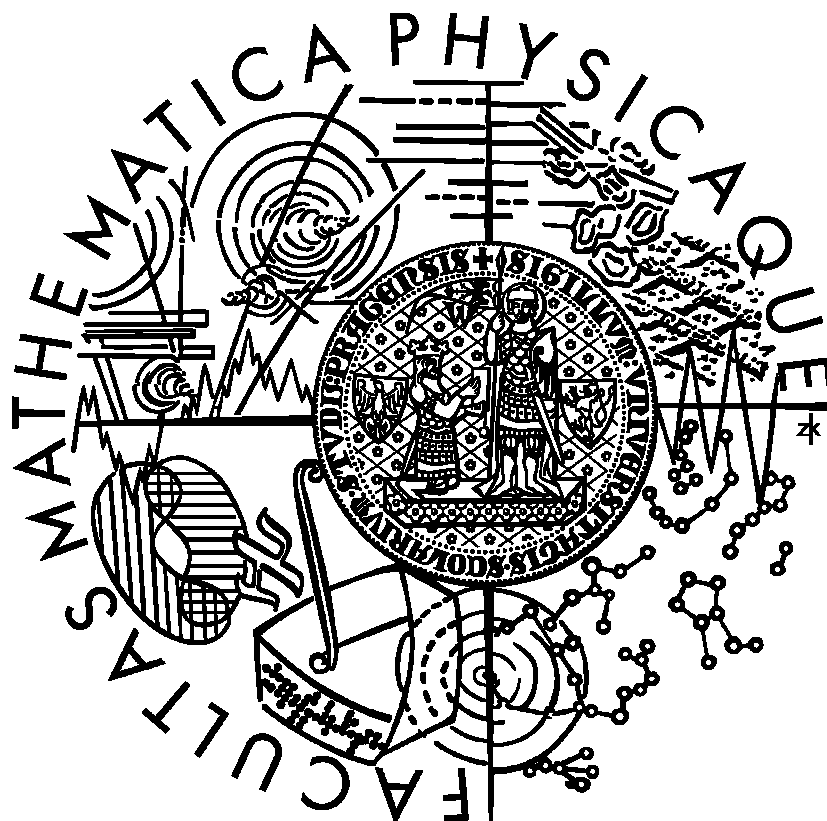


Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Tomáš Smotrila
Automatizovaná extrakcia informácií z internetu
Kabinet software a výuky informatiky
Vedoucí diplomové práce: *RNDr. František Mráz, CSc.*
Studijní program: *Informatika, softwarové systémy, softwarové inženýrství*

Ďakujem vedúcemu práce RNDr. Františkovi Mrázovi, CSc. za motiváciu, cenné rady a usmernenia počas písania tejto diplomovej práce.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne, 15.4.2011

Bc. Tomáš Smotrila

Název práce: *Automatizovaná extrakcia informácií z internetu*

Autor: *Bc. Tomáš Smotrila*

Katedra (ústav): *Kabinet software a výuky informatiky*

Vedoucí diplomové práce: *RNDr. František Mráz, CSc.*

e-mail vedoucího: *Frantisek.Mraz@mff.cuni.cz*

Abstrakt: Webové stránky sprístupňujú ohromné množstvo informácií. Často sa jedná o stránky generované z dát uložených v databázach. Pritom je kladený dôraz na zobrazenie informácií, ale nie na ich strojové spracovanie. V diplomovej práci je navrhnutý a implementovaný prototyp systému na získavanie dát z dynamicky generovaných webových stránok technikou programovania demonštráciou (programming by demonstration). Takýto systém umožní užívateľovi myšou ukázať, ako má systém postupovať pri zbieraní informácií z danej webovej stránky. Na základe takejto ukážky by systém mal odvodiť postup, ako získavať informácie na podobných stránkach. Implementovaný systém dokáže pozbierať pre užívateľa relevantné informácie z podobných stránok napríklad do tvaru jednoduchej tabuľky vhodnej na ďalšie strojové spracovanie.

Klíčová slova: extrakce dat z Webu, programování demonstrací

Title: *Automated web information extraction*

Author: *Bc. Tomáš Smotrila*

Department: Department of Software and Computer Science Education

Supervisor: *RNDr. František Mráz, CSc.*

Supervisor's e-mail address: *Frantisek.Mraz@mff.cuni.cz*

Abstract: Web sites offer a huge amount of information. Often it is a page generated from data stored in databases. However, emphasis is placed on the display of information, but not on their machine processing. Part of the thesis is design and implementation of a prototype system to retrieve data from dynamically generated web using programming by demonstration technique. Such a system allows the user to show with mouse to the system how to proceed with gathering information from the website. Based on such a example, the system will derive a procedure to acquire information on similar sites. The implemented system is able to collect user relevant information from similar sites for example in form of a simple table suitable for further machine processing.

Keywords: Web Scraping, Programming by Demonstration

Obsah

1	Úvod.....	- 1 -
1.1	Motivácia	- 1 -
1.1.1	Motivačná úloha č.1	- 1 -
1.1.2	Motivačná úloha č.2	- 1 -
1.1.3	Motivačná úloha č.3	- 2 -
1.2	Ciele práce	- 2 -
1.3	Štruktúra práce.....	- 2 -
1.4	Výstupy práce	- 4 -
2	WDE — Extrakcia dát z webu	- 5 -
2.1	Extrakcia dát všeobecne	- 5 -
2.2	WDE — Extrakcia dát z webu.....	- 6 -
2.2.1	Využitie	- 6 -
2.2.2	Metódy extrakcie dát z webu.....	- 7 -
2.2.3	Nástroje využívané v praxi.....	- 8 -
2.2.4	Právne hľadisko.....	- 9 -
2.2.5	Metódy ochrany pred extrakciou dát z webu	- 10 -
3	Programovanie demonštráciou.....	- 11 -
3.1	Programovanie koncovým užívateľom.....	- 11 -
3.2	Programovanie demonštráciou a jeho využitie.....	- 12 -
3.3	Nástroje PBD a typy techník používané v praxi.....	- 13 -
4	Analýza	- 15 -
4.1	Definícia systému WDE-PBD	- 15 -
4.1.1	Popis činnosti	- 15 -
4.1.2	Obmedzenia na implementovaný typ WDE.....	- 16 -
4.1.3	Voľba vhodnej techniky PBD na účely WDE.....	- 17 -
4.1.4	Zhrnutie	- 18 -
4.2	Definície základných pojmov	- 19 -
4.2.1	Obecné pojmy	- 19 -

4.2.2	Programovanie demonštráciou	- 21 -
4.2.3	Extrakcia dát.....	- 22 -
4.2.4	Extrakcia URL.....	- 22 -
4.3	Základné schémy algoritmov.....	- 26 -
4.3.1	Schéma techniky PBD.....	- 26 -
4.3.2	Schéma algoritmu extrakcie dát	- 27 -
4.3.3	Schéma algoritmu extrakcie URL adries.....	- 28 -
4.4	Analýza problematiky WDE	- 30 -
4.4.1	Typy webov a techniky generovania HTML dokumentov	- 30 -
4.4.2	Kompozícia webu a stránky	- 31 -
4.4.3	Navigácia užívateľa k dátam.....	- 32 -
4.4.4	Formy prezentácie dát	- 33 -
5	Softvérová analýza.....	- 38 -
5.1	Popis systému	- 38 -
5.2	Požiadavky na systém.....	- 38 -
5.3	Štúdia uskutočniteľnosti	- 39 -
5.3.1	Voľba vhodného prostredia.....	- 39 -
6	Algoritmy a dátové štruktúry	- 41 -
6.1	PBD	- 41 -
6.1.1	Návrh algoritmu	- 41 -
6.1.2	Dátové štruktúry algoritmu PBD.....	- 43 -
6.1.3	Algoritmus PBD.....	- 44 -
6.2	DE-PBD.....	- 45 -
6.2.1	Návrh algoritmu	- 45 -
6.2.2	Dátové štruktúry algoritmu DE-PBD	- 50 -
6.2.3	Algoritmus DE-PBD	- 52 -
6.3	UE-PBD.....	- 54 -
6.3.1	Návrh algoritmu	- 54 -
6.3.2	Dátové štruktúry algoritmu UE-PBD	- 58 -
6.3.3	Algoritmus UE-PBD	- 60 -
6.3.4	Algoritmus Zisti stránkovaciu oblasť.....	- 61 -

6.3.5	Algoritmus Zisti náhľadovú oblasť	- 62 -
7	Implementácia.....	- 63 -
7.1	Popis systému	- 63 -
7.2	Členenie systému	- 63 -
7.2.1	Jadro	- 63 -
7.2.2	Vlastné Komponenty	- 63 -
7.2.3	Okienková aplikácia	- 64 -
7.3	Obmedzenia	- 65 -
7.4	Rozšíriteľnosť	- 66 -
8	Testovacie scenáre	- 67 -
8.1	Testovací scenár č. 1.....	- 68 -
8.2	Testovací scenár č. 2.....	- 69 -
8.3	Testovací scenár č. 3.....	- 71 -
8.4	Testovací scenár č. 4.....	- 72 -
8.5	Zhrnutie výsledkov	- 73 -
9	Záver	- 75 -
10	Zoznam použitej literatúry	- 77 -
Príloha A.	Web Data Extractor - Užívateľská príručka.....	- 79 -
11	Úvod.....	- 79 -
11.1	Účel programu Web Data Extractor	- 79 -
11.2	Základné pojmy	- 79 -
11.3	Základné workflow programu	- 80 -
11.4	Popis hlavného okna programu	- 81 -
12	Popis funkcií.....	- 83 -
12.1	Manipulácia so Solution	- 83 -
12.2	Získanie URL adries ručne	- 84 -
12.2.1	Pridanie URL do zoznamu	- 84 -

12.2.2	Odobratie vybranej URL zo zoznamu.....	- 85 -
12.2.3	Zmazanie celého zoznamu URL	- 85 -
12.2.4	Preusporiadanie poradia v URL zozname	- 85 -
12.2.5	Získanie zoznamu URL zo automatizovaného zdroja.....	- 85 -
12.2.6	Uloženie zoznamu pracovných URL do solution	- 85 -
12.3	Automatizované získavanie URL adries	- 86 -
12.3.1	Špecifikovanie zdroja URL	- 86 -
12.3.2	Získanie zoznamu URL adries zo zadaného zdroja	- 87 -
12.3.3	Špecifikovanie nastavení po získaní URL	- 87 -
12.4	URL agent.....	- 88 -
12.4.1	Učenie URL Agentu	- 90 -
12.4.2	Spustenie URL Agentu	- 91 -
12.5	Data Agent	- 92 -
12.5.1	Učenie Data Agentu	- 93 -
12.5.2	Spustenie Data Agentu	- 95 -
12.6	Uloženie extrahovaných dát	- 96 -
12.6.1	Základné nastavenia výstupu	- 96 -
12.6.2	Nastavenia výstupu v CSV formáte	- 97 -
12.6.3	Nastavenia výstupu v SQL formáte	- 98 -
12.7	Spustenie celého Solution.....	- 98 -
Príloha B.	CD nosič.....	- 100 -

1 Úvod

1.1 Motivácia

V dnešnej dobe pokročilej informatizácie, nielen obrovského rozmachu Internetu, ale aj jeho vysokej dostupnosti a napredovania vyhľadávacích strojov sa informácie stávajú omnoho jednoduchšie dosiahnuteľnými. Pred pár desaťročiami, keď bol Internet iba malou sieťou prepojených univerzitných počítačov, bolo pre bežného človeka mnohokrát potrebné enormné úsilie na získanie informácie. Dnes je drvivá väčšina informácií o čomkoľvek na Internete prístupná každému jeho používateľovi bezplatne, ak opomenieme poplatok za pripojenie do Internetu.

Informácie sú na Webe prezentované prostredníctvom jazyka HTML, čo plne vyhovuje nárokom bežného užívateľa. Na pokročilejšie dotazovanie, či systematické získavanie informácií je však tento jazyk spolu s moderným dizajnom webových stránok nevhodný. U náročnejších užívateľov a komerčných firiem však tieto činnosti môžu priniesť veľký úžitok. Vzhľadom na veľké množstvo informácií je nutné vyvinúť automatizovaný postup ich extrakcie. Ak zoberieme do úvahy vysokú dostupnosť týchto informácií, ich nízku obstarávaciu cenu a možnosť strojovo ich získať a neskôr dotazovať, dostávame tak systém, ktorý je schopný produkovať cenné informácie a poskytnúť tak jeho užívateľovi konkurenčné výhody.

1.1.1 Motivačná úloha č.1

Ako človek zanietený do techniky by som si rád vybral a kúpil ten najlepší fotoaparát. Z webovej stránky predajcu elektroniky chcem preto získať čo najviac informácií o parametroch jednotlivých fotoaparátov, samozrejme cenu a poprípadе užívateľské hodnotenie. Tieto dáta potom vyhodnotím napríklad v tabuľkovom editore.

1.1.2 Motivačná úloha č.2

Komerčná firma ponúkajúca služby chce získať záznamy o firmách, ktoré by potenciálne mohli byť jej novým obchodným partnerom resp. klientom. Tieto dáta je potrebné importovať do internej databázy firmy za účelom neskoršieho kontaktovania prostredníctvom call centra a doplnenia ďalších údajov.

1.1.3 Motivačná úloha č.3

Inštitúcia, zaoberajúca sa investovaním do cenných papierov na burze, potrebuje získať rôzne dáta, ktoré môžu mať vplyv na vývoj ceny akcií. Na tieto dáta plánuje následne použiť data mining alebo inú sofistikovanú techniku, ktorá by umožnila predpovedanie budúceho vývoja ceny akcií.

