název `affiliation` a obsahuje afilaci (institutální příslušnost) autora v době, kdy byla publikace vydána.

Posledním elementem vnořeným do kořenového elementu každé publikace (`publication`) je element `pubinInfo`. Jeho název znamená "informace o tom, kde byla publikace vydána". Element nemá žádný obsah, ale zato má těchto šest atributů:

- `volume` - svazek nadřazené publikace, ve které byla aktuální publikace vydána,
- `issue` - číslo nadřazené publikace (např. časopisu), ve které byla aktuální publikace vydána,
- `pages` - stránky, na kterých se v nadřazené publikaci nachází text publikace aktuální,
- `year` - rok vydání aktuální publikace,
- `publisher` - vydavatel publikace,
- `title` - název nadřazené publikace.

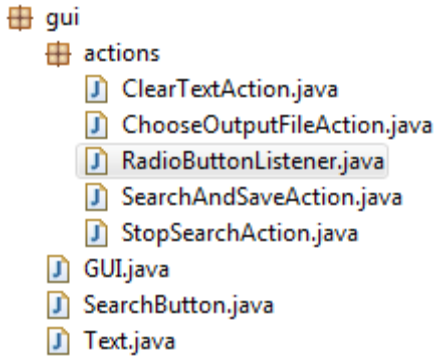
Pokud se některá z předešlých informací nevztahuje k aktuálně zpracovávané publikaci (v tom případě není uvedena), její hodnotou bude řetězec "none".

6.2 Aplikace

Zde budou popsány implementační detaily jednotlivých tříd aplikace v příslušných vrstvách. Popisovány budou především hlavní třídy `GUI`, `SearchButton` a `Text`. Třídy z podbalíku `actions` nebudou popisovány zvlášť, ale průběžně u výskytu jejich použití v třídách hlavních.

6.2.1 Prezentační vrstva

Prezentační vrstva je soustředěna do balíku `gui` a její struktura je vidět na obrázku obr. 6.1. Pohled na výsledné grafické rozhraní aplikace je na obr. A.9.



Obr. 6.1: Struktura prezentační vrstvy.

Třída `GUI`

Tato třída je zodpovědná za správné sestavení a zobrazení okna aplikace. Do jeho horní části skládá dvanáct panelů. Deset z nich napodobuje jednotlivé části formuláře služby ACM DL [8] a zbývající dva nesou prvky pro nastavení a ovládání vyhledávacího procesu. Každému z deseti panelů je přidělen rámeček s popiskem naznačujícím, čeho se týká jeho obsah, což by mělo sloužit k lepší orientaci.

Skupiny tlačítek přepínající boolovské hodnoty vyhledávání, jsou implementovány následovně. Každá skupina má několik tlačítek, každé s vlastním textovým příkazem, který odpovídá hodnotě tlačítka v dotazu služby ACM DL (a zhruba také svému názvu). Všem tlačítkům v jedné skupině je přidělen stejný posluchač typu `RadioButtonListener`, který tento příkaz při stisku tlačítka zapíše do příslušného

objektu typu `StringBuffer`. Z něho je později hodnota převzata při sestavování vyhledávacího dotazu.

V panelu s nastavením vyhledávání se nacházejí tři textová vstupní pole. Nejsou nijak formátována a tak lze zadat jakýkoli text. Jako vstup se u všech očekává nezáporné celé číslo. Z nastíněného důvodu se vstupní hodnoty musí kontrolovat, jestli obsahují validní vstup. To se děje v době stisku tlačítka *Search* a bude popsáno u příslušné třídy `SearchButton` a jejích akcí.

V panelu s ovládáním vyhledávání se vyskytuje tlačítko *Select Output*. Stisk tohoto tlačítka vyvolá dialog `FileChooser`, který je v Javě standardně dostupný a dovoluje snadnou volbu umístění i názvu výstupního souboru. Tlačítko má přidělen tooltip, který informuje o popisu tlačítka, ale také o názvu a cestě k aktuálnímu výstupnímu souboru. Panel obsahuje ještě jedno tlačítko (*Search*/*Stop*), které bude vysvětleno později u příslušné třídy (`SearchButton`).

Pod formulář je přidána textová oblast, do které se vypisují informace o vyhledávání. V průběhu vyhledávání se do ní vpisují odkazy, ze kterých se právě získávají informace. Po skončení vyhledávání se vypíše jednoduchá statistika, obsahující údaje o celkovém čase vyhledávání, počtu nalezených publikací, poslední prohledávané generaci (ať už z důvodu překročení maximální generace nebo neexistence následující generace) a nakonec ještě o počtu připojení k serveru.

Textová oblast v sobě obsáhne pouze 600 řádek (specifikováno soukromou statickou konstantou třídy `GUI`), což zhruba odpovídá informacím o stu publikací. Toto omezení musí být zavedeno proto, aby se zbytečně nezabírala fyzická paměť, kdy by postupně docházelo k jejímu vyčerpání (viz obr. A.12). Přebytečné řádky se odstraňují při vkládání nových informací v metodě `appendHrefArea`. Z objektu typu `JTextArea` se zjistí počet řádek v textové oblasti. Rozdílem tohoto počtu a maximálního počtu řádek se zjistí počet řádek k odstranění. Posléze se v cyklu nalezne počet znaků tyto řádky reprezentující a nakonec se z objektu s obsahem textové oblasti (`Document`) odstraní všechny znaky, které mají index menší než je takto zjištěný počet.

Úplně dospod grafického rozhraní byl plánován progres bar a status bar. Při implementaci jsem oba tyto prvky spojil v jeden. Je použit standardní objekt

`JProgressBar`, kterému je měněn text tak, aby uživatele informoval o stavu aplikace. Tím jsem eliminoval použití dalšího řádku, který by ještě natáhl již tak celkem převýšené okno aplikace. V případě, že neprobíhá vyhledávací proces, je progres bar v deterministickém módu s postupem nastaveným na nulu, aby se řádek jevil jako jednoduchý status bar. Pokud bylo vyhledávání spuštěno, progres bar se přepne do nedeterministického módu, čímž se zobrazí pohybující se pruh. Tímto způsobem se uživateli indikuje, že aplikace provádí nějakou činnost, tedy že probíhá vyhledávání.

Třída `SearchButton`

Tato třída je oděděna od třídy `JButton` a charakterizuje tlačítko ovládající spuštění a zastavení vyhledávání. Toto tlačítko má dva stavy, které jsou dány použitými akcemi (`SearchAndSaveAction` a `StopSearchAction`), mezi nimiž se přepíná pomocí metod `setSearchAction` a `setStopAction`.

Akce `SearchAndSaveAction` je zodpovědná za spuštění vyhledávání. Pro tento účel vždy vytváří nový objekt typu `Extractor` z aplikační vrstvy a po nastavení ho uchová v hlavní třídě `Main`. Nastavení spočívá v předání dvou údajů ze vstupních polí z grafického rozhraní nacházejících se ve třídě `GUI` (šířka nulté generace a maximální generace). Předpokládané hodnoty polí jsou nezáporná celá čísla. Kontrolu, zda-li tomu tak je, má na starost právě tato akce.

Kontrola se provádí tak, že se převzatý textový řetězec pokusí převést na celé číslo pomocí metody `Integer.parseInt`. Pokud řetězec necharakterizuje číslo, je vyhozena výjimka, při jejímž zachycení se dá pomocí výstražného dialogu uživateli najevo, že nastala chyba. Pokud vše proběhne v pořádku, na získané číslo se ještě pro všechny případy uplatní absolutní hodnota a číslo je posléze předáno `Extractor`u.

Obdobný proces akce provádí s třídou `Connector`, avšak zde se nevytváří nový objekt, ale pouze se recykluje starý metodou `init`. Předává se pouze jediný údaj a tím je prodleva připojení v milisekundách. Tímto končí proces přípravy a následuje spuštění procesu vyhledávání ve vlastním vlákně pomocí metody `start` v `Extractor`u.

Akce `StopSearchAction` má na starost zastavení procesu vyhledávání. Toho dosáhne jednoduše voláním metody `stopSearch` aktuálního `Extractor`u.

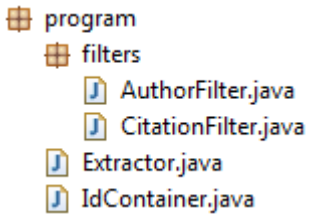
Ještě deaktivuje vyhledávací tlačítko, aby nebylo možné začít další vyhledávání, dokud se ukončuje staré.

Třída `Text`

Třída `Text` shromažďuje veškeré texty aplikace týkající se grafického uživatelského rozhranní. Obsahuje tak pouze spoustu statických řetězcových konstant, ke kterým se přistupuje při sestavování grafického rozhraní. Jsou přítomny texty popisků, tooltipů, chybových hlášení a jejich okenních titulků. Toto shromáždění textů umožňuje jejich snadnou změnu a je velkou pomůckou pro případnou budoucí implementaci internacionalizace.

6.2.2 Aplikační vrstva

Do aplikační vrstvy patří celkem čtyři třídy - dvě hlavní a další dvě pomocné v podbalíku `filters`, viz obr. 6.2.



Obr. 6.2: Struktura aplikační vrstvy.

Třída `Extractor`

Třída `Extractor` je nejdůležitější třídou aplikační vrstvy a vůbec celé aplikace. Řídí totiž celý proces vyhledávání a podílí se také na parsování informací. Její činnost se spouští ve zvláštním vlákně, aby bylo možné během vyhledávání dále pracovat s grafickým rozhraním aplikace.

Po spuštění vlákna (metodou `start`) si po `GUI` vyžádá textový řetězec obsahující dotaz sestavený ze vstupního formuláře (vyhledávací část URL za otazníkem) a následně volá hlavní řídící metodu `extract`. Tato metoda obsahuje dva vyhledávací cykly, které v podmínce oba testují, zda má proměnná `active` hodnotu `true` (tím se zajistí, že se při přerušení vyhledávání proces zastaví až po dokončení rozpracované

publikace). Nejprve však ještě musí získat první odpověď serveru (s prvními výsledky vyhledávání) a zjistit, zda služba ACM DL nějaké výsledky na náš dotaz našla.

ACM totiž v případě, že nebyly žádné výsledky nalezeny nepošle prázdnou stránku, jako je tomu např. na konkurenčním Scopusu, ale v navrácené stránce jsou vypsány stejné publikace, jaké jsou vypsány v případě prázdného dotazu. Jedinou indikací, že se vyhledávání nezdařilo, je pak jen hluboko do HTML zanořený text se žlutým pozadím "(*dotaz*) was not found".

Pokud tedy nebyl chybový text nalezen, pokračuje metoda do prvního cyklu. Ten má za úkol získat všechny identifikátory publikací, které spadají do první generace (jejich maximální počet je specifikován uživatelem v GUI). Využívá se při tom faktu, že na každou vypsanou publikaci připadá odkaz (element `<a>`) vedoucí k jejímu detailu. Tyto odkazy mají navíc jako jediné elementy v HTML dokumentu atribut `target` s hodnotou `"self"`. Nalezení odkazů je tedy jednoduchou záležitostí, neboť použitý *HTML Parser* nabízí filtrování elementů, které mají určitý atribut. Každý odkaz ve své URL obsahuje ACM ID publikace, které je již posléze snadné získat. Nalezená ID se uchovávají v objektu typu `IdContainer` popsaném níže. Nakonec se ještě do `IdContainer`u vloží jedna hodnota `null`, která označuje konec generace.

Druhý cyklus postupně vybírá identifikátory z `IdContaineru`, dokud není prázdný. Pokud narazí na hodnotu `null` a stále ještě existují další ID ke zpracování, inkrementuje číslo generace a znovu vloží hodnotu `null` do `IdContaineru` jako indikátor konce další generace. Pro každé ACM ID je volána metoda `processPublication`, která řídí samotné získávání informací o konkrétní publikaci. Na konci cyklu se ještě aktualizuje stavový řádek v GUI, aby byl uživatel v obraze.

Informace o publikaci se získávají ze dvou podkladů a před výpisem se ukládají do objektu typu `PublicationType`, jenž je popsán v datové vrstvě, do které patří. Zmíněná metoda `processPublication` nejprve získá stránku s detailem publikace z odkazu:

"http://dl.acm.org/citation.cfm?id=ACM_ID&coll=DL&dl=GUIDE&preflayout=flat".

Poté pomocí filtru `AuthorFilter` nalezne všechny autory publikace i s jejich afilacemi. Následně stránku znovu profiltruje tentokrát pomocí `CitationFilter`, aby získala seznam citujících publikací. Pokud se vyhledávací proces nenachází v poslední prohledávané generaci, přidají se identifikátory citujících publikací do `IdContainer`u k pozdějšímu zpracování v následující generaci. Druhým podkladem pro sběr informací je BibTeX exportovaný službou ACM DL pro každou publikaci. Jeho zpracování je odsunuto do metody `fillPubinInfo`. Po této činnosti se pošle naplněný objekt typu `PublicationType` datové vrstvě k výpisu.

Metoda `fillPubinInfo` má za úkol získat data ve formátu BibTeX pro příslušnou publikaci, zpracovat je a nakonec je správně vyplnit do předaného objektu typu `PublicationType`. Data jsou dostupná na adrese:

"http://dl.acm.org/exportformats.cfm?id=ACM ID&expformat=bibtex".

Místo podtržené části je třeba dosadit jednoduché ACM ID publikace, tj. jedno číslo bez tečky a bez identifikátoru nadřazené publikace.

Zpracování a ukládání dat je prováděno společně níže popsaným způsobem. Z BibTeXu se získává sedm položek, kterými jsou název publikace, název nadřazené publikace, její svazek, číslo vydání, rok vydání, stránky a vydavatel (posledních šest položek je vysvětleno v implementaci s popisem výstupního XML, část 6.1). Naneštěstí se existence jejich výskytu a dokonce i jejich název liší pro každý typ publikace (článek, kniha, kapitola atd.). Tento typ není možné lehce získat a tak se prochází všechny přípustné možnosti v pořadí od nejvíce specifického označení po nejméně specifické.

Pro ilustraci uvedu dva příklady. Prvním je název nadřazené publikace. Ten se v BibTeXu může vyskytovat pod názvy `journal`, `booktitle` nebo `title`. Druhým příkladem je položka `pages`, která je přítomna u typu článek, ale již ne (logicky) u sborníku z konference.

Tímto končí zpracování jedné publikace a může se vstoupit do další iterace hlavního řídícího cyklu.

Třída `IdContainer`

`IdContainer` je třída pro uchovávání nalezených identifikátorů, které mají být později zpracovány. Implementuje abstraktní datovou strukturu typu FIFO a jedná se

tedy o frontu. Díky tomuto faktu probíhá vyhledávací proces jako BFS (breadth first search) čili prohledávání do šířky.

Třída `IdContainer` používá dva objekty pro uchovávání identifikátorů. `ArrayList` je používán pro uchovávání identifikátorů, které se budou později zpracovávat. Naproti tomu do `HashSet`u se ukládají identifikátory publikací, které již zpracovány byly, a slouží pro kontrolu, zda jsme již nově nalezenou publikaci dříve nezpracovávali (nebo zda se již nachází ve frontě). Tímto se vyhneme zacyklení, které by mělo za následek nekonečné zpracovávání duplicitních dat.

Takovýmto ukládáním však dochází k postupnému vyčerpání přidělené paměti. Když k tomu dojde, aplikace skončí s chybovým výpisem, viz obr. A.12. Uvedený problém se v této práci neřeší, neboť pro přidělenou paměť o 2GB se dokázalo stáhnout skoro pět tisíc publikací, což pro otestování funkčnosti aplikace stačí. Problém tudíž není tak důležitý a jeho řešení uvedeno jako jedno z možných vylepšení.

Třída obsahuje čtyři metody. Metoda `push` nejprve zkontroluje, zda nalezené ID již nebylo zpracováno nebo zda již čeká ve frontě na zpracování, a pokud je identifikátor skutečně nový, vloží ho do zmíněného `HashSet`u a na konec `ArrayList`u. Pokud má předávaný parametr hodnotu `null`, značí ukončení generace a je bez dalších testů zařazen na konec `ArrayList`u. Není ovšem zařazen do `HashSet`u, protože duplicitu pro něj třeba testovat není.

Druhou metodou je metoda `pop`, která jednoduše odebere prvek z počátku `ArrayList`u a ten poté vrátí. Třetí metoda `hasNext` testuje, zda `ArrayList` ještě obsahuje nějaké identifikátory ke zpracování a poslední `getCount` vrací počet uchovaných ID určených pro zpracování a je používána pro doplnění stavu aplikace do stavového řádku.

Třída `AuthorFilter`

Třída `AuthorFilter` implementuje rozhraní `NodeFilter` použité knihovny *HTML Parser*. Tento filtr má za úkol ve webové stránce s detailem publikace identifikovat elementy s informací o jménech autorů a jejich afilací.

Takovýto element obsahuje atribut `title` s hodnotou `"Author Profile Page"`, což nám výrazně pomůže s jeho identifikací. Drobnou nepříjemností je,

že tento atribut obsahují také elementy autorů z nadřazené publikace, které jsou vypsány pod hypertextovou kotvou PUBLICATION. Z tohoto důvodu je ve třídě přítomna příznaková proměnná `authorSection`, která filtru značí, zda má elementy se zmíněným atributem propouštět nebo ne. Takto se lze rozhodovat díky skutečnosti, že při filtrování se jednotlivé uzly HTML stromu procházejí postupně v pořadí výskytu ve zdrojové stránce. Pokud tedy víme, že se naši hledaní autoři nacházejí ve stránce v horní části, zinicializujeme příznak na hodnotu `true` a jakmile tato část skončí, nastavíme hodnotu `false`.

Nyní, když je hledaný element konečně identifikovaný, mohou se provést příslušné akce. Element se jménem autora tedy máme nalezený a postačilo by provádění metody ukončit návratem hodnoty `true`. Aby ale podobné filtrování nebylo nutné provádět i pro afilace autorů (složitější, protože afilace nemají žádný specifický atribut), získáme afilaci ze "sousedního" elementu již nyní, uložíme ji do `ArrayList`u a zpřístupníme ji pod pořadovým číslem metodou `getAffiliation`.

V sousedním elementu (je to element `<td>`) může být informace o afilaci uvedena třemi způsoby. Způsob je závislý na existenci profilu příslušné afilace na serveru ACM (podobný jako detail publikace) a na tom, zda autor vůbec nějakou afilaci má. Pokud afilaci nemá, element je prázdný a jako afilace bude použit řetězec "none". Pokud profil afilace existuje, je její název uveden jako odkaz, tedy v elementu `<a>`. Poslední možností je, že profil afilace neexistuje. Potom je její název uveden jako text v elementu `<small>`.

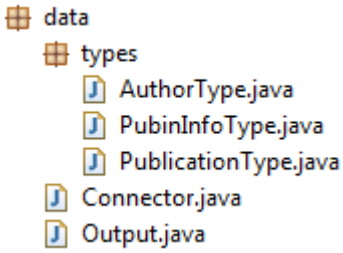
Třída `CitationFilter`

Třída `CitationFilter` je obdobná třídě `AuthorFilter`, jen je o něco jednodušší, poněvadž má za úkol identifikovat odkazy na všechny citující publikace. Určení příznaku výskytu v sekci s citujícími publikacemi je o něco složitější, jelikož je sekce umístěna uprostřed stránky a atributy, které využívám k identifikaci, jsou přítomny před i za touto sekcí. Je tedy třeba ošetřit obě její strany. K tomu jsou využity dvě hypertextové kotvy. Na začátku sekce je to kotva CITED BY. Po jejím nalezení se nastaví příznak `citationSection` na hodnotu `true`. Na konci sekce je použita kotva INDEX TERMS a příznak se nastaví na hodnotu `false`.

Jak již bylo řečeno, ACM ID se nacházejí v URL jako vyhledávací parametr. V sekci s citujícími publikacemi se tedy hledají elementy `<a>` a poté se kontroluje jejich atribut `href`. Pokud odkaz vede na detail publikace, přičemž víme, že v takovém případě se v odkazu nachází ACM ID, jeho atribut `href` začíná na *"citation.cfm?"*. Předešlý postup stačí k tomu, aby se úspěšně identifikovaly všechny požadované uzly procházející tímto filtrem.

6.2.3 Datová vrstva

Do datové vrstvy je zařazeno celkem pět tříd. Tři z nich jsou pouze typové třídy určující interní formát dat pro manipulaci s nimi a proto jsou umístěny do podbalíku `types`. Zbylé dvě jsou výkonné třídy pro získávání a ukládání dat. Struktura vrstvy je zobrazena na 6.3.



Obr. 6.3: Struktura datové vrstvy.

Třída `Connector`

Třída `Connector` se stará o veškeré aspekty komunikace se serverem ACM DL. Patří sem dodržení prodlevy mezi připojeními, připojení k serveru, získání odpovědi, práce s cookies a HTTP přesměrováním. Všechny tyto aspekty jsou v jazyce Java velmi snadné, protože jsou ve standardní knihovně velmi dobře podporovány.

Práce s cookies se v jazyce Java v základu odehrává zcela automaticky. Ovlivnit ji lze především třemi třídami. `CookieHandler` umožňuje nastavit základního správce cookies. `CookieManager` je oním správcem cookies a umožňuje nastavení základního úložiště pro cookies (`CookieStore`) a strategii pro ukládání cookies. Pro účely této práce vytvářím objekt `CookieManager` se základním úložištěm a strategií, která povolí všechna cookies. Na úložiště si ponechám referenci a vytvořeného správce cookies nastavím jako základního.