1.2 Ciele práce

Hlavným cieľom diplomovej práce je navrhnuť a implementovať systém na získavanie dát z dynamicky generovaných webových stránok technikou programovania demonštráciou (programming by demonstration). Takýto systém by mal umožniť užívateľovi za pomoci myši ukázať, ako má systém postupovať pri zbieraní informácií z danej webovej stránky. Na základe takejto ukážky by systém mal odvodiť postup, ako získavať informácie na podobných stránkach. Implementovaný systém by mal dokázať pozbierať pre užívateľa relevantné informácie z podobných stránok napríklad do tvaru jednoduchšej tabuľky vhodnej na ďalšie strojové spracovanie.

Diplomová práca by medzi iným mala taktiež nájsť odpovede na nasledujúce otázky:

1. Dá sa implementovať uvedený systém za použitia techniky programovania demonštráciou tak, aby bol použiteľný v praxi?
2. Ak áno, aké sú jeho výhody a nevýhody oproti ostatným nástrojom venujúcim sa extrakcii dát z Internetu?
3. Ak nie, aké sú hlavné obmedzenia vyplývajúce z použitia tejto techniky?

Podružným cieľom práce je snaha vyvinúť systém tak, aby svojou funkcionalitou splňal základné požiadavky a pokiaľ možno čo najviac sa približoval ideálnym požiadavkám na neho kladeným, ktoré sú definované v časti **5.2 Požiadavky na systém**.

1.3 Štruktúra práce

Kapitola **2 WDE — Extrakcia dát** z webu poskytuje úvod do tejto problematiky. Sú v nej definované pojmy ako **extrakcia dát** a **extrakcia dát z webu**. Ďalej sa tu uvádzajú techniky jej blízke, ako je napríklad **indexovanie**, praktické možnosti jej využitia, ako aj dôvody, prečo je vlastne extrakcia dát z webu pomerne zaujímavým oborom. Súčasťou kapitoly je zoznam a rozčlenenie metód, využívaných pri extrakcii, na ktorú tesne nadväzuje analýza existujúcich nástrojov a rozbor, aké metódy pre svoju činnosť využívajú. Na záver kapitoly sú uvedené záporné stránky extrakcie dát z webu nasledované vymenovaním možností chránenia sa pred neželanou extrakciou dát z webu.

Popísať techniku **programovania demonštráciou** si kladie za cieľ kapitola **3 Programovanie demonštráciou**, kde sú najprv priblížené niektoré základné pojmy používané v tejto oblasti ako napríklad **programovanie v užívateľskom rozhraní** či **popis dát**. Kapitola ďalej skúma, aké prístupy je možné zvoliť pri riešení úloh programovaním demonštráciou.

Kapitola **4 Analýza** je analytickou časťou tejto diplomovej práce. Ako podklady pre jej vypracovanie slúžili predošlé dve kapitoly **2 WDE – Extrakcia dát z webu**, **3 Programovanie demonštráciou** a vlastný výskum vychádzajúci z praktických skúseností nadobudnutých pri vývoji webových stránok. Analýza sa venuje hĺbkovému štúdiu prezentovania dát užívateľovi na Internete. Z nej vyplynuli skutočnosti a predpoklady, o ktoré sa opiera návrh a konkrétna implementácia algoritmov a dátových štruktúr, ktoré sú bližšie popísané v kapitole **6 Algoritmy a dátové štruktúry**.

V kapitole **4 Analýza** sú uvedené definície, ktoré používam taktiež v nasledujúcich kapitolách, a preto sa pre ich dostatočné pochopenie neodporúča túto kapitolu vynechať. Jej podkapitoly na seba vždy nadväzujú, a preto by mali byť čítané v poradí, v akom za sebou nasledujú.

V úvode analýzy je uvedený základný postup, akým systém, implementovaný v tejto práci, extrahuje dáta z webu. Nasleduje rozbor obmedzení implementácie a výber vhodnej techniky programovania demonštráciou, ktorá je aplikovaná pri extrahovaní dát z webu. Ďalej v **4.3.2 Schéma algoritmu extrakcie dát** je popísaná schéma primárneho algoritmu vyvíjaného v tejto práci a v podkapitole **4.3.3 Schéma algoritmu extrakcie URL adries** schéma sekundárne vyvíjaného algoritmu.

Kapitola **5 Softvérová analýza** sa zaoberá analýzou z hľadiska vývoja softvérového diela. Zo štandardnej analýzy sú v nej zahrnuté: popis systému, definícia stupňov požiadaviek na neho kladených a štúdia uskutočniteľnosti. Nakoľko charakter tejto práce je z veľkej časti experimentálny, nie je v tejto kapitole uvedená časť prípadov použitia, detailné rozpracovanie funkčných a nefunkčných požiadaviek ani UML diagramy popisujúce systém.

Detailnejším návrhom a konkrétnou implementáciou algoritmov navrhnutých v kapitole **4 Analýza** sa venuje kapitola **6 Algoritmy a dátové štruktúry**. Táto kapitola sa rozpadá na tri časti podľa algoritmov vyvinutých v tejto práci, a to na podkapitolu **6.1 PBD** popisujúcu implementáciu techniky programovania demonštráciou, podkapitolu **6.2 DE-PBD**, v ktorej je rozoberaný algoritmus extrakcie dát technikou programovania demonštráciou a na kapitolu **6.3 UE-PBD**, ktorá bližšie popisuje algoritmus extrakcie URL adries technikou programovania demonštráciou.

Podrobnostiam týkajúcim sa implementácie vyvinutého systému WDE-PBD sa v krátkosti venuje kapitola **7 Implementácia**, ktorá obsahuje popis a členenie systému z hľadiska zvolenej architektúry. Na záver uvádza, ktoré body návrhu neboli implementované resp. sa od nich implementácia odklonila a popisuje možnosti ďalšieho rozšírenia systému.

Kapitola **8 Testovacie scenáre** uvádza výsledky testovania programu snaží sa overiť schopnosť jeho použitia v reálnych prípadoch pomocou testovacích scenárov.

1.4 Výstupy práce

Hlavným výstupom práce je program umožňujúci automatizovanú extrakciu dát z webu nachádzajúci sa na priloženom CD, ktorého obsah je popísaný v **Príloha B. CD nosič**.

2 WDE – Extrakcia dát z webu

Táto časť je úvodom do problematiky extrakcie dát z webu a slúži ako podklad pre kapitolu 4 **Analyza**. Jej prvá časť **2.1 Extrakcia dát všeobecne**, čerpá informácie z [1]. V úvode definuje pojem extrakcie dát, ďalej uvádza jej obecný popis a na záver vymenúva existujúce obory extrakcie dát.

Druhá časť tejto kapitoly **2.2 WDE – Extrakcia dát z webu** na úvod uvádza jej definíciu a popisuje techniky jej blízke z [2]. Ďalej v časti **2.2.1. Využitie** sú uvedené praktické možnosti využitia extrakcie dát z webu, ako aj dôvody, prečo je vlastne extrakcia dát z webu pomerne zaujímavým oborom.

Časť **2.2.2 Metódy extrakcie dát z webu** poskytuje zoznam a rozčlenenie metód, využívaných na extrakciu dát z webu, na ktorú tesne nadväzuje časť **2.2.3 Nástroje využívané v praxi**, kde je uvedená analýza existujúcich nástrojov a rozbor, aké metódy pre svoju činnosť využívajú.

Na záver kapitoly sú uvedené záporné stránky extrakcie dát z webu v časti **2.2.4 Právne hľadisko** nasledované vymenovaním možností chránenia sa pred neželanou extrakciou dát z webu v časti **2.2.5 Metódy ochrany pred extrakciou dát z webu**.

2.1 Extrakcia dát všeobecne

Extrakcia dát je technika, v ktorej počítačový program extrahuje dáta z človeku určenému výstupu, ktorý pochádza z iného programu.

Zvyčajne je výmena dát medzi dvomi programami realizovaná použitím dátových štruktúr vhodných na automatizované spracovanie počítačmi, nie ľuďmi. Takéto výmenné formáty a protokoly sú typicky striktné štruktúrované, dobre zdokumentované, jednoducho parsovateľné (rozložiteľné), a majú minimálnu nejednoznačnosť. Veľmi často sú tieto formáty pre človeka úplne nečitateľné.

Kľúčový faktor, ktorý odlišuje extrakciu dát od bežného parsovania je ten, že extrahované dáta boli určené na zobrazenie koncovému používateľovi, a nie ako vstup pre iný program. Preto takéto dáta nie sú typicky takmer vôbec zdokumentované, nie to ešte štruktúrované na pohodlné parsovanie. Extrakcia dát často zahŕňa ignorovanie binárnych dát (typicky obrázky a multimédiá), formátovanie zobrazenia, nadbytočné nápisy, prebytočné komentáre a iné informácie, ktoré sú buď irelevantné alebo obmedzujú automatizované spracovanie.

Extrakcia dát je najčastejšie realizovaná buď oproti rozhraniu informačného systému, ktorý nemá žiadny iný mechanizmus kompatibilný s aktuálnym hardvérom, alebo oproti rozhraniu systémov tretích strán, ktoré neponúkajú pohodlnejšie API na tento účel. V druhom prípade správca takéhoto systému môže považovať takúto extrakciu za neželanú, kvôli možnému zvýšeniu záťaže, strate príjmov z reklamy, alebo strate kontroly nad obsahom informácií.

Technika extrakcie dát je všeobecne považovaná ako prvoplánová, neelegantná technika, často používaná ako posledná možnosť, kedy už nie sú dostupné iné mechanizmy. Na rozdiel od programovania a strojového spracovania, výstupy určené ľuďom často menia svoju štruktúru. Ľudia sa s týmto problémom dokážu jednoducho vysporiadať, avšak počítačové programy často spadnú alebo vyprodukujú nesprávne výsledky.

Medzi oblasti extrakcie dát z webu patrí:

- **OCR** (z anglického **Optical Character Recognition**) je metóda, pri ktorej sa počítačový program snaží rozpoznať písmo na základe predloženého obrázku v elektronickej podobe.
- **Extrakcia dát z webu** sa venuje získavaniu dát z HTML dokumentov dostupných prostredníctvom Internetu.
- **Report Mining** je dolovanie znalostí z textov, ktoré môžu byť jednoduchými textovými súborami alebo uložené v PDF, HTML či inom formáte.

2.2 WDE – Extrakcia dát z webu

Extrakcia dát z webu (**WDE** z anglického **Web Data Extraction**, taktiež zvaná **Web Harvesting** alebo aj **Web Scraping**) je softvérová technika, v ktorej sa automatizovane extrahujú informácie z webových stránok. Program, ktorý vykonáva extrakciu dát z webu sa označuje ako **wrapper**. Softvéry, pomocou ktorých je možné vytvoriť wrappery, zvyčajne implementujú nízkoúrovňový Hypertext Transfer Protocol alebo zabudovávajú do seba plnohodnotné internetové prehliadače ako je Internet Explorer či Mozilla Firefox [2].

Extrakcia dát z webu je podobná technika ako indexovanie webu, v ktorom sa použitím robota prechádzajúceho HTML stránky indexuje obsah webu. Rozdiel medzi indexáciou webu a extrakciou dát z webu je ten, že indexácia webu iba indexuje obsah stránky, kdežto pri extrakcii dát z webu sa spracúva obsah webu, typicky z HTML formátu na štruktúrované dáta, ktoré môžu byť uložené a analyzované v databázovom systéme alebo tabuľkovom editore. Extrakcia dát z webu je taktiež blízka webovej automatizácii, ktorá simuluje prehliadanie internetu človekom [2].

2.2.1 Využitie

Medzi najdôležitejšiu oblasť použitia WDE patrí on-line monitoring dát ako napríklad: porovnávanie cien výrobkov, monitorovanie údajov o počasí, získanie finančných dát, sledovanie realitného trhu, či aukcií. V týchto prípadoch hrá čas významnú rolu a použitie automatizovanej techniky WDE môže poskytnúť jej používateľovi nesmiernu výhodu pred ostatnými užívateľmi, či konkurenciou.

Ďalšou zaujímavou oblasťou využitia techniky WDE sú hybridné webové stránky (anglicky Mashups), ktoré integrujú dáta. Tieto weby majú vysokú pridanú hodnotu v zmysle úplnosti ponúkaných informácií, ktoré sú inak roztrúsené na desiatkach, niekedy aj stovkách iných webov. Zozbieranie takýchto informácií ručne by predstavovalo pre bežného používateľa webu niekedy až nadľudské úsilie.

Domény integrovaných dát sú rôzne, od stránok, ktoré ponúkajú možnosť vyhľadať si denné menu na konkrétny deň v užívateľom zadanej lokalite (napr. www.lunchtime.cz), až po web, ktorý poskytuje užívateľom on-line porovnávanie cien výrobkov s ich detailným popisom (napr. www.heureka.cz). Práve vďaka zaujímavosti domény integrovaných dát vďaka heureka.cz obrovskej návštevnosti až 24 miliónom vzhliadnutí za august 2010 [3], čo ju suverénne zaradzuje medzi najlukratívnejšie weby nielen u jej poskytovateľa reklamy.