Potřeba reference na úložiště cookies pramení z omezení služby ACM DL na počet připojení za čas. Pokud probíhají dotazy na server příliš často, server nepošle regulérní odpověď a místo toho odpoví chybou číslo 403 (viz obr. A.11). Aby k tomu nedocházelo, je ve třídě přítomna metoda `renewCookies`, která smaže všechna cookies z úložiště a po dodržené prodlevě si vyžádá cookies nová (jedná se především o cookies CFID a CFTOKEN). K této myšlence mě dovedlo pozorování, že po zablokování aplikace a jejím restartu vyhledávání znovu okamžitě fungovalo, zatímco po převzetí poslední URL (i se zmíněnými cookies) do webového prohlížeče se zobrazila ona chyba 403. Metoda je volána po uplynutí každého počtu připojení definovaném konstantou `CONNECTIONS_PER_RENEWAL`.

Bez tohoto opatření bylo možné stáhnout v průměru jen okolo sta publikací. V průběhu implementace jsem zkoušel různé jiné možnosti odstranění tohoto problému včetně cyklického čekání s exponenciálním navýšením prodlevy až na řádově hodiny. S úspěchem se setkala až metoda s obnovou cookies a tak je zatížení serveru touto aplikací řízeno pouze prodlevou mezi připojeními, viz níže o metodě `waitDelay`, případně dobou zpracování dat.

Dalším aspektem, který třída `Connector` řeší, je automatické přesměrování HTTP komunikace. I tato činnost je v jazyce Java v základu podporována a tak spíš jen pro jistotu explicitně automatické přesměrování povoluji.

O získání odpovědi ze serveru se pro celou aplikaci stará výhradně metoda `getResponse`. Nejprve zkontroluje, zda již není čas na obnovu cookies, následně počká zbylý čas na dodržení nastavené prodlevy mezi připojeními a poté započne se získáváním dat. O HTTP komunikaci se stará objekt typu `URL`, který se vytvoří z bázové adresy služby ACM DL a požadované relativní adresy. Nyní stačí metodou `openConnection` třídy `URL` otevřít spojení se serverem, čímž v našem případě získáme objekt typu `HttpURLConnection`, a poté získat vstupní proud díky metodě `getInputStream`. Následně metoda z tohoto proudu přečte v cyklu po jednotlivých řádcích celou odpověď.

Metoda `waitDelay` se stará o pozdržení nového připojení k serveru tak, aby časová prodleva od posledního připojení byla minimálně stejná nebo delší než hodnota nastavená uživatelem v GUI. Základní hodnota prodlevy je 2000 ms, tedy dvě

sekundy. Pozdržení metoda docílí snadno tím, že si nejprve jednoduchým odečtem zjistí zbývající dobu k dodržení prodlevy a pomocí metody `Thread.sleep` vyhledávací vlákno na tento čas uspí.

Třída `Output`

Třída `Output` je zodpovědná za výpis nalezených a zpracovaných dat do souboru na disk. Všechna data o jedné publikaci jsou v době výpisu dostupná v objektu typu `PublicationType`, který je popsaný níže. Pro výpis do souboru je použita technologie StAX.

V konstruktoru se vytvoří výstupní proud do uživatelem zvoleného souboru spolu se zapisovací strukturou technologie StAX. Poté se vypíše hlavička XML souboru a kořenový element navrženého a výše specifikovaného XML formátu.

Třída obsahuje jedinou veřejnou metodu, která jako parametr bere objekt typu `PublicationType`. Všechny informace, které jsou v tomto objektu obsažené, budou vypsány ve specifikovaném formátu. K řízení výpisu třída pro přehlednost používá čtyři specializované metody. Každá z nich je určena k výpisu jednoho typu informace.

Další užitečná metoda má název `indent`. Technologie StAX sama o sobě nezajišťuje odřádkování a odsazování jednotlivých elementů. Proto se o tuto činnost stará právě tato metoda. Jednoduše pošle na výstup znaky CR a LF a poté v cyklu několik zvolených odsazovacích řetězců. Kolik jich má vypsat zjistí podle svého parametru, který se při zanoření zvyšuje a při návratu naopak snižuje.

Poslední metoda `close` se stará o korektní ukončení zápisu. Nejprve ukončí zbývající koncové tagy dosud otevřených elementů a následně postupně vyprázdní a ukončí oba proudové zapisovače.

Třídy `PublicationType`, `AuthorType` a `PubinInfoType`

Tyto třídy určují komplexní datový typ charakterizující nalezenou publikaci. Mají v podstatě stejnou hierarchickou strukturu jako element `publication` v navrženém výstupním XML formátu a obsahují pouze několik proměnných určených pro data k vypsání a metody, které tato data při sestavování nastavují a při výpisu zpřístupňují.

7 Testování

Testování implementované aplikace probíhalo již v průběhu jejího vytváření následujícím způsobem. Jelikož implementace probíhala iterativně, kdy byla k dosavadnímu stavu programu přidána pouze jedna nová funkčnost za čas, probíhalo také testování postupně.

Nejprve jsem vždy novou funkčnost implementoval odděleně. Následně jsem tuto funkčnost ručně otestoval včetně všemožných speciálních případů, které mne jen napadly. Nakonec jsem takto otestovanou funkčnost zakomponoval do aplikace a znovu zkontroloval všechny návaznosti na zbytek aplikace během reálné činnosti.

Během vývoje byl průběžný postup aplikace pravidelně konzultován se zadavatelem. Zadavatel měl také možnost aplikaci průběžně zkoušet a všechny jeho připomínky byly postupně zapracovány do výsledné aplikace.

Na testování nebyly použity žádné speciální nástroje nebo automatické testy, kterým je např. JUnit. Vytvořená aplikace není zase až tolik rozsáhlá a tak absence automatického testování nevadí. Pro větší projekt bych však přistoupil i k této možnosti, protože by zajistila větší míru spolehlivosti a nezanedbatelnou úsporu času.

8 Závěr

Při pohledu na zadání této bakalářské práce mi zbývá již jen zhodnotit dosažené výsledky, pro což nastal čas právě nyní. Celá práce byla tvořena pod stálým dohledem zadavatele, jímž byly všechny příslušné činnosti vedeny prostřednictvím domluvených konzultací. Průběžný postup byl diskutován pravidelně jednou až dvakrát do měsíce v závislosti na objemu práce v uplynulé iteraci. Díky tomuto vedení se podařilo všechny body zadání úspěšně a v klidu splnit ke spokojenosti všech zúčastněných.

Velká pozornost byla věnována seznámení se službou ACM Digital Library, protože je důležité službu dobře znát vzhledem k tomu, že smyslem této práce je komunikace a získávání jejích informací. Mimo popsaného uživatelského pohledu (prostřednictvím webového prohlížeče) byly studovány také zdrojové kódy webových stránek, bez kterého by nešlo zjistit přesné umístění hledaných dat, což je pro jejich extrakci esenciální.

Dále jsem se zaměřil na analýzu dostupných technologií, přístupů a programovacích jazyků, které připadaly v úvahu k použití při implementaci praktické části bakalářské práce. Následný výběr řešení totiž může práci na implementaci aplikace ulehčit nebo naopak velmi ztížit. Kapitola s návrhem je potom svým účelem podobnou, ale již konkrétní pomůckou pro efektivní programování.

Nevýhodou aplikace je, že možný počet zpracovaných publikací je omezen. To je dáno tím, že veškeré nalezené identifikátory musí být aplikaci dostupné kvůli kontrole duplicity, která by způsobila zacyklení. Na testovacím HW o Intel Mobile Core 2 Quad Q9000 2GHz, 4GB RAM DDR3 a OS Microsoft Windows 7 Pro 64b při 2GB přidělené RAM a 2s prodlevy mezi připojeními k serveru bylo za 3 hodiny a 20 minut zpracováno 4717 publikací. Poté byla přidělená paměť vyčerpána.

Možné vylepšení je tedy nasnadě. Ukládání identifikátorů na disk by nejen vyřešilo zmíněný problém, ale také by dovolilo vyhledávání přerušit (včetně havárií) a později opět navázat. Dále by šlo doplnit vyhledávací formulář nebo přidat další data k extrakci.

Přehled zkratek

ACM	Association for Computing Machinery
ACM DL	ACM Digital Library
AJAX	Asynchronous JavaScript and XML
API	Application Programing Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Data Type Definition
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JAXB	Java Architecture for XML Binding
JDK	Java Development Kit
JVM	Java Virtual Machine
SGML	Standard Generalized Markup Language
SJSXP	Sun Java Streaming XML Parser
StAX	Streaming APIs for XML
URL	Uniform Resource Locator
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

Zdroje

[1] PÍSEK, Slavoj. HTML a XHTML: začínáme programovat: podrobný průvodce začínajícího uživatele. 1. vyd. Praha: Grada Publishing, 2003. 255 s. ISBN 80-247-0571-0.

[2] YOUNG, Michael J. XML krok za krokem. 2. vyd. Brno: Computer Press, 2006. 471 s. ISBN 80-251-1070-2.

[3] HEROUT, Pavel. Java a XML. 1. vyd. České Budějovice: Kopp, 2007. 313 s. ISBN 978-80-7232-307-4.

[4] HEROUT, Pavel. Učebnice jazyka Java. 3., rozš. vyd. České Budějovice: Kopp, 2007. 381 s. ISBN 978-80-7232-323-4.

[5] HEROUT, Pavel. Učebnice jazyka C. 3., upr. vyd. České Budějovice: Kopp, 1996. 269 s. ISBN 80-85828-21-9.

[6] LIBERTY, Jesse. Naučte se C++ za 21 dní. 2., aktualiz. vyd. Brno: Computer Press, 2007. 796 s. ISBN 978-80-251-1583-1.

[7] ACM Digital Library [online]. [cit. 2012-04-15]. <http://dl.acm.org/>

[8] ACM Digital Library Advanced Search [online]. [cit. 2012-04-15]. <http://dl.acm.org/advsearch.cfm>

[9] The JavaTM Tutorials [online]. [cit. 2012-04-15]. <http://docs.oracle.com/javase/tutorial/>

[10] W3C HTML [online]. [cit. 2012-04-15]. < http://www.w3.org/html/>

[11] The W3C Markup Validation Service [online]. [cit. 2012-04-15]. <http://validator.w3.org/>

[12] The Extensible Style Sheet Family (XSL) [online]. [cit. 2012-04-15]. <http://www.w3.org/Style/XSL/>

[13] TIOBE Software: Tiobe Index [online]. [cit. 2012-04-15]. <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>

[14] BibTeX [online]. [cit. 2012-04-15]. <http://www.bibtex.org/>

[15] RFC 1738 - A Gopher URL Format [online]. [cit. 2012-04-15]. <tools.ietf.org/html/rfc1738>

[16] Learn Visual C# on MSDN [online]. [cit. 2012-04-15]. <http://msdn.microsoft.com/en-us/vstudio/hh341490>

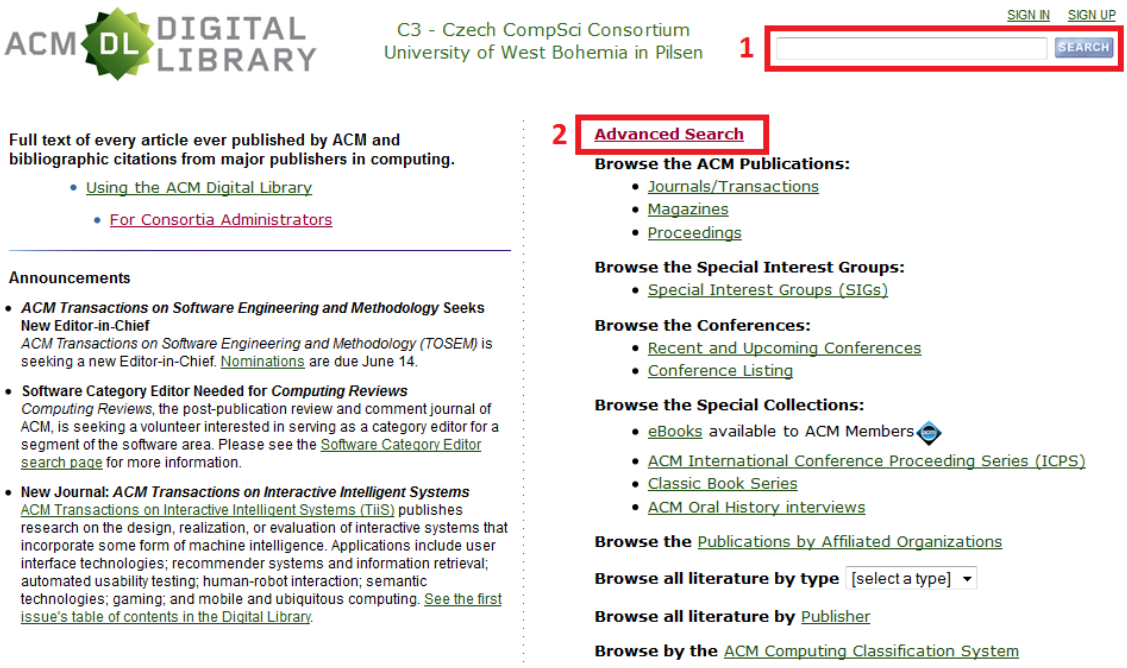
[17] The SAXON XSLT and xQuery Processor [online]. [cit. 2012-04-15]. <http://saxon.sourceforge.net/>

[18] The Apache Xalan Project [online]. [cit. 2012-04-15]. <http://xalan.apache.org/>

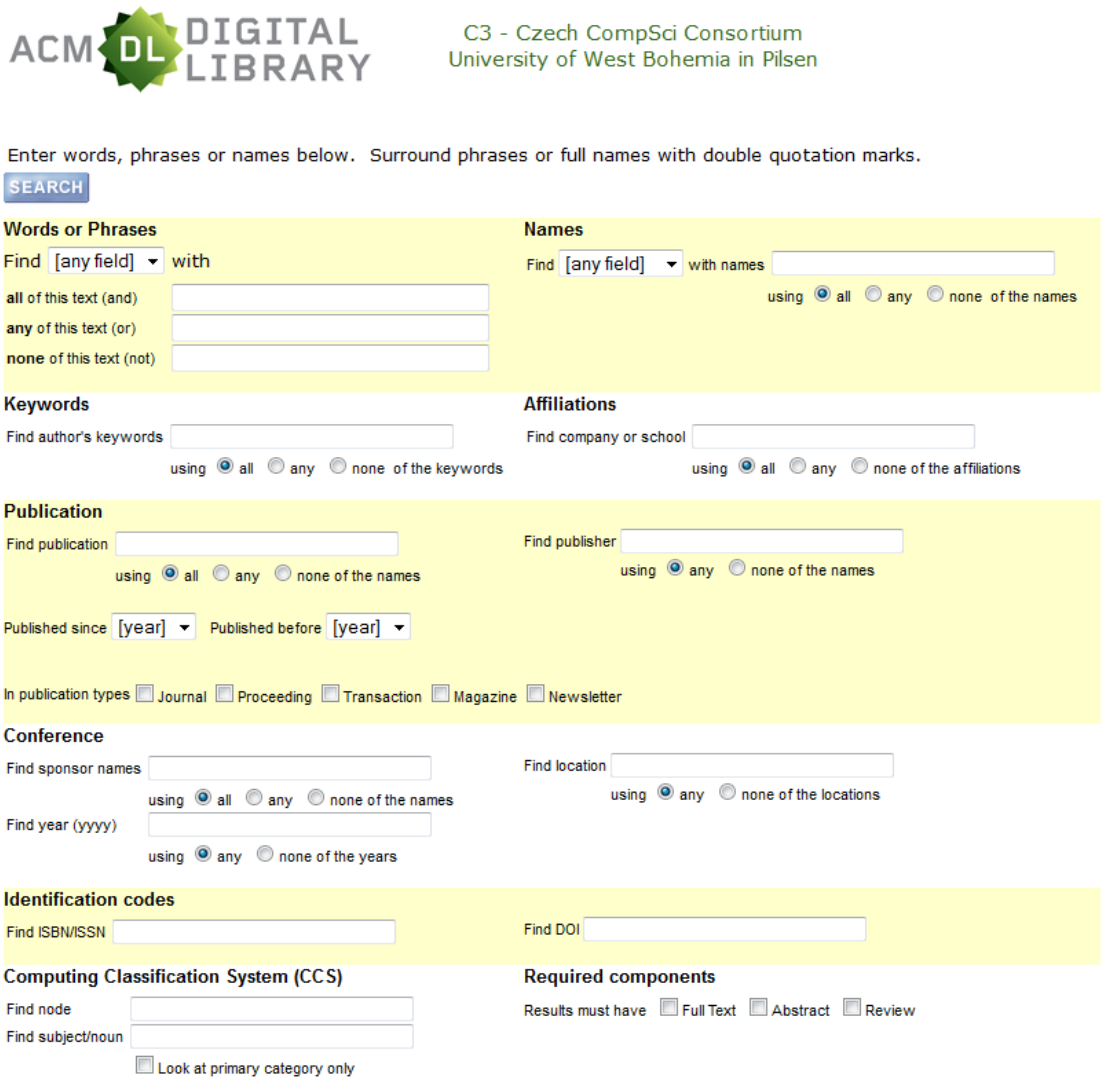
[19] Open Source HTML parsers in Java [online]. [cit. 2012-04-15]. <http://java-source.net/open-source/html-parsers>

[20] HTML Parser - HTML Parser [online]. [cit. 2012-04-15]. <http://htmlparser.sourceforge.net/>