2.2.2 Metódy extrakcie dát z webu

Extrakcia dát z webu je obor, ktorý má rovnaký cieľ ako sémantický web. Sémantický web je podľa [2] považovaný za ambicióznú iniciatívu, ktorá stále potrebuje prelomové riešenia v spracovaní textu, sémantickom porozumení, umelej inteligencii a interakcii medzi človekom a počítačom. Extrakcia dát z webu miesto toho uprednostňuje riešenia na základe existujúcich technológií, ktoré sú často ad hoc riešeniami. Preto je mnoho rôznych úrovní automatizácie, ktorú ponúkajú technológie extrakcie dát z webu:

1. Manuálne kopírovanie a vkladanie [2]

Najjednoduchšia technika na extrakciu dát z webu je manuálne kopírovanie a vkladanie dát človekom. Podľa [2] v niektorých prípadoch ani najlepšie systémy na extrakciu dát z webu nenahradia manuálne preskúmanie dát človekom a použitie techniky kopíruj a vlož. Niekedy môže byť tento spôsob jediné fungujúce riešenie, keďže na extrahovaných webových stránkach sa môžu objaviť explicitne vytvorené bariéry, aby sa zabránilo ich strojovému spracovaniu [2].

2. Jednorazové softvérové riešenia

V mnohých prípadoch sa za účelom získania dát z webu použije jednorazový skript, prispôbený na konkrétne webové stránky a ich štruktúru. Takéto skripty sú v obmedzenej miere znovu použiteľné, a to prispôbením už existujúceho funkčného skriptu. Tieto skripty môžeme deliť podľa technológie ktorú využívajú:

a. Grep a regulárne výrazy

Jednoduchý a overený prístup k extrakcii dát z webu je založený na využití unixovského príkazu grep, resp. funkcií na hľadanie výskytu textu pomocou regulárnych výrazov [2]. Medzi často využívané jazyky v tejto skupine patria Perl, Python a PHP.

b. DOM parser

Jedným z použití DOM (Document Object Model) rozhrania je získavanie dát z dokumentov v HTML. Ide o dobre definované API použiteľné na mnohých platformách a v rozličných jazykoch, umožňujúce programátorom nielen

pristupovať k HTML dokumentu ako ku stromu, ale podporuje aj selektívnu funkcionálnosť ako napríklad výber uzlu špecifikovaním jeho identifikátora, či HTML značky.

c. HTML parser

Na parsovanie a extrakciu semi-štruktúrovaných dát z HTML dokumentov sú taktiež vhodné technológie XML a to predovšetkým dotazovací jazyky XPath a XQuery, ktoré umožňujú výber množiny uzlov a jazyk XSLT umožňujúci transformovať HTML stránku na výsledné dáta.

3. Softvér na extrakciu dát z webu

Základnou charakteristikou týchto systémov je, že už nie sú ad hoc, ale vyznačujú sa vysokou znovupoužitelnosťou. Na svoj cieľ okrem vyššie spomínaných technológií využívajú nejakú sofistikovanejšiu metódu, či vlastný programovací jazyk. Ďalšou charakteristikou týchto nástrojov je možnosť ukladať dáta do lokálnych databáz či tabuľkových editorov.

a. Softvér používajúci ustálené technológie

Na trhu existuje mnoho nástrojov na vytvorenie konfigurovateľných riešení získavania dát z webu za použitia vyššie uvedených technológií a nejakej vlastnej logiky. Užívateľ takýchto systémov väčšinou musí vytvoriť nejakú konfiguráciu, či skript, ktorý poskytuje inštrukcie pre pod ním položenú vrstvu technológií na získavanie informácií. Niektoré nástroje ponúkajú nahrávacie rozhranie [2] ako náhradu za takéto skripty a snažia sa tak vytvoriť robota. Ako technológie na jeho vytvorenie im môžu slúžiť neurónové siete, či strojové učenie.

b. Softvér využívajúci sémantické rozpoznávanie ako je citované na Wikipedii [2]: "Webové stránky môžu zahrnúť sémantické značky a anotácie do svojho obsahu, čo umožní lokalizovať dáta v nich obsiahnuté. V prípade, že stránky do svojho obsahu zahrnú aj anotácie, využíva tento softvér technológie DOM či HTML parsovania. V opačnom prípade sú anotácie organizované do sémantickej vrstvy a sú uložené oddelene od webových stránok, takže tieto systémy môžu získať schému dát a inštrukcie z tejto vrstvy predtým, než začnú spracovávať samotné stránky."

2.2.3 Nástroje využívané v praxi

Pretože WDE je pomerne novou činnosťou, ktorá s rozmachom Internetu nadobúda stále väčšiu dôležitosť, existuje v súčasnosti na trhu veľké množstvo nástrojov, ktoré dokážu extrahovať dáta z webu.

Tie menej kvalitné ponúkajú jednoduché a prvoplánové možnosti extrakcie dát, ako napríklad výber HTML elementov rovnakého typu zadaním XPath cesty (napr. Dattex).

Ďalším faktorom brániacim širšiemu využitiu nástrojov WDE, ktoré už ponúkajú pokročilejšie možnosti extrakcie dát, je, že vyžadujú aspoň základné znalosti programovania (napr. Web Data Extractor, či bezplatný Web-harvest). K ich nevýhodám patrí taktiež

neschopnosť wrapperov vytvorených za pomoci týchto nástrojov vysporiadať sa s malými zmenami v štruktúre HTML stránky.

V sekcii sú ďalej rozoberané profesionálne, kvalitné a v praxi uprednostňované nástroje. Ich získanie či používanie je veľmi drahé (Unit Miner až 1000\$ [4]). Len jeden z uvedených je dostupný v trial verzii, čo značne sťažuje širšiu analýzu kvalitných riešení WDE.

UnitMiner

Použitie nástroja UnitMiner vyžaduje znalosť programovacieho jazyka PHP. Užívateľ v ňom pomocou definovaného frameworku a sady funkcií je schopný vytvoriť wrappery, ktoré sú podľa [4] schopné vyrovnat' sa s malými zmenami v štruktúre spracovávaných HTML dokumentov.

Lixto

Lixto vznikol na akademickej pôde rakúskej Technickej Univerzity vo Viedni a talianskej Kalábrijskej Univerzity [5]. Lixto je systém a metóda pre vizuálne a interaktívne generovanie wrapperov pre webové stránky pod dohľadom človeka-vývojára [5]. Slúži na automatickú extrakciu informácie z webu použitím týchto wrapperov a na preklad extrahovaného obsahu do XML[5]. Medzi pokročilé funkcie Lixto patrí napríklad definícia disjunktných vzorov, špecializované pravidlá, zber a agregovanie informácií z niekoľkých previazaných stránok [6].

Nástroj obsahuje vlastné užívateľské IDE na vývoj WDE programov [7]. Vytváranie programu na extrakciu dát v nástroji Lixto funguje na základe jeho vlastného funkcionálneho jazyka ELOG podobného neprocedurálnemu jazyku Prolog [5].

Mozenda

Komplexný softvér Mozenda obsahuje zabudovaný internetový prehliadač a nahrávacie rozhranie, kde užívateľ buduje agenta, ktorý potom bude extrahovať želané dáta z webu. Budovanie agenta spočíva v navigácii k dátam užívateľom a to buď priamym označením textu myšou alebo použitím funkcie vyhľadávania, ktorá označuje HTML značku, kde sa hľadaná fráza nachádza. Ak vyhľadaná značka obsahuje nejaký nepotrebný text, užívateľ ho potom musí označiť v HTML kóde[8] . Nástroj využíva technológie HTML parsera, DOM a XPath. Okrem poplatku za program musí koncový užívateľ spúšťať agenta na harvest serveroch Mozenda a platiť za každý zozbieraný záznam [9].

2.2.4 Právne hľadisko

Použitím techniky WDE vo veľkom meradle (napríklad na celý web internetového obchodu) je možné v konečnom dôsledku skopírovať všetky verejne dostupné dáta z nejakého cieľového webu, či znížiť príjmy z reklamy. Prevádzkovateľ tohto skopírovaného webu tak

stráca kontrolu nad svojimi dátami, s ktorými ich nový vlastník môže nakladať po svojom, kde vzniká priestor na ich zneužitie.

Podľa [2] prebieha v USA v súčasnosti viacero súdnych sporov ohľadom WDE, kde poškodené strany žalujú spoločnosti, ktoré neoprávnene použili WDE na ich weby. Zatiaľ však žiadny z týchto spomínaných sporov neskončil konečným a právoplatným verdiktom, pretože právo v oblasti Internetu ešte nie je v USA ustálené. Na druhej strane v Dánsku už existuje súdny spor, v ktorom súd neuznal žalobu proti používateľovi WDE ako oprávnenú.

2.2.5 Metódy ochrany pred extrakciou dát z webu

Kvôli záporným aspektom spomínaným v predošlej časti vzniká u niektorých majiteľov webov potreba chrániť svoje dáta pred automatizovanou extrakciou. Správca webu má potom dva spôsoby ako zamedziť neoprávnenej extrakcii dát a to:

1. Bude blokovať komunikáciu so systémom extrahujúcim jeho dáta použitím niektorej z metód známej a používanej pri indexácii webu ako napríklad:
 - úplne zastaviť robota vykonávajúceho WDE blokovaním jeho IP adresy,
 - značne spomaliť robota znížením rýchlosti spojenia.
2. Skomplikuje štruktúru generovaných HTML dokumentov tak, aby bolo náročné pre extrahujúci systém spracovať tieto dokumenty. Tieto metódy sú špecifické pre WDE:
 - generovanie pozmenenej HTML štruktúry dokumentu za každou ďalšou požiadavkou robota na zobrazenie stránky, ktorá je však pre človeka vizuálne rovnaká,
 - vytvorenie zle spracovateľnej štruktúry HTML dokumentov
 - asynchrónne generovanie obsahu HTML stránky na strane klienta (za použitia AJAXu).

Vo všeobecnosti je účinnejšou technikou prevencie WDE druhý spôsob, keďže napríklad blokovanie IP adresy robota je možné obísť použitím viacerých robotov, ktoré môžu naviac používať dynamické IP adresy. Úloha extrakcie dát z webu sa tak v mnohých prípadoch stáva veľkou výzvou pre programátora systému na extrakciu dát.

3 Programovanie demonštráciou

Táto kapitola si kladie za cieľ popísať techniku programovania demonštráciou. Hlavným informačným zdrojom pre túto kapitolu je [10]. Úvodná kapitola **3.1 Programovanie koncovým užívateľom** vysvetľuje vznik potreby programovania koncovým užívateľom a jeho účel, zaraďuje programovanie demonštráciou do tejto skupiny a uvádza ďalšie techniky, ktoré môže koncový užívateľ použiť na prispôsobenie si softvéru. Podkapitola **3.2 Programovanie demonštráciou** popisuje túto techniku a približuje niektoré jej základné pojmy. Podkapitola **3.3 Nástroje PBD a typy techník používané v praxi** ďalej skúma nástroje spomínané v [10] a aké prístupy je možné zvoliť pri riešení úloh programovaním demonštráciou.

3.1 Programovanie koncovým užívateľom

Programovanie demonštráciou spadá do odboru informatiky zvaného **Programovanie koncovým užívateľom**, ktoré je v [11] definované ako: "sada metód, techník a nástrojov, ktoré umožňujú užívateľom softvérových systémov, ktorí vystupujú ako neprofesionálni vývojári softvéru, v nejakom momente vytvoriť, modifikovať alebo rozšíriť nejaký softvérový artefakt."

V súčasnosti sa za najznámejšiu formu programovania koncovým užívateľom považujú tabuľkové editory [12]. Príčinou ich vzniku je umožnenie širšej skupine ľudí realizovať výpočty bez toho, aby sa museli naučiť nejaký programovací jazyk.

Ďalším dôvodom vzniku programovania koncovým užívateľom je tiež fakt, že vývojári systémov nie sú schopní zohľadniť všetky potreby koncových užívateľov. Nespokojní užívatelia tak dostávajú možnosť (od vývojárov softvéru) si tento softvér prispôbiť. Podľa [10] sú vtedajšie¹ prístupy k programovaniu koncovými užívateľmi zhrnuté do štyroch typov: nastavenia, skriptovacie jazyky, makro rekordéry a programovanie demonštráciou.

Nastavenia aplikácie je vopred daná množina volieb, ktorými si užívateľ môže prispôbiť správanie sa aplikácie, s ktorou pracuje, podľa svojich požiadaviek. Keďže nastavenia ponúkajú iba obmedzené a nemenné množstvo volieb, nie sú v skutočnosti dostatočne všeobecné, aby mohli byť považované za formu programovania [10].

Ako uvádza Cypher v [10]: "Skriptovacie jazyky sú v súčasnosti populárnym prístupom k programovaniu koncovými užívateľmi." Skriptovacie jazyky sú jednoduchšími programovacími jazykmi, slúžiacimi na rozšírenie funkčností systému. Nádejou na ich širšie použitie je, že pre koncového užívateľa nebude obtiažne sa naučiť takýto jazyk. Avšak skriptovanie nie je pre koncových užívateľov príliš vhodné, pretože stále ide o formu programovania, kde sa užívatelia musia jednak naučiť základné koncepty programovania ako

¹ rok 1993

sú premenné, cykly a podmienky, a za druhé sú nútení učiť sa jazyk, ktorý možno je síce jednoduchší, no stále je považovaný za programovací jazyk.

Makro rekordéry poskytujú užívateľom spôsob ako zaznamenávať sledy svojich úkonov a neskôr ich opakovane vykonávať použitím jedného príkazu. Hoci užívateľ vytvorením makra vytvára istú formu programu, nemôžeme makro rekordéry považovať za skutočnú formu programovania, pretože vytvorené makrá sú až príliš priamočiare.

"Programovanie demonštráciou je rozvinutie princípu makro rekordérov tak, aby generalizované programy mohli obsahovať cykly a podmienky", tvrdí Cypher v [10]. Programovanie demonštráciou tak je možné považovať za skutočnú formu programovania, ktorá sa svojou robustnosťou môže porovnávať s programovaním.

3.2 Programovanie demonštráciou a jeho využitie

Podľa Wikipedie [12] je **programovanie demonštráciou** (ďalej označované ako **PBD** z anglického **Programming by Demonstration**) definované ako: "vývojová technika, kde koncový užívateľ naučí počítač alebo robot novému správaniu sa demonštrovaním úlohy priamo namiesto jej programovania strojovými príkazmi."

Podľa [10] je najväčšou výhodou PBD oproti tradičnému programovaniu fakt, že ide o **programovanie v užívateľskom rozhraní** (anglicky **Programming in the User Interface**) - pojem zavedený Danom Halbertom. Pri tradičnom programovaní sa vyžaduje od programátora, aby operácie, ktoré chce, aby systém vykonal, abstrahoval do zdrojového kódu, na čo mu slúži vývojové prostredie. V programovaní demonštráciou užívateľ programuje priamo v užívateľskom prostredí systému. Ak užívateľ chce, aby systém vykonal nejakú operáciu, jednoducho ju užívateľ vykoná sám, čo je niečo, čo už dávno ovláda a nemusí tak mapovať z vizuálnej reprezentácie tejto operácie do zdrojového programu. Pri tradičnom programovaní sa programátori musia naučiť iný, abstraktný a cudzí spôsob odkazovania sa na objekty a akcie.

Ako tvrdí Cypher v [10]: "Najväčšou výzvou, ktorej čelí PBD je, ako odvodiť užívateľov zámer". Na to aby systém dokázal vytvoriť program, ktorý zodpovedá procedúre, ktorú ho chce užívateľ naučiť, musí systém správne určiť užívateľov zámer pri jej vykonávaní. Keď bude program vykonávaný v budúcnosti, kontext bude samozrejme iný, a bude nutné vykonať akciu, ktorá je ekvivalentom zaznamenananej akcie, ale v tomto novom kontexte. Proces odvodzovania užívateľovho zámeru vo výbere konkrétneho objektu bol opísaný Danom Halbertom ako vytváranie "**popisu dát**" [10].

Otázkou je, ako má systém správne odvodiť užívateľov zámer, ak má k dispozícii len minimálne množstvo informácií o zázname užívateľských akcií. Systémy spomínané v [10] uvádzajú rozličné prístupy: v niektorých systémoch užívateľ vyberá z malého množstva alternatív, v iných systém porovnáva viacero príkladov a v ďalších užívateľ ukazuje relevantné informácie, či systém žiada užívateľa, aby schválil jeho interpretácie.

Užívatelia môžu vykonať danú úlohu mnohými rozličnými spôsobmi, čo je značne mäťúce pri odvodzovaní užívateľovho zámeru. Napríklad, vloženie kurzora do textu na začiatok druhého riadka predchádzajúceho odstavca môže zahŕňať niekoľko kliknutí myšou, kým si užívateľ vyberie tú správnu polohu, alebo môže byť vykonaný nespočtom rôznych kombinácií pohybových príkazov.

Programovanie demonštráciou má široký rozsah použitia. Najjednoduchším použitím je prispôsobovanie správania sa systému osobným potrebám koncového užívateľa. Takéto zmeny nazývané tweaking často korešpondujú s nastavovaním volieb, ktoré vývojár aplikácie nepredpokladal [10].

Asi najväčší potenciál použitia programovania demonštráciou je pri automatizácii opakujúcich sa úkonov, ako sa uvádza v [10]. Užívatelia musia bežne vykonávať iteratívne a periodické aktivity, ktoré im systém nedovoľuje automatizovať. Programovanie demonštráciou im tak umožní vykonať tieto procesy za nich, automatizovane.

Ďalším potenciálnym využitím programovania demonštráciou je tvorba jednoduchých aplikácií [10]. V prípade, že má užívateľ príliš špeciálne požiadavky a na trhu neexistujú nástroje, ktoré by ich zohľadňovali, môže použiť systém programovania demonštráciou na vytvorenie vlastnej aplikácie.

3.3 Nástroje PBD a typy techník používané v praxi

Hlavný zdroj pre túto kapitolu [10] kniha Watch What I Do : Programming by demonstration vydaná v roku 1993 je venovaná tematike PBD. Prezентuje celkovo 18 PBD počítačových programov rozličného použitia využívajúcich široký rozsah prístupov učenia a techník interakcie. Medzi programy zo spomínanej knihy, ktoré boli hlbšie analyzované patrí textový editor TELS. Ďalším zaujímavým programom relevantným k navrhovanému riešeniu extrakcie dát je systém Tourmaline, ktorý sa nezaobera spôsobom, ako užívateľ dospel k výsledku, ale samotný výsledok používa na odvodzovanie procesu učenia.

TELS

Textový editor TELS popísaný v spomínanej knihe bol hlavným podnetom na vznik tejto diplomovej práce - aplikovanie techniky PBD na extrakciu dát z webu. Podnetom sa stal najmä preto, lebo z uvedených systémov sa jeho zameranie na spracovanie textu najviac približovalo činnosti extrakcie (textových) dát z webu. Cieľom systému TELS je naučiť sa procedúru na spracovanie tetu v textovom editore opakovaním jednoduchých úkonov po užívateľovi.

Ako uvádza Cypher v [10] : "Systém TELS sa opiera o päť princípov, ktoré boli navrhnuté po úvodnom experimente s niekoľkými úlohami týkajúcimi sa preformátovania pološtruktúrovaného textu, ktoré boli zadane vykonať ľuďom ". Týchto päť princípov je:

- Využi abstraktný editovací model - ten obsahuje základné atomické operácie: vlož, nájdí, vyber, zmaž a všetky ostatné sú z nich poskladané.
- Zaznamenávaj bohatý kontext pre každú operáciu - jeho atribúty sa spoliehajú na lexikálnu dekompozíciu textu do menších jednotiek ako sú znaky, slová, riadky a odstavce. Použité sú tri základné typy atribútov. Prvý charakterizuje lokálny kontext aktuálnej pozície špecifikovaním predošlých a nasledujúcich slov. Druhý udáva pozičnú informáciu relatívnu k obklopujúcim jednotkám textu ako je začiatok, stred a koniec súboru, odstavca, alebo riadka. Tieto nazývame lexikálne atribúty. Lexikálne atribúty poskytujú menej lokálnu a oveľa menej detailnú informáciu ako lokálny kontext. Tretí meria relatívnu vzdialenosť od predošlej pozície a môže byť vyjadrený ako znaky, slová, riadky a odstavce.
- Nauč sa z jediného príkladu.
- Rozšír program, ak sa vyskytne chyba - na to, aby sa systémy vyrovnali s malými zmenami, musí užívateľ byť schopný rozšíriť procedúru za behu, aby zahrnul nové okolnosti. Toto je on-line odlaďovanie programov naučených demonštráciou [10].
- Nechaj užívateľa cvičiť.

Tourmalline

Tourmalline je systém na formátovanie textu, ktorý nevytvára makrá z prepisu užívateľských operácií. Namiesto toho sleduje konečný výsledok (príklad želaného štýlu) a z neho potom odvodzuje zovšeobecnenia. Je pre neho irelevantné, aká sekvencia krokov bola použitá na vytvorenie príkladu.

Naviac, na rozdiel od systému TELS tento systém obsahuje značnú znalosť typických formátov použitých v dokumente. Kdežto iný systém spracovania textu musí byť inštruovaný ako nájsť políčko autora v bibliografii sledovaním interpunkcie, Tourmalline má koncept "políčko autora" zabudovaný v sebe, takže vie kde ho hľadať. Použitie znalostí domény robí odvodzovanie v tomto systéme oveľa viac mocnejším a správnejším.

A predsa, systém nepracuje podľa príkladu, keďže používa heuristiky na odvodenie generalizácií z formátovaného príkladu textu. Preto by čitateľ nemal predpokladať, že všetky systémy "príkladom" alebo "demonštráciou" musia byť založené na makro rekordéroch.

4 Analýza

Táto kapitola je základným stavebným kameňom tejto práce, o ktorý sa opierajú všetky nasledujúce časti. Úvodná podkapitola **4.1 Definícia systému WDE-PBD**. Uvádza popis systému, ohraničenie jeho funkcionality a venuje sa výberu vhodnej techniky programovania demonštráciou. Nasledujúca podkapitola **4.2**

Definície základných pojmov definuje pojmy, s ktorými sa v celej tejto práci bude narábať, a preto sa pre dostatočné pochopenie neodporúča túto podkapitolu vynechať. Na pojmoch v nej definovaných stavia kapitola **4.3 Základné schémy algoritmov**, kde sú uvedené základné schémy algoritmov, ktoré musia byť zohľadnené pri návrhu a implementácii systému. Posledná podkapitola **4.4 Analýza problematiky WDE** sa venuje hĺbkovému štúdiu prezentovania dát užívateľovi na Internete. Vychádza z praktických skúseností nadobudnutých pri vývoji webových stránok. Skutočnosti a predpoklady analyzované v tejto podkapitole sú využívané pri návrhu a konkrétnej implementácii algoritmov a dátových štruktúr, ktoré sú detailnejšie popísané v kapitole **6 Algoritmy a dátové štruktúry**.

4.1 Definícia systému WDE-PBD

Táto podkapitola má za cieľ v hrubých rysoch definovať systém implementovaný v tejto práci. Uvádza sa tu základný postup, ktorým systém rieši úlohu extrakcie dát z webu (v **4.1.1 Popis činnosti**), na aké typy WDE sa systém obmedzuje (v **4.1.2 Obmedzenia na implementovaný typ WDE**) a akú PBD techniku systém použije pri učení (v **4.1.3 Voľba vhodnej techniky PBD na účely WDE**). Základné schémy algoritmov vyvinutých v tejto práci sú popísané v **4.3 Základné schémy algoritmov** a podrobnosti týkajúce sa návrhu algoritmov a ich implementácie sú uvedené v kapitole **6 Algoritmy a dátové štruktúry**.

Cieľom práce je vyvinúť systém na automatizovanú extrakciu dát z webu využitím techniky programovania demonštráciou. Vyvinutý systém je preto označovaný **WDE-PBD** (čo je skratka z anglického **Web Data Extraction by Programming by Demonstration**). Námet aplikovať techniku programovania demonštráciou na extrakciu dát z webu pochádza z knihy Watch What I Do [10], zo systému TELS. TELS je textový editor, ktorý slúži na uľahčenie práce užívateľovi, ktorý potrebuje vykonať často opakujúcu sa činnosť nad textovými dokumentmi.

4.1.1 Popis činnosti

WDE úloha znie:

Získať požadované informácie o nejakej entite z webových stránok do formy vhodnej na strojové spracovanie.

Postup extrakcie dát pri **štandardnom riešení WDE** je:

1. Získaj URL dokumentov, z ktorých budú extrahované dáta.
2. Na získaných URL vykonaj extrakciu.

Postup extrakcie dát v systéme **WDE-PBD** je definovaný nasledovne:

1. Získaj URL dokumentov, z ktorých budú extrahované dáta, pomocou procedúry naučenej technikou programming by demonstration.
2. Na získaných URL vykonaj extrakciu pomocou procedúry naučenej programovaním demonštráciou.

Kroku 2 zodpovedá primárny algoritmus vyvinutý v tejto práci **DE-PBD** — extrakcia dát technikou PBD (z anglického Data Extraction by Programming by Demonstration).

Kroku 1 zodpovedá sekundárny algoritmus vyvíjaný v tejto práci **UE-PBD** — extrakcia URL adres technikou PBD (z anglického URL Extraction by Programming by Demonstration), ktorého úlohou je poskytnúť algoritmu DE-PBD URL adresy obsahujúce vstupné dáta pre extrakciu. Ide o špecializovanejší prípad algoritmu DE-PBD.

Pre efektívny zber dát systémom je nielen nutné vedieť automatizovane extrahovať dáta z HTML dokumentov, ale aj získavať ich URL adresy pokiaľ možno tiež automatizovane. Preto je v kroku 1 použitá technika programovania demonštráciou, ktorá tak výrazne automatizuje a zrýchľuje celý proces extrakcie dát. Zvolený postup je potom vysoko automatizovateľný a modulárny, keďže umožňuje spúšťanie jednotlivých krokov osobitne, takže zohľadňuje vytýčené cieľové požiadavky, z ktorých najdôležitejšia je automatizovateľnosť.

Základné schémy týchto algoritmov sú definované neskôr v časti **4.3 Základné schémy algoritmov** a podrobnejšie sú rozobrané v kapitole **6 Algoritmy a dátové štruktúry**.

4.1.2 Obmedzenia na implementovaný typ WDE

Pripomeňme si definíciu WDE z časti 2.2:

Extrakcia dát z webu je softvérová technika, v ktorej sa automatizovane extrahujú informácie z HTML dokumentov.

Z hľadiska počtu webov, z ktorých HTML dokumenty obsahujúce extrahované dáta pochádzajú, môžeme rozdeliť WDE na:

- WDE z jedného webu,
- WDE z viacerých rozličných webov.