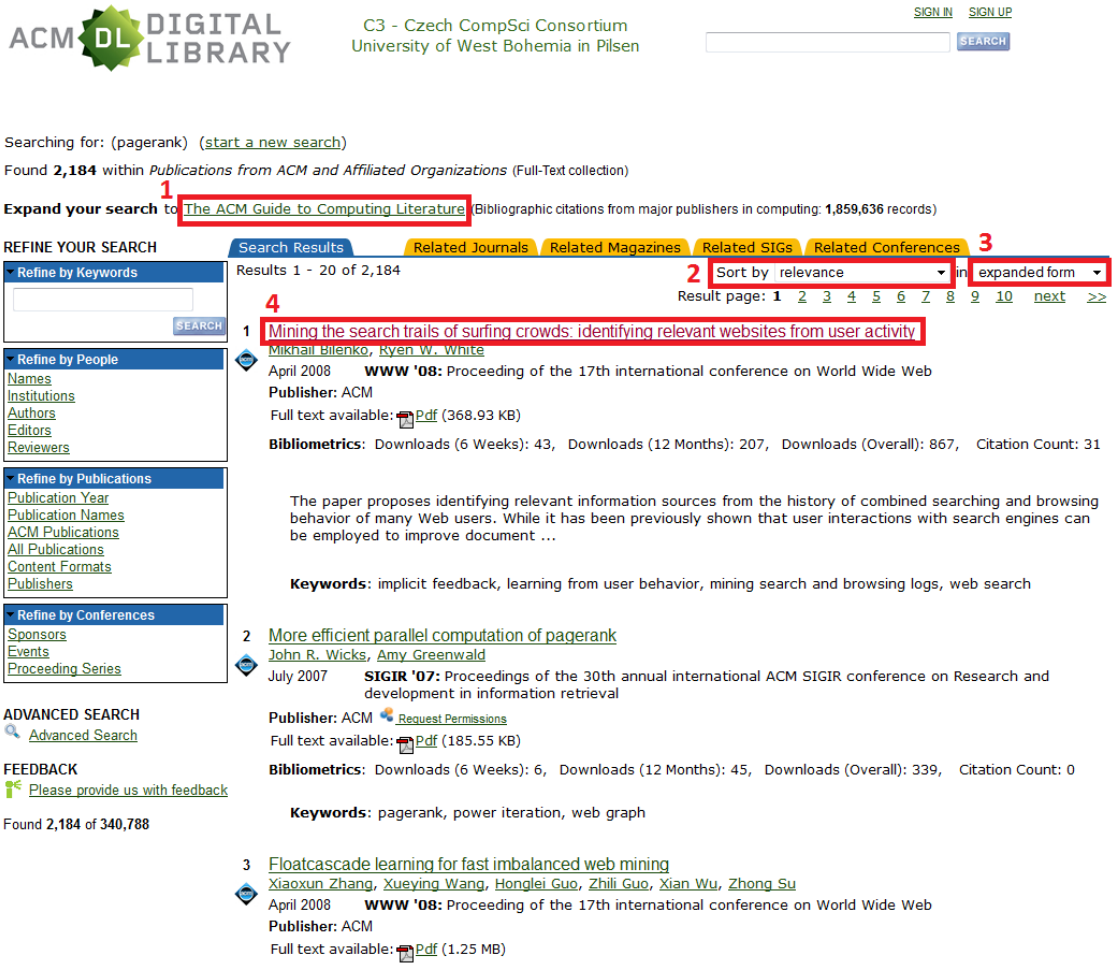
Příloha A: Obrázky



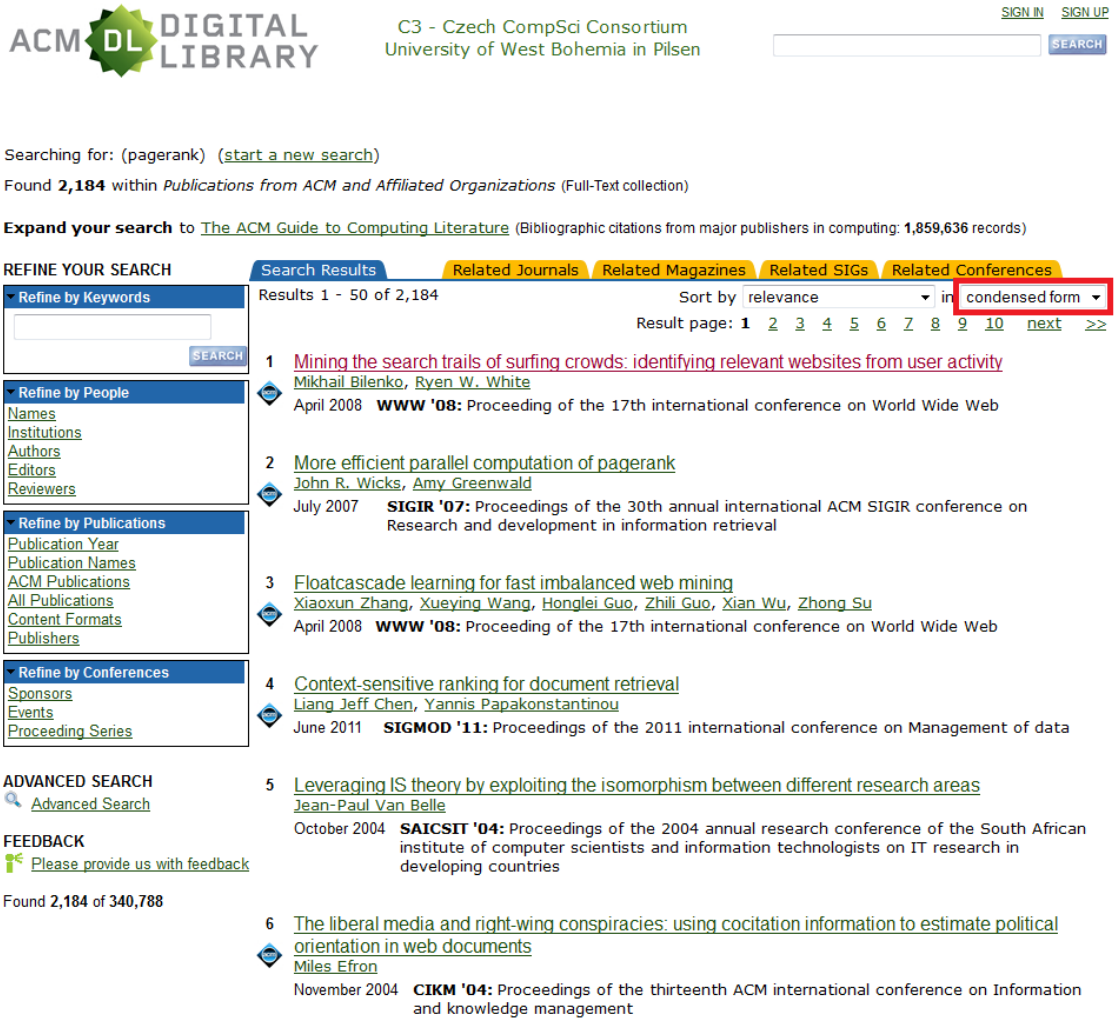
Obr. A.1: Úvodní stránka služby ACM Digital Library: (1) jednoduché vyhledávání, (2) odkaz na pokročilé vyhledávání.



Obr. A.2: Formulář pokročilého vyhledávání služby ACM DL.

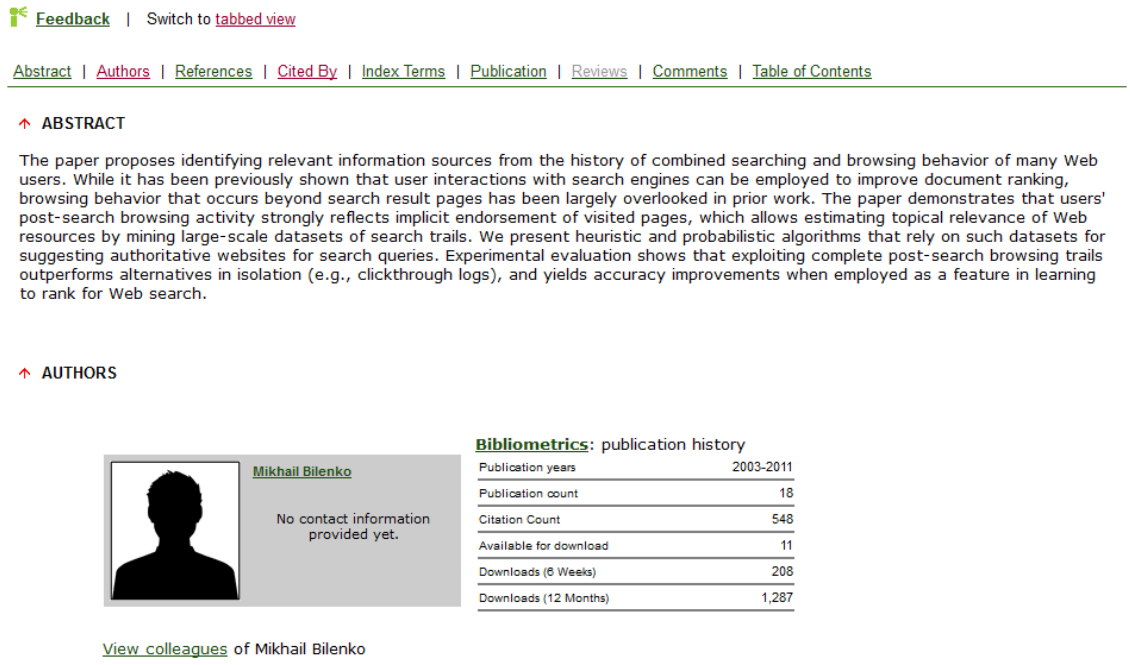


Obr. A.3: Výsledky vyhledávání - expanded form: (1) přepínač knihoven ACM a GUIDE, (2) volba možností řazení, (3) volba množství informací na stránce, (4) jednotlivé vyhledané záznamy.



Obr. A.4: Výsledky vyhledávání - condensed form.

Obr. A.5: Detail publikace - tabbed view: (1) seznam autorů, (2) formáty pro export, (3) přepínač stylu stránky, (4) sekce s citacemi.



Obr. A.6: Spodní část detailu publikace při single page view.

REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 P. Boldi and S. Vigna. Codes for the world-wide web. Internet Mathematics Journal, 2(4):405--427, 2005.

2 S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University Technical Report, 2003.

3 C. Kohlschutter, P.-A. Chirita, and W. Nejdl. Efficient parallel computation of pagerank. In Advances in Information Retrieval, volume 3936 of Lecture Notes in Computer Science, pages 241--252. Springer, 2006.

4 L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

5 Yuan Wang , David J. DeWitt, Computing pagerank in a distributed internet search system, Proceedings of the Thirtieth international conference on Very large data bases, p.420-431, August 31-September 03, 2004, Toronto, Canada

CITED BY

Citings are not available

Obr. A.7: Ukázka seznamu referencí (references).

CITED BY

32 Citations

Deepayan Chakrabarti , Ravi Kumar , Kunal Punera, Quicklink selection for navigational query results, Proceedings of the 18th international conference on World wide web, April 20-24, 2009, Madrid, Spain

Deirdre Lungley , Udo Kruschwitz, Automatically Maintained Domain Knowledge: Initial Findings, Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval, April 06-09, 2009, Toulouse, France

Wenhui Liao , Isabelle Moulinier, Feature engineering on event-centric surrogate documents to improve search results, Proceeding of the 18th ACM conference on Information and knowledge management, November 02-06, 2009, Hong Kong, China

Milad Shokouhi , Leif Azzopardi , Paul Thomas, Effective query expansion for federated search, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2009, Boston, MA, USA

Fan Guo , Chao Liu , Yi Min Wang, Efficient multiple-click models in web search, Proceedings of the Second ACM International Conference on Web Search and Data Mining, February 09-12, 2009, Barcelona, Spain

Huihsin Tseng , Longbin Chen , Fan Li , Ziming Zhuang , Lei Duan , Belle Tseng, Mining search engine clickthrough log for matching *N*-gram features, Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2, August 06-07, 2009, Singapore

Bo Zhou , Yiqun Liu , Min Zhang , Yijiang Jin , Shaoping Ma, Incorporating web browsing activities into anchor texts for web search, Information Retrieval, v.14 n.3, p.290-314, June 2011

Ryen W. White , Jeff Huang, Assessing the scenic route: measuring the value of search trails in web logs, Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2010, Geneva, Switzerland

Adish Singla , Ryen White , Jeff Huang, Studying trailfinding algorithms for enhanced web search, Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2010, Geneva, Switzerland

Ravi Kumar , Andrew Tomkins, A characterization of online browsing behavior, Proceedings of the 19th international conference on World wide web, April 26-30, 2010, Raleigh, North Carolina, USA

Adish Singla , Ryen W. White, Sampling high-quality clicks from noisy click data, Proceedings of the 19th international conference on World wide web, April 26-30, 2010, Raleigh, North Carolina, USA

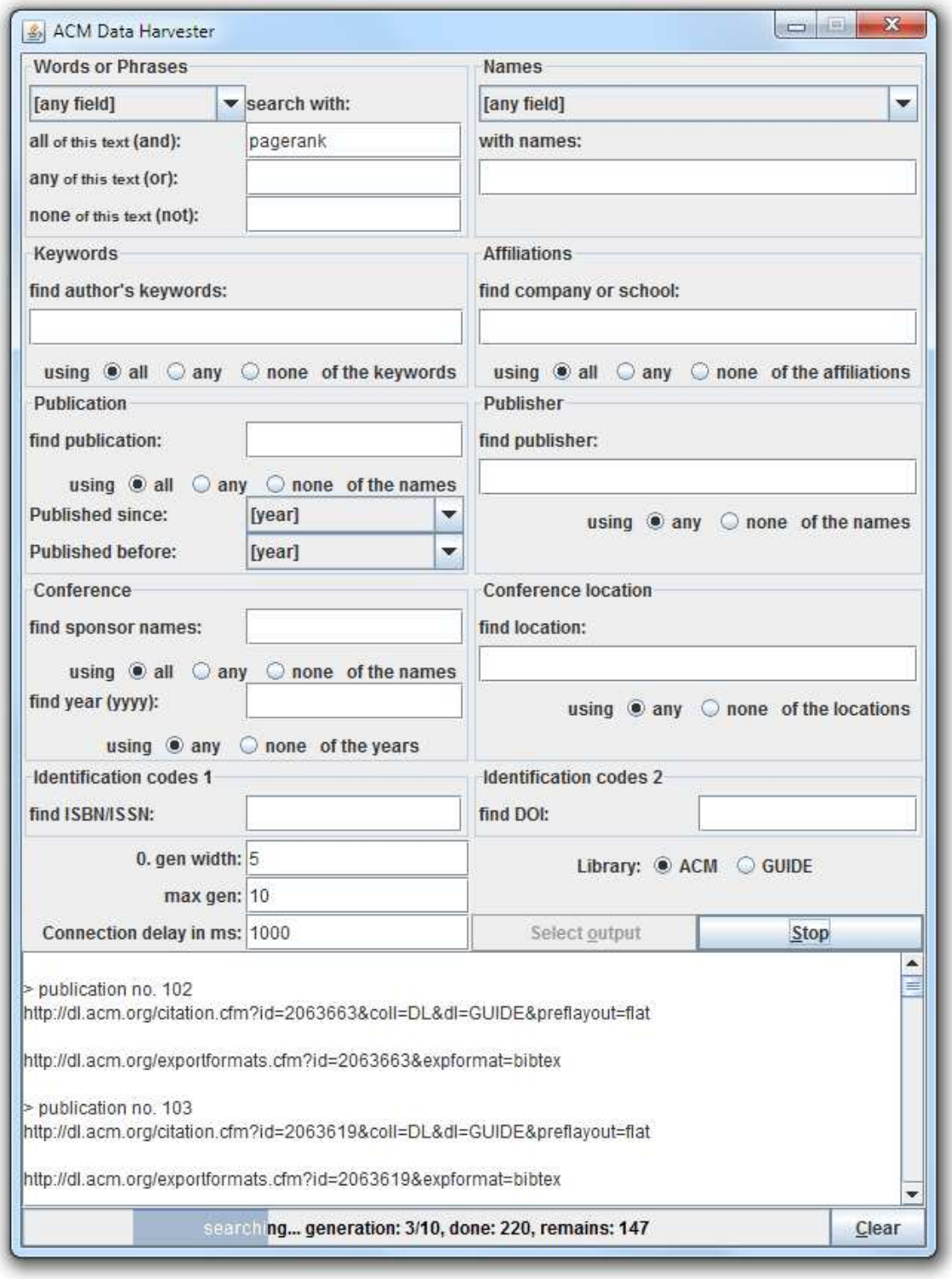
Yuchen Liu , Xiaochuan Ni , Jian-Tao Sun , Zheng Chen, Unsupervised transactional query classification based on webpage form understanding, Proceedings of the 20th ACM international conference on Information and knowledge management, October 24-28, 2011, Glasgow, Scotland, UK

Kosuke Takano , Xing Chen , Keisuke Masuda, A framework for a feedback process to analyze and personalize a document vector space in a feature extraction model, Information Technology and Management, v.10 n.2-3, p.151-176, September 2009

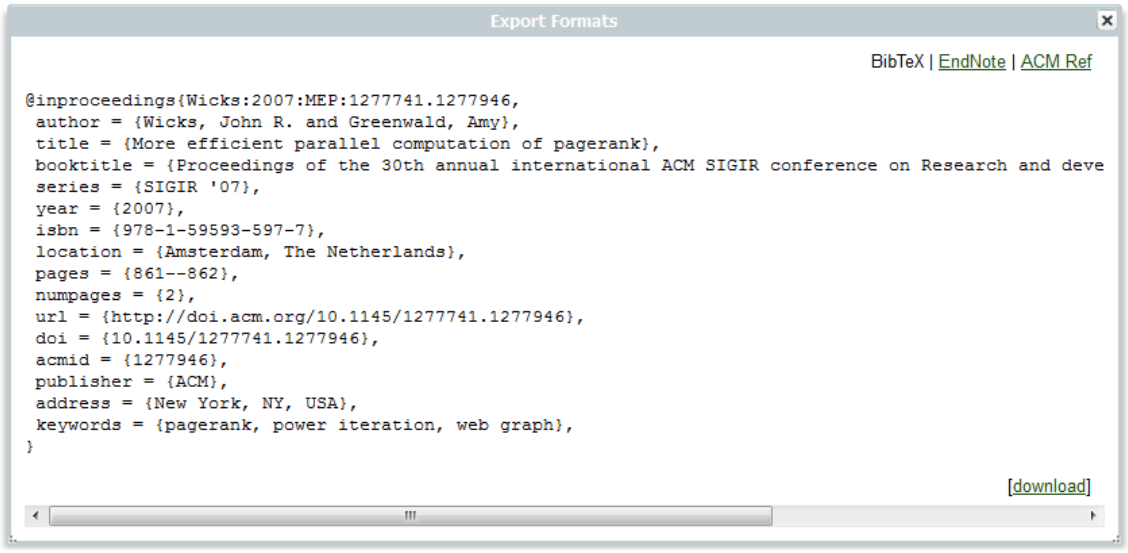
Pavel Dmitriev , Pavel Serdyukov , Sergey Chernov, Enterprise and desktop search, Proceedings of the 19th international conference on World wide web, April 26-30, 2010, Raleigh, North Carolina, USA

Hongbo Deng , Irwin King , Michael R. Lyu, Entropy-biased models for query representation on the click graph, Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, July 19-23, 2009, Boston, MA, USA

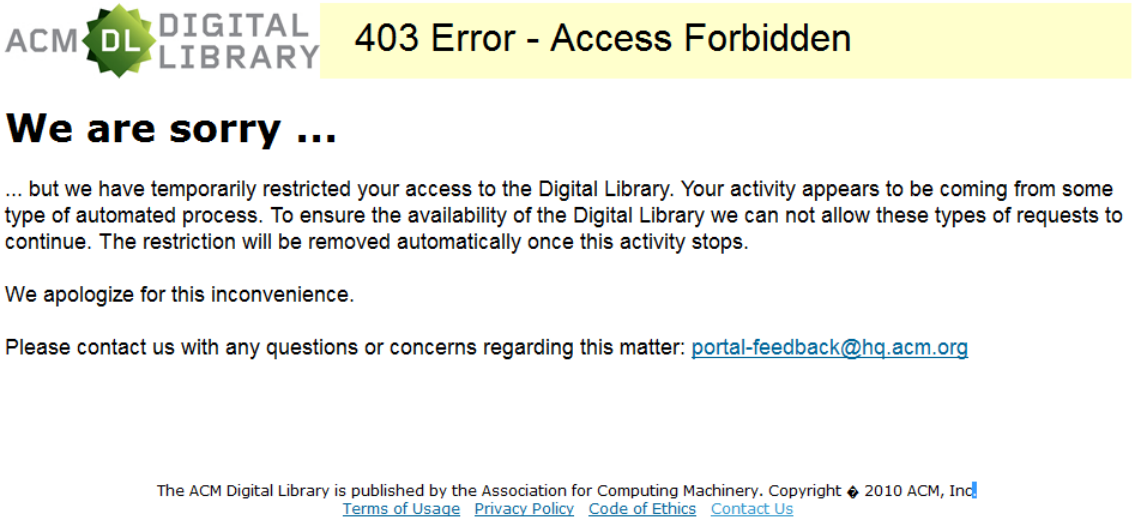
Obr. A.8: Ukázka seznamu citací (cited by).



Obr. A.9: Grafické uživatelské rozhraní implementované aplikace.



Obr. A.10: Ukázka formátu BibTeX.



Obr. A.11: Zablokování uživatele serverem při příliš častém dotazování.

```
Exception in thread "D3D Screen Updater" java.lang.OutOfMemoryError: Java heap space
        at sun.java2d.d3d.D3DScreenUpdateManager.run(Unknown Source)
        at java.lang.Thread.run(Unknown Source)
Exception in thread "AWT-EventQueue-0" java.lang.OutOfMemoryError: Java heap space
        at java.awt.GradientPaint.<init>(Unknown Source)
        at java.awt.GradientPaint.<init>(Unknown Source)
        at javax.swing.plaf.metal.MetalUtils$GradientPainter.getGradient(Unknown Source)
        at javax.swing.plaf.metal.MetalUtils$GradientPainter.drawVerticalGradient(Unknown Source)
        at javax.swing.plaf.metal.MetalUtils$GradientPainter.paintToImage(Unknown Source)
        at sun.swing.CachedPainter.paint0(Unknown Source)
        at sun.swing.CachedPainter.paint(Unknown Source)
        at javax.swing.plaf.metal.MetalUtils$GradientPainter.paint(Unknown Source)
        at javax.swing.plaf.metal.MetalUtils.drawGradient(Unknown Source)
        at javax.swing.plaf.metal.MetalButtonUI.update(Unknown Source)
        at javax.swing.JComponent.paintComponent(Unknown Source)
Exception in thread "Thread-3" java.lang.OutOfMemoryError: Java heap space
        at java.lang.String.substring(Unknown Source)
        at org.htmlparser.lexer.StringSource.getString(StringSource.java:361)
        at org.htmlparser.lexer.Page.getText(Page.java:1056)
        at org.htmlparser.lexer.PageAttribute.getName(PageAttribute.java:210)
        at org.htmlparser.PrototypicalNodeFactory.createTagNode(PrototypicalNodeFactory.java:500)
        at org.htmlparser.lexer.Lexer.makeTag(Lexer.java:1181)
        at org.htmlparser.lexer.Lexer.parseTag(Lexer.java:1157)
        at org.htmlparser.lexer.Lexer.nextNode(Lexer.java:368)
        at org.htmlparser.scanners.CompositeTagScanner.scan(CompositeTagScanner.java:110)
        at org.htmlparser.util.IteratorImpl.nextNode(IteratorImpl.java:91)
        at org.htmlparser.Parser.parse(Parser.java:700)
        at cz.zcu.kiv.prj5.data.Extractor.zpracujPublikaci(Extractor.java:224)
        at cz.zcu.kiv.prj5.data.Extractor.extract(Extractor.java:174)
        at cz.zcu.kiv.prj5.data.Extractor.run(Extractor.java:94)
Exception in thread "AWT-EventQueue-0" java.lang.OutOfMemoryError: Java heap space
```

Obr. A.12: Chybový výpis po vyčerpání přidělené paměti úschovou všech ID v RAM.