Nájsť všeobecný spôsob na extrakciu dát z viacerých rozličných webov pre jeden typ entity je všeobecne ťažšia úloha ako z jedného webu, keďže nemôžeme predpokladať, že stránky pochádzajúce z viacerých rozličných webov budú obsahovať tie isté dáta alebo podobnú HTML štruktúru pre rovnaký typ entity. Aplikovanie techniky PBD v tomto prípade by bolo veľmi obtiažne, keďže tá je založená na všímaní si podobností operácií v novom kontexte. WDE z viacerých rozličných webov je veľmi náročná úloha, ktorá rozsahom siaha nad rámec tejto diplomovej práce, a preto v tejto práci budem ďalej uvažovať o extrakcii dát už len pre jeden web.

Extrakciu môžeme rozdeliť taktiež z hľadiska počtu vstupných záznamov obsiahnutých v jednom HTML dokumente na prípady:

1. Jedna HTML stránka obsahuje jeden extrahovaný záznam.
2. Jedna HTML stránka obsahuje viacero vstupných záznamov.

V prvom prípade máme zaručené, že HTML dokumenty, ktoré sú vstupom pri extrakcii, budú mať veľmi podobné vlastnosti a formu a budú obsahovať rovnaké typy dát. V druhom prípade je tento predpoklad ešte silnejší, keďže sa dáta nachádzajú na jednej HTML stránke. Oba tieto prípady sú zohľadnené pri ďalšej analýze a implementácii.

Systém WDE-PBD bude určený na extrakciu dát z jedného webu a bude umožňovať oba typy extrakcie dát z hľadiska počtu záznamov.

4.1.3 Voľba vhodnej techniky PBD na účely WDE

Textový editor TELS popísaný v [10] bol hlavným podnetom na vznik tejto diplomovej práce - aplikovanie techniky programovania demonštráciou na extrakciu dát z webu. Hoci TELS pracuje s textom, podobne ako systém WDE-PBD, technika programovania demonštráciou, ktorá je v ňom použitá, je nevhodná pre definovaný postup extrakcie WDE-PBD.

V programe TELS sa systém snaží naučiť a zopakovať nejakú zatiaľ neznámu procedúru, ktorú mu demonštruje užívateľ. Na tento účel má nahrávacie rozhranie, ktoré zaznamenáva úkony vykonávané užívateľom, a na pozadí, vždy po každej operácii, sa snaží zo zaznamenaného sledu operácií odvodzovať procedúru.

Ak by sme v našom prípade WDE nadefinovali nad HTML dokumentom obmedzený editor, ktorý povoľuje iba označovať, dostali by sme sa do rovnakej situácie, kedy je potrebné naučiť systém procedúru na extrakciu textu.

Rozoberme si však činnosť extrakcie v systéme WDE-PBD. Pri demonštrácii extrakcie dát z dokumentu technikou PBD sú pre koncového užívateľa učiaceho systém nutné len 2 operácie: označenie textu myšou a extrakcia označeného textu do výstupných dát. Samozrejme, je možné označovať text HTML dokumentu klávesnicou, avšak takýto spôsob je pre užívateľa veľmi nepohodlný. Pre každý extrahovaný záznam je treba z HTML dokumentu extrahovať niekoľko atribútov za radom. V tomto zmysle je možné považovať extrakciu jedného záznamu za procedúru, avšak táto procedúra je veľmi dobre definovaná a neobsahuje mnoho rozličných krokov ako v prípade TELS, a teda nemá ani zmysel ju hľadať.

Vo WDE systéme by však mohla byť použitá technika programovania demonštráciou, v ktorej sa využíva nahrávacie rozhranie podobne ako v TELS, a to za predpokladu, že definujeme postup WDE-PBD nasledovne: Užívateľ by počas extrakcie navigoval z úvodnej stránky s nejakým zoznamom extrahovaných entít až na stránku obsahujúcu prvý záznam, kde by vykonal extrakciu. Následne by sa vrátením v histórii prehliadania dostal na úvodnú

stránku odkiaľ by navigoval na druhý záznam a t.d'. Po skončení extrakcie na úvodnej stránke zoznamov by prešiel na druhú a celý tento postup by opakoval, kým by sa systém (ne)naučil.

Tento postup sa však skladá z čiastkových krokov, ktoré už sú zohľadnené v tejto práci, už by ich bolo treba len prepojiť. Toto prepojenie by však vyžadovalo kontrolu nad históriou prehliadania, kde by bolo nutné zaručiť, aby pri navigácii späť, prehliadač neprenačítal stránku znova, ale použil už navštívenú verziu.

Ďalšou nevýhodou takto nadefinovanej procedúry by bola jej komplikovanosť a dĺžka z pohľadu učenia sa od užívateľa. Ten by svoju procedúru extrakcie dát musel vykonávať podľa možností čo najpresnejšie a najdemonštratívnejšie. V prípade, že by sa v nejakom kroku pomýlil, by vyvstal problém, ako postupovať ďalej. Možnosť neinformovať užívateľa a nechať ho vykonať zdĺhavú procedúru ďalej, až kým ju vykoná bezchybne a systém sa ju naučí, by bola z užívateľského hľadiska nemysliteľná. Alternatíva implementovať funkciu undo v procese učenia zase náročná na implementáciu, nehľadiac na to, že by bolo nutné použitie vlastnej histórie v kontexte prehliadača.

Miesto metódy s nahrávacím rozhraním bude systém WDE-PBD používať techniku odvodzovania podobností predkladaných vzorov na základe silnej znalosti domény, ako je to napríklad v systéme Tourmalline z [10].

4.1.4 Zhrnutie

Systém WDE-PBD bude určený na extrakciu dát z jedného webu a bude umožňovať oba typy extrakcie dát z hľadiska počtu záznamov.

Postup extrakcie dát v systéme **WDE-PBD** je definovaný nasledovne:

1. Získaj URL dokumentov, z ktorých budú extrahované dáta, pomocou algoritmu **DE-PBD**.
2. Na získaných URL vykonaj extrakciu pomocou algoritmu **UE-PBD**.

Systém WDE-PBD bude používať PBD techniku odvodzovania podobností predkladaných vzorov na základe silnej znalosti domény.

4.2 Definície základných pojmov

V tejto časti sú definované základné pojmy, týkajúce sa či už programovania demonštráciou, extrakcie dát alebo extrakcie URL. Pojmy z tejto podkapitoly sú používané naprieč celou prácou a preto sa neodporúča túto časť vynechať.

4.2.1 Obecné pojmy

Entita — nejaký objekt alebo vec, pre ktorú sa bude vykonávať extrakcia dát.

Atribúty entity — všetky vlastnosti entity.

Extrahované atribúty — podmnožina atribútov entity, ktorá je extrahovaná z HTML dokumentov.

Extrahovaný záznam — neprázdna množina (textových) hodnôt všetkých extrahovaných atribútov pre jednu entitu.

Extrahované dáta — neprázdna množina extrahovaných záznamov pre rôzne entity.

HTML dokument — neprázdna množina textov a HTML značiek, nevhodná na strojové spracovanie.

Časť HTML dokumentu — vlastná podmnožina (textov a HTML značiek) HTML dokumentu

Vstupný záznam — časť HTML dokumentu, obsahujúca nejaký extrahovaný záznam. Jeden vstupný záznam sa vždy nachádza len v jednom HTML dokumente. Jeden HTML dokument ich však môže obsahovať viac (a nemusí). (viď Obrázok 1 a Obrázok 3.)

Príklad č.1:²

Entita:	digitálny fotoaparát Canon EOS 5D Mark II.
Atribúty entity:	výrobca, model, ohnisková vzdialenosť, typ pamäťovej karty, farba ...
Extrahované atribúty:	výrobca, model, cena s DPH
HTML Dokument:	http://www.alza.cz/canon-eos-5d-mark-d105257.htm
Vstupný záznam:	viď Obrázok 1. Vstupný záznam
Extrahovaný záznam:	"Canon", "EOS 5D Mark II.", "49 990"

² obrázok v príklade nie je originálny screenshot a je upravovaný za účelom názornosti

CANON EOS 5D Mark II. body		Digitální zrcadlovka pouze tělo, 21.1 mil. bodů	
	Vaše cena bez DPH:	41 658 Kč	
	Vaše cena včetně DPH:	49 990 Kč	
	Běžná cena:	56 942 Kč	
	Ušetříte:	12% / 6 952 Kč	
	Na splátky:	Například dnes 4 999 Kč později 10 x 4 999 Kč	
		zobrazit další úvěry	
		Přidat záruku	
Prodloužení záruky od 4 199 Kč			
<u>Na cestě 1-2 ks</u>		1	KOUPIT
Popis zboží CANON EOS 5D Mark II. body			
Špičková profesionální digitální zrcadlovka Canon EOS Mark II nabízí vysoké rozlišení, výbornou rychlost zpracování a navíc pohodlné a kvalitní ovládání. To vše navíc doplňuje velký displej a možnost natáčet video ve full HD rozlišení. Obrazový snímač CMOS s rozměry 36 x 24 mm nabídné rozlišení 22 milionů bodů. Snímač se může pochytit také speciální funkci čistícího systému EOS s fluorovou povrchovou vrstvou. Zaostřovač funguje na principu TTL-CTR-SIR a nabídné celou řadu režimů i široký rozsah citlivosti. Při expozici je možné nastavit čas v rozpětí 30 s - 1/8000 s a opět nechybí celá řada možných přizpůsobení, kterou najdete také u blesku či možnosti vyvážení bílé. Pro ukládání pořízených snímků a videa slouží paměťové karty Compact Flash I a II včetně podpory MicroDrive. O pohodlné nastavení kompozice i následné prohlížení snímků se postará výborný displej s úhlopříčkou 3 palce a rozlišením 900 000 pixelů. S počítačem je fotoaparát možné propojit přes USB kabel, k propojení např. s televizorem či projektorom je možné využít vestavěný mini HDMI konektor. O napájení se stará dodávaný Li-Ion akumulátor, který postačí v závislosti na nastavení fotoaparátu a prostředí zhruba na 860 snímků. Hmotnost těla fotoaparátu je 810 gramů.			
Parametry a specifikace:			
Obrazový snímáček:			
Typ: CMOS 36 x 24 mm			
Rozlišení: 22 milionů pixelů			
Efektivní počet bodů: 21,1 milionů pixelů			
Poměr stran: 3 : 2			
Čistění snímáče: Integrovaný čistící systém EOS s fluorovou povrchovou vrstvou			
Typ barevného filtru: Primární barvy			
Zaostřování:			
Typ: TTL-CTR-SIR			
Režimy zaostření: inteligentní AF, jednosnímkový AF, inteligentní AF-servo			
Rozsah citlivosti: EV -0,5–18 (pri 20 °C a ISO 100)			
Ostřicí oblast: 9bodový AF (a 6 pomocných bodů AF)			
Výběr oblasti ostření: automatické a manuální			
Expozice:			
Rychlost závěrky: 30 - 1/8000 s			
Systém měření světla: měření TTL při max. rozevření clony s 35mmovým SPC			
Režim měření: poměrové měření, částečné měření, celoplošné měření se zdůrazněným středem			
Kompenzace expozice: +/-2 EV s krokem nastavení po 1/3 nebo 1/2 (lze kombinovat s AEB)			
LCD displej:			
Typ: TFT LCD			
Úhlopříčka: 3"			
Rozlišení: 920000 pixelů			
Pokrytí: přibližně 100%			
Hledáček			
Typ: pětiboký hranol			
Pokrytí: přibližně 98%			
Zvětšení: 0,71x			
Dioptrická korekce: -3 až +1 dioptrie			
Zrcadlo: polopropustné zrcadlo s rychlým návratem			
Rozměry:			
152 x 115,5 x 75 mm			

Obrázok 1. Vstupný záznam

Príklad č.2

Entita:	Vypísané kurzy stávkovej spoločnosti pre jeden futbalový zápas
Atribúty entity:	Domáci, hostia, deň , dátum stretnutia, podujatie, kurz výhry domácich, kurz výhry hostí, kurz remízy, kurz neprehry domácich, kurz neprehry hostí, ...
Extrahované atribúty:	Domáci, hostia, kurz výhry domácich, kurz výhry hostí, kurz remízy
Vstupné záznamy:	Vid' Obrázok 2. Vstupné záznamy
Extrahované dáta:	"Irán", "Brazília", "9.00", "5.00", "1.30" "Bolívia", "Venezuela", "1.85", "3.50", "3.90" "Čína", "Sýria", "1.88", "3.80", "3.50" "India", "Vietnam", "2.35", "3.60", "2.65" "Omán", "Gabon", "3.40", "3.40", "2.05" "Panama", "El Salvador", "1.88", "3.30", "4.10"
Vstupný záznam:	vid' Obrázok 3. Vstupný záznam
Extrahovaný záznam	"Irán", "Brazília", "9.00", "5.00", "1.30"

št	7. 10. 2010 19:00	Medzinárodné priateľské zápasy	Irán - Brazília (Neutrálna pôda)	9.00	5.00	1.30	+59
št	7. 10. 2010 23:00	Medzinárodné priateľské zápasy	Bolívia - Venezuela	1.85	3.50	3.90	+14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	Čína - Sýria	1.88	3.80	3.50	+14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	India - Vietnam	2.35	3.60	2.65	+14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	Omán - Gabon	3.40	3.40	2.05	+14
so	9. 10. 2010 4:00	Medzinárodné priateľské zápasy	Panama - El Salvador	1.88	3.30	4.10	+14

Obrázok 2. Vstupné záznamy

št	7. 10. 2010 19:00	Medzinárodné priateľské zápasy	Irán - Brazília (Neutrálna pôda)	9.00	5.00	1.30	+59
----	-------------------	--------------------------------	----------------------------------	------	------	------	-----

Obrázok 3. Vstupný záznam

4.2.2 Programovanie demonštráciou

Vzor — HTML značka alebo text, ktorý je demonštrovaný užívateľom.