Příloha B: Výstupní XML

Uvedený soubor byl vygenerován 2.5.2012 pro následující dotaz:

- Find = Title
- all of this text = Pagerank
- Published since = 2008
- 0. gen width = 2
- max. gen = 10
- Library = ACM

```
<?xml version="1.0" encoding="UTF-8"?>
<publications>
    <publication id="1367497.1367738" gen="0" no="1">
        <title>Fast algorithms for topk personalized pagerank
            queries</title>
        <citedBy count="4">
            <id>2140441</id>
            <id>1920891</id>
            <id>1877771</id>
            <id>1997927</id>
        </citedBy>
        <authors count="3">
            <author id="81100021061" affiliation="IIT Bombay, Mumbai,
                  India">Manish Gupta</author>
            <author id="81435593822" affiliation="IIT Bombay, Mumbai,
                  India">Amit Pathak</author>
            <author id="81100424876" affiliation="IIT Bombay, Mumbai,
                  India">Soumen Chakrabarti</author>
        </authors>
        <pubinInfo volume="none" issue="none" pages="1225--1226"
            year="2008" publisher="ACM" title="Proceedings of the 17th
            international conference on World Wide Web"/>
    </publication>
    <publication id="1989323.1989425" gen="0" no="2">
        <title>Fast personalized PageRank on MapReduce</title>
        <citedBy count="1">
            <id>2095694</id>
        </citedBy>
        <authors count="3">
            <author id="81472643101" affiliation="Stanford University,
                  Stanford, CA, USA">Bahman Bahmani</author>
            <author id="81341488629" affiliation="Microsoft Research,
                  Redmond, WA, USA">Kaushik Chakrabarti</author>
            <author id="81486651185" affiliation="Google Inc.,
                  Mountain View, CA, USA">Dong Xin</author>
        </authors>
        <pubinInfo volume="none" issue="none" pages="973--984"
            year="2011" publisher="ACM" title="Proceedings of the 2011
            international conference on Management of data"/>
    </publication>
    <publication id="2140441" gen="1" no="3">
        <title>Fast and exact top-k search for random walk with
            restart</title>
        <citedBy count="0"/>
        <authors count="4">
            <author id="81367593789" affiliation="NTT Cyber Space
                  Labs, and The University of Tokyo">Yasuhiro
                  Fujiwara</author>
            <author id="81100257310" affiliation="NTT Cyber Space
                  Labs">Makoto Nakatsuji</author>
            <author id="81496655579" affiliation="NTT Cyber Space
                  Labs">Makoto Onizuka</author>
            <author id="81496672154" affiliation="The University of
                  Tokyo">Masaru Kitsuregawa</author>
        </authors>
        <pubinInfo volume="5" issue="5" pages="442--453" year="2012"
            publisher="VLDB Endowment" title="Proc. VLDB Endow."/>
    </publication>
    <publication id="1920891" gen="1" no="4">
        <title>Retrieving top-k prestige-based relevant spatial web
            objects</title>
```

```
        <citedBy count="5">
            <id>1996798</id>
            <id>2063641</id>
            <id>1989361</id>
            <id>1989363</id>
            <id>2159279</id>
        </citedBy>
        <authors count="3">
            <author id="81447594224" affiliation="Nanyang
                Technological University, Singapore">Xin Cao</author>
            <author id="81100468764" affiliation="Nanyang
                Technological University, Singapore">Gao
                Cong</author>
            <author id="81331495182" affiliation="Aarhus University,
                Denmark">Christian S. Jensen</author>
        </authors>
        <pubinInfo volume="3" issue="1-2" pages="373--384" year="2010"
            publisher="VLDB Endowment" title="Proc. VLDB Endow."/>
    </publication>
    <publication id="1877771" gen="1" no="5">
        <title>Dependable filtering: Philosophy and
            realizations</title>
        <citedBy count="0"/>
        <authors count="2">
            <author id="81351603868" affiliation="Eurecom, cedex,
                France">Matteo Dell'Amico</author>
            <author id="81100419515" affiliation="University College
                London, London, United Kingdom">Licia Capra</author>
        </authors>
        <pubinInfo volume="29" issue="1" pages="5:1--5:37" year="2010"
            publisher="ACM" title="ACM Trans. Inf. Syst."/>
    </publication>
    <publication id="1997927" gen="1" no="6">
        <title>Index design and query processing for graph conductance
            search</title>
        <citedBy count="0"/>
        <authors count="3">
            <author id="81100424876" affiliation="IIT Bombay, Powai,
                Mumbai, India">Soumen Chakrabarti</author>
            <author id="81435593822" affiliation="IIT Bombay, Powai,
                Mumbai, India">Amit Pathak</author>
            <author id="81100021061" affiliation="IIT Bombay, Powai,
                Mumbai, India">Manish Gupta</author>
        </authors>
        <pubinInfo volume="20" issue="3" pages="445--470" year="2011"
            publisher="Springer-Verlag New York, Inc." title="The VLDB
            Journal"/>
    </publication>
    <publication id="2095694" gen="1" no="7">
        <title>Relational approach for shortest path discovery over
            large graphs</title>
        <citedBy count="0"/>
        <authors count="6">
            <author id="81100132510" affiliation="Peking
                University">Jun Gao</author>
            <author id="81100054574" affiliation="Kent State
                University">Ruoming Jin</author>
            <author id="81486647694" affiliation="Peking
                University">Jiashuai Zhou</author>
            <author id="81447600785" affiliation="Chinese University
                of Hong Kong">Jeffrey Xu Yu</author>
```

```
            <author id="81472650291" affiliation="Peking
                University">Xiao Jiang</author>
            <author id="81100212828" affiliation="Peking
                University">Tengjiao Wang</author>
        </authors>
        <pubinInfo volume="5" issue="4" pages="358--369" year="2011"
            publisher="VLDB Endowment" title="Proc. VLDB Endow."/>
    </publication>
    <publication id="1996798" gen="2" no="8">
        <title>On the querying for places on the mobile web</title>
        <citedBy count="0"/>
        <authors count="1">
            <author id="81331495182" affiliation="Department of
                Computer Science, Aarhus University,
                Denmark">Christian S. Jensen</author>
        </authors>
        <pubinInfo volume="none" issue="none" pages="4--4" year="2011"
            publisher="Springer-Verlag" title="Proceedings of the 13th
            Asia-Pacific web conference on Web technologies and
            applications"/>
    </publication>
    <publication id="2063641" gen="2" no="9">
        <title>Text vs. space: efficient geo-search query
            processing</title>
        <citedBy count="0"/>
        <authors count="5">
            <author id="81490688403" affiliation="Polytechnic
                Institute of NYU, Brooklyn, NY, USA">Maria
                Christoforaki</author>
            <author id="81418597661" affiliation="Polytechnic
                Institute of NYU, Brooklyn, NY, USA">Jinru
                He</author>
            <author id="81490685281" affiliation="Polytechnic
                Institute of NYU, Brooklyn, NY, USA">Constantinos
                Dimopoulos</author>
            <author id="81350598404" affiliation="University of Bonn,
                Bonn, Germany">Alexander Markowetz</author>
            <author id="81100583468" affiliation="Polytechnic
                Institute of NYU, Brooklyn, NY, USA">Torsten
                Suel</author>
        </authors>
        <pubinInfo volume="none" issue="none" pages="423--432"
            year="2011" publisher="ACM" title="Proceedings of the 20th
            ACM international conference on Information and knowledge
            management"/>
    </publication>
    <publication id="1989361" gen="2" no="10">
        <title>Reverse spatial and textual k nearest neighbor
            search</title>
        <citedBy count="0"/>
        <authors count="3">
            <author id="81100053513" affiliation="Renmin University of
                China, Beijing, China">Jiaheng Lu</author>
            <author id="81486647316" affiliation="Renmin University of
                China, Beijing, China">Ying Lu</author>
            <author id="81100468764" affiliation="Nanyang
                Technological University, Singapore, Singapore">Gao
                Cong</author>
        </authors>
```

```
        <pubinInfo volume="none" issue="none" pages="349--360"
            year="2011" publisher="ACM" title="Proceedings of the 2011
            international conference on Management of data"/>
    </publication>
    <publication id="1989363" gen="2" no="11">
        <title>Collective spatial keyword querying</title>
        <citedBy count="1">
            <id>2094001</id>
        </citedBy>
        <authors count="4">
            <author id="81447594224" affiliation="Nanyang
                Technological University, Singapore, Singapore">Xin
                Cao</author>
            <author id="81100468764" affiliation="Nanyang
                Technological University, Singapore, Singapore">Gao
                Cong</author>
            <author id="81331495182" affiliation="Aarhus University,
                Aarhus , Denmark">Christian S. Jensen</author>
            <author id="81100202401" affiliation="National University
                of Singapore, Singapore, Singapore">Beng Chin
                Ooi</author>
        </authors>
        <pubinInfo volume="none" issue="none" pages="373--384"
            year="2011" publisher="ACM" title="Proceedings of the 2011
            international conference on Management of data"/>
    </publication>
    <publication id="2159279" gen="2" no="12">
        <title>SKIF-P: a point-based indexing and ranking of web
            documents for spatial-keyword search</title>
        <citedBy count="0"/>
        <authors count="0"/>
        <pubinInfo volume="16" issue="3" pages="563--596" year="2012"
            publisher="Kluwer Academic Publishers"
            title="Geoinformatica"/>
    </publication>
    <publication id="2094001" gen="3" no="13">
        <title>Multi-approximate-keyword routing in GIS data</title>
        <citedBy count="0"/>
        <authors count="3">
            <author id="81100544467" affiliation="Shanghai JiaoTong
                University">Bin Yao</author>
            <author id="81493650862" affiliation="School of Computing,
                University of Utah">Mingwang Tang</author>
            <author id="81493647637" affiliation="School of Computing,
                University of Utah">Feifei Li</author>
        </authors>
        <pubinInfo volume="none" issue="none" pages="201--210"
            year="2011" publisher="ACM" title="Proceedings of the 19th
            ACM SIGSPATIAL International Conference on Advances in
            Geographic Information Systems"/>
    </publication>
</publications>
```

Příloha C: Uživatelská příručka

C.1 Požadavky

Ke spuštění programu je potřeba mít nainstalováno prostředí jazyka Java JRE (Java Runtime Environment) verze 1.1 nebo vyšší. Doporučená verze prostředí je 1.6. Nejnovější verzi jazyka Java naleznete na webových stránkách společnosti Oracle:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

C.2 Spuštění programu

Program je dodáván jako spustitelný JAR soubor a lze ho spustit následujícími dvěma způsoby.

C.2.1 Grafické rozhraní systému

Pokud je v operačním systému tento typ souboru asociován s interpreterem jazyka Java (měl by být automaticky po nainstalování prostředí jazyka), měl by mít soubor tradiční ikonu se šálkem, viz obr. C.1, a program by měl jít spustit poklepáním na soubor nebo stiskem klávesy ENTER.

Obr. C.1: Ikona souboru s programem při asociaci s interpreterem jazyka Java.