Vstupná množina — množina, z ktorej sa vyberá jeden prípadne viacero vzorov. V prípade učenia robí výber užívateľ, v prípade extrakcie systém resp. heuristika.

Kontext - množina vlastností charakterizujúca vzor vzhľadom k vstupnej množine.

Heuristika — funkcia, ktorej hlavnou úlohou je samostatne vybrať nejakú podmnožinu (napríklad URL alebo hodnotu atribútu) zo vstupnej množiny (HTML dokument alebo nejaká jeho časť) podľa nejakej predom definovanej logiky. Heuristika v sebe nesie parametre podľa, ktorých dokáže vyfiltrovať zo vstupnej množiny výstupnú. Heuristika má schopnosť prispôbovať sa — to znamená, že po predložení vstupnej množiny a vzorov z nej vybraných prispôsobí tomuto výberu svoje parametre podľa svojej logiky.

Agent — obsahuje sadu heuristik, ktorými dokáže zo vstupného HTML dokumentu extrahovať výstupný záznam alebo zoznam URL.

4.2.3 Extrakcia dát

HTML dokument — vstupná množina.

Označenie — označený text HTML dokumentu, vzor pri učení.

Kontext označenia — všetky informácie popisujúce dané označenie.

Extrakcia — potvrdenie označenia užívateľom a uloženie jeho textu ako hodnoty atribútu práve spracovávaného výstupného záznamu.

Dátová heuristika — heuristika, ktorá má za úlohu predpovedať označenie v HTML dokumente určené na extrakciu, na základe znalosti predošlých extrakcií a ich kontextov. Jej vstupom je HTML dokument a výstupom je označenie.

Dátový agent — sada heuristik, slúžiaca na extrakciu celého záznamu z jedného HTML dokumentu. Počet heuristik dátového agenta je rovnaký ako počet atribútov výstupného záznamu.

4.2.4 Extrakcia URL

Detail záznamu — celý vstupný záznam. (viď Obrázok 1.)

Náhľad záznamu — časť HTML dokumentu, obsahujúca nejaké (zvyčajne málo a určite nie všetky extrahované) atribúty nejakej extrahovanej entity. Tento náhľad obsahuje tiež URL odkazujúcu na detail záznamu. (viď Obrázok 8.)

Náhľadová oblasť — časť HTML dokumentu obsahujúca neprázdnu množinu náhľadov záznamov. Obsahuje vstupnú množinu pri učení. (viď Obrázok 5.)

Stránkovacia oblasť — časť HTML dokumentu, obsahujúca URL vedúce na nasledujúce náhľadové stránky. Vstupná množina pri učení. (viď Obrázok 6. a Obrázok 7.)

Navigačná stránka — HTML dokument obsahujúci náhľadovú oblasť a stránkovaciu oblasť. (viď Obrázok 4.)

Východisková stránka — náhľadová stránka, na ktorú užívateľ naviguje web po vybraní si skupiny entity. (viď Obrázok 4.)

URL — vzor pri použití techniky programovania demonštráciou.

Kontext URL — index v zozname URL (bud' stránkovacej alebo náhľadovej oblasti).

URL heuristika — heuristika, ktorá má za úlohu vybrať množinu URL náhľadovej oblasti vstupného HTML dokumentu na základe znalostí predošlých výberov takýchto URL užívateľom.

URL Agent — sada dvoch heuristík:

- heuristika na získanie URL z náhľadov záznamov náhľadovej oblasti vstupného HTML dokumentu.
- heuristika na získanie URL na ďalšiu navigačnú stránku z navigačnej oblasti vstupného HTML dokumentu.

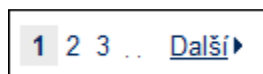
Príklad³:

Typ Entity:	digitálny fotoaparát
Východisková stránka:	vid' Obrázok 4. Východisková/navigačná stránka (http://www.alza.cz/foto-video/digitalni-foto/zrcadlovy/tela-bez-objektivu/28851892.htm)
Navigačná stránka	vid' Obrázok 4. Východisková/navigačná stránka
Náhľadová oblasť	vid' Obrázok 5. Náhl'adová oblasť
Stránkovacia oblasť	vid' Obrázok 6. Horná stránkovacia oblasť
Stránkovacia oblasť	vid' Obrázok 7. Dolná stránkovacia oblasť
Náhľad záznamu:	vid' Obrázok 8. Náhl'ad záznamu (odkazuje na http://www.alza.cz/canon-eos-5d-mark-d105257.htm)
Detail záznamu:	vid' Obrázok 1. Vstupný záznam (sa nachádza na http://www.alza.cz/canon-eos-5d-mark-d105257.htm)

³ obrázky v príklade nie sú originálne screenshoty a sú mierne modifikované za účelom názornosti

CANON EOS 5D Mark II. body Digitální zrcadlovka pouze tělo, 21.1 mil. bodů + ZDARMA Software Adobe Photoshop Elements 8  -12% 56 942,- 41 658,- s DPH 49 990,- Na cestě 1-2 ks Koupit Špičková profesionální digitální zrcadlovka Canon EOS Mark II nabízí vysoké rozlišení, výbornou rychlost zpracování a navíc pohodlné a kvalitní...	SONY DSLR-A330 tělo Digitální zrcadlovka černá, 10 Mpx, výklopný 2.7" LCD, Li-Ion, LifeView, stabilizátor - pouze tělo  -13% 10 567,- 7 658,- s DPH 9 190,- Skladem 3-5 ks Koupit Vysoce kvalitní, propracovaná a cenově dostupná digitální zrcadlovka pro náročné fotografy. Zrcadlovka disponuje rozlišením 10, 2 megapixelů, což...	SONY DSLR-A450 Digitální zrcadlovka černá, 14 Mpx, 2.7" LCD, Li-Ion, CF, SDHC, stabilizátor, pouze tělo  -12% 14 269,- 10 408,- s DPH 12 490,- Skladem 1-2 ks Koupit Výjimečnou kvalitu obrazu, rychlé odezvy a výkonné kreativní funkce nabízí tato nová digitální zrcadlovka. Snímač CMOS s vysokým rozlišením zajišťují...
CANON EOS 550D body Digitální zrcadlovka 18 mil. bodů CMOS DIGIC IV, RAW, 3" LCD, HDMI, FullHD video, SD/ SDHC  -12% 19 266,- 14 158,- s DPH 16 990,- Produkt je dočasně nedostupný Koupit Špičková digitální zrcadlovka vhodná pro ty nejnáročnější uživatele přináší rozlišení 18 megapixelů a obrovské množství funkcí a možností nastavení, ...	CANON EOS 1Ds Mark III body Digitální zrcadlovka pouze tělo, zrcadlovka 21 mil. bodů + ZDARMA Software Adobe Photoshop Elements 8 + ZDARMA 16GB SD karta  -10% 171 953,- 129 158,- s DPH 154 990,- Dostupnost na dotaz Koupit Revoluční digitální zrcadlovka s rozlišením neuvěřitelných 21 megapixelů pro nejnáročnější použití. Vyznačuje se především...	SONY SLT-A55V tělo Digitální zrcadlovka černá, 16 Mpx, výklopný 2.7" LCD, Li-Ion, LifeView, stabilizátor - pouze tělo  -5% 20 460,- 16 242,- s DPH 19 490,- Obdobně, termín nepotvrzen Koupit Lehká a kompaktní digitální zrcadlovka vynikající se velmi rychlým souvislým fotografováním až 10 snímků za sekundu, přesným automatickým ostřením ...

Obrázok 5. Náhl'adová oblasť



Obrázok 6. Horná stránkovacia oblasť



Obrázok 7. Dolná stránkovacia oblasť

CANON EOS 5D Mark II. body
 Digitální zrcadlovka pouze tělo, 21.1 mil. bodů
 + **ZDARMA** Software Adobe Photoshop Elements 8

 -12% 56 942,-
41 658,-
 s DPH 49 990,-
 Na cestě 1-2 ks
 Koupit
 Špičková profesionální digitální zrcadlovka Canon EOS Mark II nabízí vysoké rozlišení, výbornou rychlost zpracování a navíc pohodlné a kvalitní...

Obrázok 8. Náhl'ad záznamu

4.3 Základné schémy algoritmov

Táto podkapitola obsahuje základné schémy algoritmov vyvíjaných v tejto práci a nevenuje sa detailom ich fungovania. Časť 4.3.1 popisuje všeobecnú schému techniky PBD aplikovanej na extrakciu dát z HTML. Časť 4.3.2 je špecifickejšia a popisuje schému algoritmu extrakcie dát. Nakoniec 4.3.3 predstavuje schému extrakcie URL adres technikou PBD. Tieto schémy sú rozvinuté v kapitole **6 Algoritmy a dátové štruktúry**.

4.3.1 Schéma techniky PBD

Cieľ algoritmu:

Nájsť heuristiku zodpovedajúcu užívateľovej logike výberu vzorov zo vstupnej množiny.

Zovšeobecnenie:

Tu popísaný algoritmus učenia je zobecnený pre jeden typ heuristiky (napríklad heuristika výberu hodnoty atribútu z dokumentu alebo heuristika výberu množiny URL z náhľadovej oblasti.).

Vstupy:

HTML dokument, demonštrácia užívateľom.

Výstupy:

Heuristika výberu (prípadne informácia o nemožnosti nájsť adekvátnu heuristiku, ktorá by zodpovedala užívateľovej demonštrácii).

Parametre:

Dĺžka učenia — počet krokov algoritmu, počas ktorých musí byť splnená istá podmienka a učenie heuristiky sa skončí.

Schéma:

1. Počiatočná inicializácia:

1. Systém vytvorí všetky inštancie heuristik rovnakého typu.

2. Iterácia:

1. Užívateľ demonštruje výber vzoru zo vstupnej množiny.
2. Systém sám načíta a udržiava informácie o vstupnej množine na základe demonštrovaného vzoru.
3. Systém si podrobne zaznamená kontext demonštrovaného vzoru v rámci vstupnej množiny.

4. Systém predloží tento vzor všetkým heuristikám na adaptovanie:
5. V prípade, že heuristika nie je schopná adaptovať sa na vzor, je označená za neschopnú učenia a systém ju vyradí zo zoznamu heuristík.
6. V prípade, že heuristike bol predložený dostatočný počet vzorov, je označená za naučenú. Inak je označená za nenaučenú.

3. Ukončenie:

Iterácia (t.j. celý proces učenia sa) môže skončiť nasledujúcimi možnosťami:

1. Zoznam heuristík je prázdny — nepodarilo sa nájsť odpovedajúcu heuristiku.
2. V zozname heuristík existuje naučená heuristika v toľkých iteráciách ako je dĺžka učenia — užívateľovi je ponúknutá táto heuristika na schválenie.
3. V zozname heuristík neexistuje naučená heuristika v toľkých iteráciách ako je dĺžka učenia — proces demonštrácie sa končí, nepodarilo sa nájsť heuristiku.

Poznámky:

Záver procesu sa odvíja od implementovaných heuristík. V prípade, že všetky heuristiky daného typu potrebujú presný počet predložených vzorov na výpočet parametrov, sa učenie vždy skončí prípadom a) alebo b).

Podrobnejší popis algoritmu sa nachádza v **6.1 PBD**.

4.3.2 Schéma algoritmu extrakcie dát

Cieľ algoritmu:

Za pomoci agenta naučeného technikou programovania demonštráciou automatizovane získať z HTML dokumentov na vstupe extrahované dáta o množine entít za účelom strojového spracovania.

Vstupy:

Zoznam URL HTML dokumentov obsahujúcich vstupné záznamy, demonštrácia užívateľom.

Výstupy:

Extrahované dáta v podobe jednoduchkej tabuľky, agent na extrakciu dát.

Schéma:

1. Iterácia:

1. Načítaj HTML dokument z ďalšej URL zo vstupu.
2. Nechaj užívateľa demonštrovať zber dát na dokumente: pre každý extrahovaný atribút aplikuj techniku PBD na nájdenie dátovej heuristiky, ktorá tento atribút hľadá.

2. Ukončenie:

Iterácia sa môže skončiť nasledujúcimi možnosťami:

- a. Agent sa nepodarilo naučiť — nepodarilo sa nájsť heuristiku pre nejaký atribút.
- b. Agent sa podarilo naučiť — pre každý atribút existujú naučené heuristiky:
- c. Ulož agenta.
- d. Spusti agenta samostatne na zvyšok vstupu.

Poznámky:

Vstup

Vyhľadanie URL adries obsahujúcich vstupné záznamy, ktoré majú byť strojovo spracované môže byť realizované:

- o načítaním z textového súboru,
- o načítaním HTML dokumentov z adresára na disku,
- o použitím algoritmu UE-PBD (URL extraction by programming by demonstration) na extrakciu URL adries zo stránok extrahovaného webu.

Krok 2.

Pre zber dát užívateľom bude primárne slúžiť označenie vybraného textu myšou, ďalšie operácie slúžiace na vyhľadanie a označenie dát môžu byť:

- posuň kurzor na ďalšie/predchádzajúce slovo/znak,
- označ/odznač ďalšie/predchádzajúce slovo/znak,
- pridaj/odstráň ďalšie/predchádzajúce slovo/znak do výberu,
- vyhľadaj text.

Dôležitou podmienkou realizácie tejto metódy je schopnosť získať presné miesto v HTML strome, ktoré zodpovedá označenému textu užívateľom.

Algoritmus je detailnejšie popísaný v kapitole **6.2 DE-PBD**.

4.3.3 Schéma algoritmu extrakcie URL adries

Cieľ:

Za pomoci agenta naučeného technikou programovania demonštráciou automatizovane získať z HTML dokumentu na vstupe URL obsahujúce vstupné záznamy.

Vstupy:

východisková stránka, demonštrácia užívateľom.

Výstupy:

zoznam URL adries obsahujúcich vstupné záznamy, agent na extrakciu URL.

Schéma:**Inicializácia:**

Načítaj východiskovú stránku.

Kroky:

1. Učenie agenta výberu URL v stránkovacej oblasti pomocou techniky programming by demonstration — užívateľ demonštruje URL náhľadov záznamov vedúce na detaily vstupných záznamov, kým nie sú vyčerpané všetky URL v navigačnej oblasti východiskovej stránky alebo nie je učenie heuristiky výberu URL náhľadov záznamov dokončené.
2. Učenie agenta výberu URL ďalšej navigačnej stránky v stránkovacej oblasti technikou programming by demonstration — užívateľ demonštruje v stránkovacej oblasti URL vedúcu na ďalšiu navigačnú stránku.
3. Naviguj na URL demonštrovanú v kroku 2.
4. V prípade, že:
 - heuristika z kroku 1 je naučená — vykonaj pomocou nej extrakciu URL z náhľadovej oblasti,
 - inak opakuj krok 1.
5. Opakuj kroky 2, 3 a 4 kým nie je učenie ukončené alebo sú vyčerpané všetky URL zo stránkovacej oblasti.

Ukončenie:

Iterácia sa môže skončiť nasledujúcimi možnosťami:

- Agentu sa nepodarilo naučiť — neporadilo sa nájsť heuristiku výberu URL z náhľadovej oblasti alebo heuristiku prechodu na ďalšiu navigačnú stránku.
- Agentu sa podarilo naučiť — existujú naučené heuristiky z kroku 1 i 2:
 1. Ulož agenta.
 2. Spuť naučeného agenta na zvyšok dát samostatne, kým nie sú vyčerpané URL navigačnej oblasti poslednej stránky výsledkov.

Poznámky:**Krok 1, Krok 4**

Algoritmus učenia v prvom kroku skonverguje relatívne skoro (ešte pred vyčerpaním URL odkazov). Keďže očakávaný počet unikátnych URL v náhľadovej oblasti je rádovo v desiatkach, maximálne stovkách, od tohto momentu v kroku 4 sa môže spúšťať už naučená časť agenta, ktorý rozpoznáva želané URL v náhľadovej oblasti.

Algoritmus je bližšie popísaný v podkapitole **6.3 UE-PBD**.

4.4 Analýza problematiky WDE

Na to, aby bol vyvinutý systém čo najviac univerzálny je nutné, aby dokázal extrahovať dáta z najrozmanitejších zdrojov. Preto je nutné detailnejšie preskúmať spôsoby prezentovania dát užívateľovi na Internete. V časti **4.4.1 Typy webov a techniky generovania HTML dokumentov** je analyzovaný vplyv typu webu na obsah a štruktúru výsledných HTML dokumentov, ktoré ho tvoria. Táto časť implikuje existenciu určitej šablóny používanej webmi, ktorej štruktúra je rozoberaná v časti **4.4.2 Kompozícia webu a stránky**. Na časť **4.4.3 Navigácia užívateľa k dátam**, ktorá zase hovorí o nevyhnutnosti existencie dvoch spôsobov prezentácie dát užívateľovi za účelom navigácie k nim, nadväzuje časť **4.4.4 Formy prezentácie dát**, ktorá si potom analyzuje možné formy jednotlivých spôsobov.

4.4.1 Typy webov a techniky generovania HTML dokumentov

Na Internete sa nachádza mnoho typov webov od osobných stránok, cez stránky prezentujúce súkromné firmy, stránky štátnych inštitúcií, až po rôzne rezervačné systémy, internetové obchody, sociálne siete či portály. Každý z nich používa rôzne technológie a metódy pri ich tvorbe, ktoré sa odvíjajú najmä od množstva informácií a služieb na ňom poskytovaných.

Vezmime si osobné stránky. Tie neobsahujú také veľké množstvo informácií ako napríklad portály a už zo svojej podstaty informovať o osobe ani zďaleka neposkytujú toľko služieb ako portály. Pri typoch webov, ktoré nemusia poskytovať veľké množstvo informácií či služieb je typické, že dokumenty, z ktorých sa skladajú sú statické. To znamená, že po príchode užívateľovej požiadavky na server je stránka vyzdvihnutá z disku servera a ihneď bez akejkoľvek ďalšej modifikácie odoslaná späť užívateľovi, ktorý o ňu zažiadal. Preto na ich tvorbu a údržbu nie sú kladené žiadne zvláštne nároky a tieto úkony sa zväčša vykonávajú ad hoc, ručne, bez použitia nejakých univerzálnych vývojových nástrojov či riešení a často nepotrebujú ani databázu. Ako už bolo spomenuté, táto práca sa venuje extrakcii dát len z jedného zdroja a teda v tomto prípade sú takéto typy webov nezaujímavé z hľadiska extrakcie dát kvôli malému množstvu informácie v nich obsiahnutých.

Na druhú stranu si vezmime napríklad internetové obchody, ktoré sú pre extrakciu dát omnoho zaujímavejším a vhodnejším kandidátom. Pri súčasnom množstve informácií a záznamov, ktoré musia takéto komplexné weby zobrazovať, je neefektívne, aby pre každú jednu entitu (napr. fotoaparát) z ich domény fyzicky existovala jedna statická HTML stránka vytvorená programátorom, ktorá ju zobrazuje⁴. Takýto prístup by bol z hľadiska zmien v zobrazovaní typu entity v praxi neudržiavateľný a enormne náročný na pamäť.

⁴ Pri použití dočasných cache pamätí však je možné mať pre každú entitu jednu HTML stránku. Tieto stránky však nie sú generované ručne - webovým programátorom, ale strojovo a existujú len za účelom zníženia vyťaženia servera.

Kvôli týmto nedostatkom sa pre zobrazovanie tried záznamov používajú dynamicky generované stránky, ktorých obsah je vytváraný po prijíme užívateľskej požiadavky na zobrazenie konkrétneho záznamu. Technológií na dynamickú tvorbu HTML stránok je mnoho a všetky (či už je to ASP .NET, PHP alebo JSP) fungujú všeobecne na tom istom princípe. Pre každú triedu záznamov definujú istú šablónu, ktorá má v konečnej fáze HTML formát. Do nej po príchode požiadavky už len dynamicky vyplňajú textové dáta uložené v databáze, poprípade doplnia obrázky a iné HTML elementy a výslednú HTML stránku potom pošlú užívateľovi.

Túto šablónu môžeme formálne definovať ako transformáciu, ktorej vstupom je (databázová) entita a jej výstupom je HTML dokument, ktorý túto entitu prezentuje užívateľovi. Proces extrakcie dát z webu je presne inverznou funkciou k tejto transformácii, kde sa z HTML výstupu určeného človeku extrahujú dáta do strojovej formy.

Práve o existenciu týchto šablón sa opiera systém WDE-PBD. Systému sú predkladané záznamy vytvorené za použitia jednej šablóny, ktoré majú veľmi podobnú (ak nie takmer identickú) HTML štruktúru. Pri spracovávaní aktuálneho záznamu to umožňuje systému ľahšie predpovedať polohu konkrétneho atribútu určeného na extrakciu, na základe informácií o kontextoch extrakcie tohto atribútu v predošlých krokoch.

4.4.2 Kompozícia webu a stránky

K dobrým postupom pri vývoji webu patrí dodržiavanie istých konvencií pri rozvrhovaní stránky. Aby stránky webu plnili svoju funkciu a nepôsobili na užívateľa máťúco, musia vyzeráť jednotne a všetky mať rovnaké rozvrhnutie (layout). Medzi štandardné časti rozvrhnutia patria tieto oblasti: záhlavie stránky, pätička stránky, bočné panely a oblasť určená pre obsah konkrétneho HTML dokumentu.

Menu, slúžiace užívateľovi k navigácii po webe, sa nachádza buď pod záhlavím (v prípade jednoduchšieho menu) alebo v jednom z bočných panelov (v prípade zložitejšieho menu). Samotné dáta sú užívateľovi prezentované v paneli obsahu, ktorý je pre extrakciu dát cieľovou oblasťou.

Ďalším častým faktom je, že rozvrhnutie HTML stránky je robené podľa nejakého profesionálneho grafického návrhu. Existuje celkom silný predpoklad, že tento návrh sa nebude často meniť najmä z dôvodu finančného, ale aj kvôli už existujúcim užívateľom webu, ktorý si na jeho vizuál zvykli.

Dôležitým faktorom, ktorý je treba zohľadniť pri extrakcii dát z webu je reklama. Dnes už prakticky neexistuje web, ktorý by ju neobsahoval. Z hľadiska extrakcie predstavuje reklama neželaný obsah, ktorý sa často miesi s dátami a sťažuje tak podmienky na ich účinnú extrakciu. Navyše reklama je veľmi dynamickým prvkom HTML dokumentu, ktorý často (rádovo v niekoľkých dňoch či týždňoch) mení svoj obsah.

Ak zjídeme v analýze ešte ďalej, zistíme, že dnešné moderné weby si už nevystačia iba s HTML jazykom a obsah stránok je mnohokrát veľmi dynamický. Jazyk HTML bol navrhnutý pre účely informovať užívateľov, no s rozvojom webu sa do popredia dostávajú technológie JavaScript a AJAX, ktoré užívateľom umožňujú lepšiu interakciu s webom, či v hraničnom prípade dokonca simuláciu samostatnej okienkovej aplikácie v internetovom prehliadači. Medzi ich najčastejšie využitie patrí dynamické prekresľovanie menu, kontrola validity formuláru, či rôzne animácie. Stránka sa tak stáva doslova žijúcim obsahom, kde majú niektoré elementy obmedzenú životnosť, môžu byť neviditeľné a zobrazovať sa v príhodný čas, či ich textový obsah sa mení podľa užívateľom vyvolanej akcie. Za takýchto podmienok sa štruktúra HTML dokumentu stáva veľmi nestálou a dynamickou, čo opäť sťažuje podmienky na extrakciu dát z nej.

4.4.3 Navigácia užívateľa k dátam

Panel s obsahom, spomínaný v predošlej časti, však neslúži len na zobrazovanie dát, ale aj navigáciu k nim. Weby, zaujímavé pre WDE, majú vo veľkej väčšine doménu, ktorej sa venujú, veľkú. Vo svojich databázach majú tisícky možno aj milióny záznamov. Tieto weby musia svojím užívateľom ponúknuť nejakú rozumnú formu navigácie k dátam. Väčšinou je táto navigácia riešená kombináciou menu a panelu obsahu.

Základným krokom umožňujúcim navigáciu užívateľa k dátam je rozčlenenie entít na skupiny. Tieto skupiny sú potom všetky zobrazené buď v menu, alebo v paneli obsahu. Keďže týchto skupín (počítajmo podskupín) nebýva veľa (rádovo desiatky), zmestia sa do menu či panelu obsahu všetky. Po navigácii užívateľom do skupiny mu web neponúkne zoznam všetkých entít v danej skupine, ale možnosť ich prehliadania pomocou filtrov, ktoré sa nachádzajú najčastejšie vo vrchnej časti panelu obsahu.

Filtrovaný obsah sa vypisuje pod filter. V tomto zozname bývajú zobrazené len náhľady a po kliknutí na náhľad sa užívateľ dostane k detailu cieľovému záznamu, o ktorý má záujem. Na tejto konečnej⁵ stránke sú dáta zobrazované rozličnými formami, ktoré sú rozoberané v ďalšej podkapitole.

Počet zobrazovaných detailov v zozname položiek býva v drvivej väčšine v rádoch desiatok, výnimočne stoviek, hoci počet položiek, vyhovujúci kritériám zadaným vo filtri, môže byť ďaleko vyšší. Na to, aby sa užívateľ dostal k položkám vyhovujúcim filtru, ktoré momentálne nie sú zobrazené, sa používajú odkazy na ďalšie (resp. predošlé) stránky výsledkov. Tieto odkazy nasledujú zvyčajne hneď za zoznamom náhľadov, niekedy sú umiestnené dvojmo, aj tesne pred zoznamom. Užívateľ tak dostáva možnosť navigovať po stránkach výsledkov a má prístup ku všetkým položkám vyhovujúcim filtru. Môže prejsť na prvú a poslednú stránku výsledkov a môže prejsť priamo na niekoľko predchádzajúcich a niekoľko nasledujúcich

⁵ v zmysle navigácie užívateľa k dátam

stránok pomocou týchto odkazov. Ich počet je znova malý výnimočne presahujúci číslo 20, a všetky sú umiestnené blízko seba.

4.4.4 Formy prezentácie dát

Informácie sú vo svete Webu sú prezentované človeku v podobe HTML dokumentov, čo je forma slúžiaca na prezentáciu dát človeku. Pri výmene informácií medzi strojmi je formátovanie dát nepodstatné. Počítač sa netrápi, že je nejaký text zobrazený tučne červenou farbou, pretože významu tejto skutočnosti nerozumie. Pre človeka však táto kombinácia môže mať veľký význam. Práve takéto skryté informácie by sa mal systém WDE-PBD snažiť všímať si čo najviac.