C.2.2 Příkazová řádka systému

Druhým způsobem je spuštění programu příkazovou řádku operačního systému. Spusťte příkazovou řádku systému nebo jeho terminál, přesuňte se v něm do adresáře s programem a následně zadejte následující příkaz:

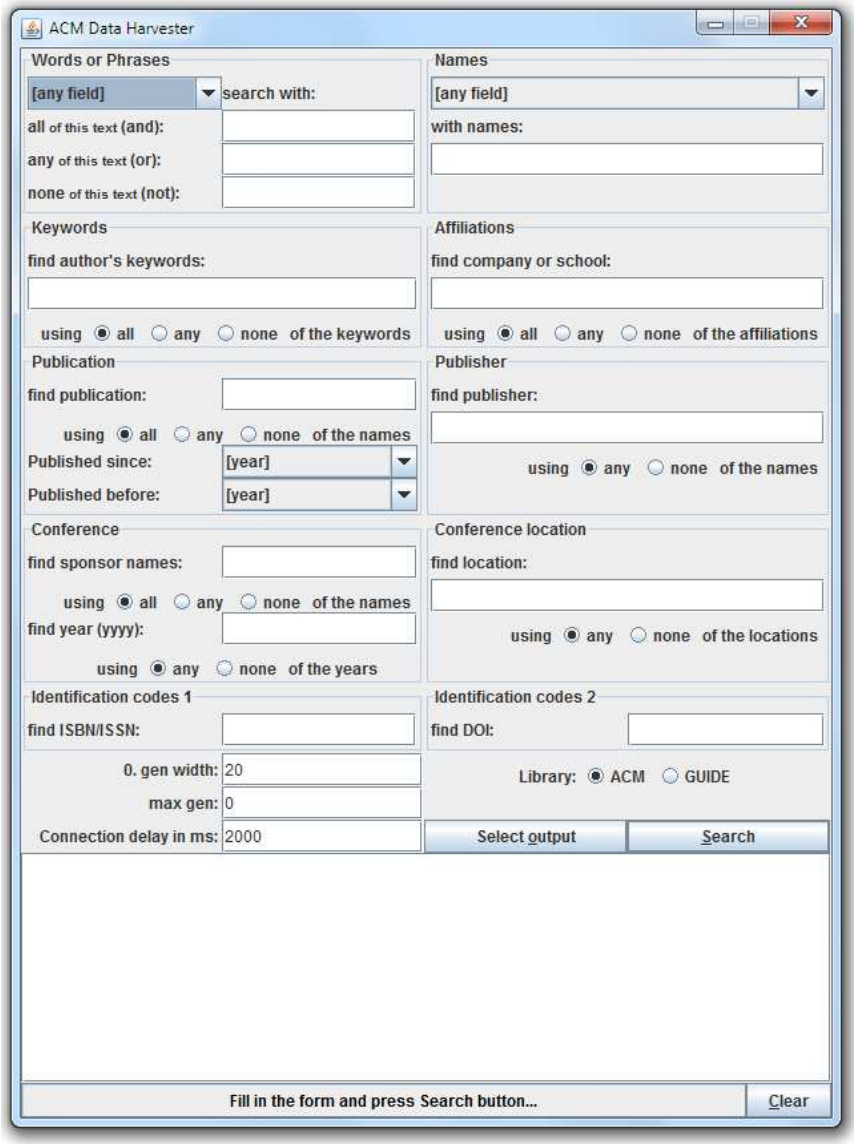
```
java -jar Krupicka_Jan_A09B0309P_ACM_Data_Harvester.jar
```

V případě, že byl program přejmenován, zadejte místo poslední části příkazu název aktuální.

Tímto způsobem spuštění také získáte přístup k průběžným výpisům programu, které informují o stavu programu přesněji a obsáhleji než je tomu v grafickém uživatelském rozhraní.

C.3 Okno aplikace

Po spuštění programu se vám zobrazí okno jeho grafického rozhraní, viz obr. C.2.



Obr. C.2: Okno grafického rozhraní aplikace po spuštění programu.

C.3.1 Vyhledávací formulář

Horní více než polovinu okna zabírá vyhledávací formulář. Do něj dle možností jednotlivých prvků vyplňte informace o vámi vyhledávaných publikacích.

C.3.2 Panel s nastavením

Mezi formulářem a (zatím prázdnou) textovou oblastí se nachází panel s nastavením. Obsahuje čtyři volby a dvě tlačítka, vše vysvětleno níže.

Nastavení "0. gen width" určuje počet publikací, který se má vyhledat v nulté generaci. Všechny publikace, které citují publikace z této generace, budou zpracovávány v generaci následující - první.

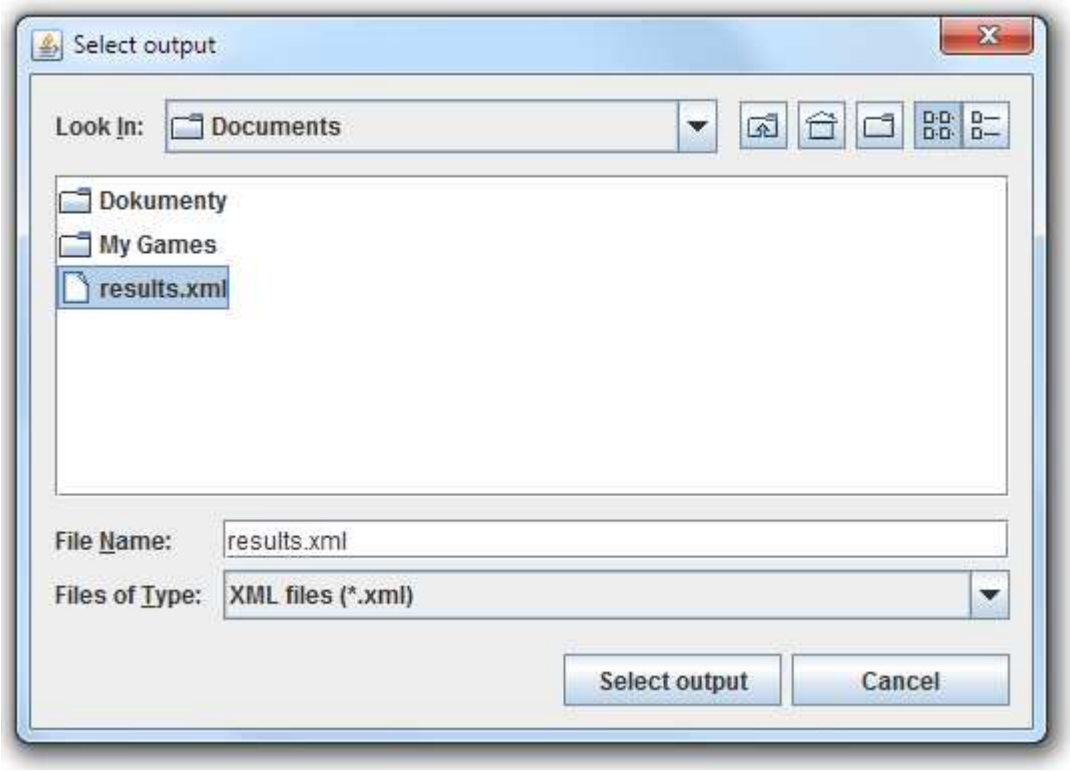
Nastavení "max.gen" označuje číslo generace vyhledávacího procesu, která se má zpracovat naposledy. Pokud tedy proces dospěje k této generaci a existuje generace následující, aplikace vyhledávání ukonči a nebude pokračovat dále. Za předpokladu, že neexistuje dostatek generací k dosažení takto specifikované generace, vyhledávání skončí dříve.

Nastavení "Connection delay in ms" určuje prodlevu mezi dvěma po sobě jdoucími připojeními k serveru služby ACM DL. Toto nastavení je přítomno proto, aby se zamezilo nepřiměřenému zatížení serveru na úkor ostatních uživatelů. Z tohoto důvodu je doporučováno mít vždy nastaveno alespoň 1000 milisekund.

Nastavení "Library" nabízí výběr dvou knihoven, ve kterých lze publikace vyhledávat. Volba "ACM" označuje knihovnu ACM Digital Library, která obsahuje kompletní informace o publikacích. Volba "GUIDE" označuje knihovnu GUIDE, která je oproti ACM DL rozšířena o strohé citační informace a abstrakty.

Tlačítko "Select output" umožňuje vybrat soubor, do kterého se budou získané informace ukládat. Výběr probíhá prostřednictvím dialogu (viz obr. C.3), ve kterém zvolte nejprve cílový adresář a poté požadovaný výstupní soubor. K vytvoření nového souboru zadejte jeho jméno do řádky "File Name". Nakonec vše potvrďte.

Tlačítko "Search" spustí vyhledávání publikací, které vyhovují hodnotám vyplněným ve formuláři. Pro předběžné ukončení vyhledávání stiskněte znovu toto tlačítko, nyní již přejmenované na "Stop".



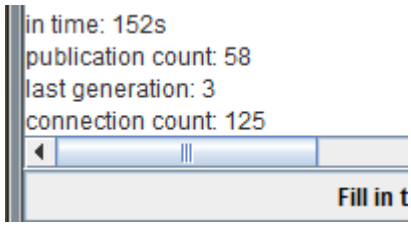
Obr. C.3: Dialog výběru výstupního souboru.

C.3.3 Textová oblast

Pod panelem s nastavením se nachází textová oblast. Do té se v průběhu vyhledávání vypisují jednotlivé URL adresy, ze kterých se aktuálně získávají informace pro zpracování. Text lze mazat pomocí tlačítka "Clear", které se nachází ve spodním pravém rohu okna aplikace (viz obr. C.5).

Po skončení vyhledávání se zde objeví koncová statistika tohoto hledání (viz obr. C.4), která obsahuje čtyři údaje:

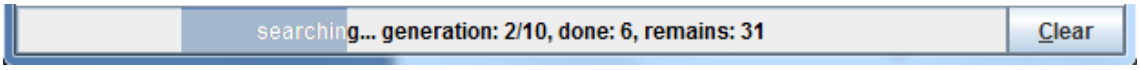
- in time - uvádí čas v sekundách, po který vyhledávání probíhalo
- publication count - informuje o počtu zpracovaných publikací
- last generation - udává číslo poslední zpracovávané generace
- connection count - uvádí celkový počet připojení k serveru služby ACM



Obr. C.4: Koncová statistika po ukončení vyhledávání.

C.3.4 Stavový řádek

Zcela vespod okna aplikace se nachází její stavový řádek, jenž je vidět i na následujícím obrázku C.5.



Obr C.5: Stavový řádek aplikace.

Během vyhledávání se po řádku přesouvá tmavý pruh, který indikuje činnost aplikace. Řádek také informuje o třech údajích:

- generation - aktuální a maximální zpracovávaná generace
- done - počet úspěšně zpracovaných publikací
- remains - počet dosud nalezených publikací, které teprve budou zpracovány

C.4 Ukončení programu

Program se ukončuje pomocí standardního ukončovacího tlačítka, které nabízí používaný operační systém. Většinou se toto tlačítko nachází v pravém horním rohu okna v liště aplikace. Nejčastěji je označeno křížkem a červenou barvou.

Pokud probíhá vyhledávání, je doporučeno ho nejprve zastavit tlačítkem "Stop" a teprve potom aplikaci ukončit výše uvedeným způsobem. Tento postup zaručí korektně ukončený a tudíž validní XML výstup ve zvoleném souboru s výsledky aplikace.