Najväčšou výzvou pri vývoji tohto systému WDE tak, aby bol čo najviac univerzálny, je schopnosť adaptácie algoritmu na rozličné formy prezentácie dát. Preto je nutné hlbšie analyzovať spôsob prezentácie dát. Ako podklad pre túto analýzu slúžilo množstvo reálnych stránok internetových obchodov, katalógov, či rezervačných systémov (medzi nimi napríklad www.alza.cz, www.dpreview.com, www.mobil.idnes.cz, www.pallacecinemas.cz a iné).

Na každom z analyzovaných príkladov boli sledované nasledujúce charakteristiky:

- účel zobrazenia:
 - o náhľad/odkaz slúžiaci na navigáciu k detailu záznamu,
 - o podrobný detail záznamu;
- počet zobrazených záznamov v jednom HTML dokumente;
- HTML štruktúra poprípade element, v ktorej sú dáta zobrazované:
 - o tabuľka (značky `<table>`, `<tr>`, `<td>` poprípade `<th>`),
 - pri značke `<th>` a `<td>` si budeme všímať taktiež plochosť dát v nich obsiahnutých podľa 1. normálnej formy (teda obsah už nemá ďalšiu podštruktúru);
 - o iná štruktúra: odstavec (značka `<p>`) alebo blok (značka `<div>`);

Z výskumu plynie, že HTML dokumenty zobrazujúce detaily viacerých záznamov (viď Obrázok 9.) používajú na ich zobrazenie v drvivej väčšine tabuľku, ktorá je na tento účel najvhodnejším HTML elementom. Tieto tabuľky sa vyskytujú v 2 variantoch: buď sú názvy atribútov v záhlaví tabuľky alebo sa nachádzajú v prvom stĺpci tabuľky. Druhý variant s prvým stĺpcom je však zriedkavejší, keďže umožňuje zobraziť len obmedzené množstvo dát v smere toku textu dokumentu (stránky sú vždy vyššie ako širšie). Ďalšou vlastnosťou takýchto zobrazení je plochosť väčšiny dát v elementoch `<td>` resp. `<th>`.

Futbal						
Day	Date	Event	Game	1X2		
				1	X	2
št	7. 10. 2010 19:00	Medzinárodné priateľské zápasy	Irán - Brazília (Neutrálna pôda)	9.00	5.00	1.30 +59
št	7. 10. 2010 23:00	Medzinárodné priateľské zápasy	Bolívia - Venezuela	1.85	3.50	3.90 +14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	Čína - Sýria	1.88	3.80	3.50 +14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	India - Vietnam	2.35	3.60	2.65 +14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	Omán - Gabon	3.40	3.40	2.05 +14
pi	8. 10. 2010 12:00	Medzinárodné priateľské zápasy	Kuvajt - Bahrajn	2.20	3.30	3.10 +14
pi	8. 10. 2010 12:50	Medzinárodné priateľské zápasy	Japonsko - Argentína	4.50	4.00	1.65 +14
pi	8. 10. 2010 13:30	Medzinárodné priateľské zápasy	Taiwan - Macau	1.20	6.00	12.50 +14
pi	8. 10. 2010 15:00	Medzinárodné priateľské zápasy	Indonézia - Uruguaj	11.50	5.75	1.22 +14
pi	8. 10. 2010 20:00	Medzinárodné priateľské zápasy	Ukrajina - Kanada	1.25	5.25	11.00 +14
pi	8. 10. 2010 23:00	Medzinárodné priateľské zápasy	Peru - Kostarika	1.70	3.75	4.40 +14
pi	8. 10. 2010 23:00	Medzinárodné priateľské zápasy	Guatemala - Belize	1.34	4.50	9.00 +14
so	9. 10. 2010 2:30	Medzinárodné priateľské zápasy	Ekvádor - Kolumbia (Neutrálna pôda)	2.85	3.50	2.25 +14
so	9. 10. 2010 4:00	Medzinárodné priateľské zápasy	Panama - El Salvador	1.88	3.30	4.10 +14

Obrázok 9. Dokument zobrazujúci viacero detailov záznamov, názvy atribútov v záhlaví

Dokumenty zobrazujúce viacero náhl'adov (vid' Obrázok 10.) slúžia na navigáciu k vstupným záznamom, a preto obsahujú iba niekoľko atribútov entity (zväčša názov a niekoľko ďalších základných údajov). Značky <td> v nich obsahujú väčšinou štruktúrované dáta. Ostatné formy, nepoužívajúce tabuľku, sú veľmi zriedkavé.

CANON EOS 5D Mark II. body Digitálna zrcadlovka pouze tělo, 21.1 mil. bodů + ZDARMA Software Adobe Photoshop Elements 8  -15% 56 942,- 40 408,- s DPH 48 490,- Skladem > 5 ks Koupit Špičková profesionální digitální zrcadlovka Canon EOS Mark II nabízí vysoké rozlišení, výbornou rychlost zpracování a navíc pohodlné a kvalitní...	SONY DSLR-A330 tělo Digitální zrcadlovka černá, 10 Mpx; výklopný 2.7" LCD, Li-Ion, LifeView, stabilizátor - pouze tělo  -13% 10 567,- 7 658,- s DPH 9 190,- Skladem 3-5 ks Koupit Vysoce kvalitní, propracovaná a cenově dostupná digitální zrcadlovka pro náročné fotografy. Zrcadlovka disponuje rozlišením 10,2 megapixelů, což...	SONY DSLR-A450 Digitální zrcadlovka černá, 14 Mpx, 2.7" LCD, Li-Ion, CF, SDHC, stabilizátor, pouze tělo  -12% 14 269,- 10 409,- s DPH 12 491,- Skladem 1-2 ks Koupit Výjimečnou kvalitu obrazu, rychlé odezvy a výkonné kreativní funkce nabízí tato nová digitální zrcadlovka. Snímač CMOS s vysokým rozlišením zajišťují...
OLYMPUS E-450 KIT + objektiv EZ 14-42mm Digitální zrcadlovka 11.8 mil. bodů + objektiv EZ-1442 14-42mm/ F3.5-5.6  -14% 9 316,- 6 665,- s DPH 7 998,- Skladem > 5 ks Koupit Olympus E-450 je dokonalou kombinací kvality digitálních zrcadlovek s výhodami kompaktních fotoaparátů. Model E450 je nástupcem dodnes velmi...	PENTAX K-x KIT black + objektiv D-AL 18-55mm Digitální zrcadlovka 12.4 mil. bodů, 2.7" LCD, 4x AA, SD/ SDHC  -5% 13 666,- 10 794,- s DPH 12 953,- Skladem 3-5 ks Koupit Špičková digitální zrcadlovka se senzorom 12 MP CMOS má dostatečně malé tělo na to, aby ste si ji zaměnili s kompaktem, ale narozdíl od kompakty...	CANON EOS 7D + 15-85 IS Digitální zrcadlovka - 18 mil. bodů CMOS, 2x DIGIC IV, RAW, 3" LCD, HDMI, FullHD video, CF, objektiv 15-85mm (F3.5-5.6) + ZDARMA Software Adobe Photoshop Elements 8 + ZDARMA Hrací karty 007 - všechny filmy  -12% 56 394,- 41 242,- s DPH 49 490,- Skladem 1-2 ks Koupit Špičková profesionální digitální zrcadlovka Canon EOS 7D nabízí vysoké rozlišení, výbornou rychlost zpracování a navíc pohodlné a kvalitní...

Obrázok 10. Dokument zobrazujúci viacero náhl'adov záznamov

Dokumenty zobrazujúce detaily jedného záznamu vo veľkej väčšine používajú zoskupovanie atribútov, čím umožňujú užívateľovi lepšiu orientáciu v dokumente. Od typu domény, ktorej sa web venuje, sa odvíja typ použitého zoskupovania.

Jedným z variantov zoskupovania je použiť tabuľku s plochými dátami (viď Obrázok 11.) a priradiť jej nejaký titulok či nadpis. Každý riadok tabuľky tak obsahuje názov atribútu v prvom stĺpci a jeho hodnotu v druhom.

Podrobnejšie informácie	
Typ telefonu	Klasický
Kategorie	Manažerský
Rozmery (v × š × h)	119 × 63 × 13 mm
Hmotnosť	135 g
Pamäť telefonu	1 000 MB
Pamät'ová karta	mikroSD
Základní barvy	bílá, černá
Operační systém	Android
Zvýšená odolnost	NE
Výměnné kryty	NE
Baterie typ / kapacita	Li-Pol / 1 500 mAh
Doba hovoru / pohotovostní režim	10 hodin / 17 dnů

Obrázok 11. Dokument zobrazujúci detail záznamu, zoskupené dáta v tabuľke

Ďalším variantom zoskupovania (viď Obrázok 12.) je umiestniť nadpis zoskupenia do prvého stĺpca a zoskupené atribúty vypísať v druhom stĺpci, za použitia nejakej ďalšej štruktúry, ako je napríklad zoznam (značky). Obdobou tohto postupu je zobraziť názvy atribútov v druhom stĺpci a ich hodnoty v treťom.

street price (body only)	• US: \$ 2,699 • EU: € 2,499 • UK: £ 2,299
List price (EF 24-105 L kit)	• US: \$ 3,499 • EU: € 3,299 • UK: £ 3,049
Body material	Magnesium alloy
Sensor *	• 36 x 24 mm CMOS sensor • Full 35 mm size frame • RGB Color Filter Array • Built-in fixed low-pass filter (with self-cleaning unit) • 22.0 million total pixels • 21.1 million effective pixels • 3:2 aspect ratio
Lenses	• Canon EF lens mount (does not support EF-S lenses) • No field of view crop (1.0x)
Dust reduction	• "EOS Integrated Cleaning System" • Self-cleaning sensor unit (filter in front of sensor vibrates at high frequency at start-up and shutdown - can be disabled)

Obrázok 12. Detail záznamu, zoskupené dáta v bunke tabuľky

Posledným variantom zoskupovania atribútov pri zobrazovaní detailu záznamu (viď Obrázok 13.) je použiť voľný text a nadpis zoskupenia v ňom nejakým spôsobom odlíšiť (napríklad pomocou značiek pre nadpis <h1> až <h6>, či značky pre tučné písmo). Zobrazovaný text je tak kombináciou značiek <p>, <div>, , , <h1> - <h6>.

Parametry a specifikace:
Obrazový snímač:
Typ: CMOS 36 x 24 mm
Rozlišení: 22 milionu pixelů
Efektivní počet bodů: 21,1 milionů pixelů
Poměr stran: 3 : 2
Čištění snímače: Integrovaný čistící systém EOS s fluorovou povrchovou vrstvou
Typ barevného filtru: Primární barvy
Zaostřování:
Typ: TTL-CT-SIR
Režimy zaostření: inteligentní AF, jednosnímkový AF, inteligentní AF-servo
Rozsah citlivosti: EV -0,5-18 (při 20 °C a ISO 100)
Ostřicí oblasti: 9bodový AF (a 6 pomocných bodů AF)
Výběr oblasti ostření: automatické a manuální

Obrázok 13. Časť detailu záznamu, dáta zoskupené textovo

Zobrazovanie do tabuľky je typické pre weby so špecifickejšou doménou (napr. www.mobil.idnes.cz venujúci sa recenziám mobilných telefónov). Tento fakt súvisí s tým, že tieto weby majú svoje databázy navrhnuté tak, aby mohli podrobne informovať o jednom type entity z svojej domény. Na druhej strane weby so širšou doménou (napr. www.alza.cz, internetový obchod ponúkajúci široké množstvo elektroniky) nemôžu mať pre každý typ entity zo svojej domény osobitnú tabuľku v databáze a transformáciu, ktorou ju zobrazujú na webe, pretože databáza by tak musela mať stovky tabuliek, čo je náročné nielen na ich správu ale aj dotazovanie. Tento zoskupený obsah je uložený v stĺpci tabuľky odpovedajúcej entite a porušuje tak prvú normálnu formu.

Ďalšou odpozorovanou skutočnosťou je, že pre každú hodnotu atribútu je v HTML dokumente zobrazovaný aj jeho názov, ktorý je vizuálne odlíšený od dát. Najčastejším typom vizuálneho odlíšenia metadát od dát je použitie tučného písma. V prípade tabuľky sa tento názov nachádza buď v prvom stĺpci rovnakého riadku ako je hodnota jeho atribútu, alebo v záhlaví tabuľky v rovnakom stĺpci (nie však nutne prvom riadku). V prípade zobrazenia v inej štruktúre ako v tabuľke sa často používa ešte dvojbodka, ktorá nasleduje hneď za názvom atribútu.

Dôležitým faktorom pri extrakcii dát z webu, ktorý je nutné zohľadniť je hustota dát v tabuľke. V prípade rozvrhov (viď Obrázok 14.) je tabuľka riedko vyplnená. V takomto prípade je pri extrahovaní dôležité uvedomiť si fakt, že cieľové dáta sa nenachádzajú na rovnakých pozíciách tabuľky, ale sú to práve neprázdné políčka tabuľky (resp. políčka obsahujúce nejaké zmysluplné hodnoty)

Film	Screen	Rating	Lang.	Mins.	«	11	12	13	14	15	16	17	18	19	20	21	22	»
Slovanský dům																		
Devil	8	15	Sub	80							16:15				20:20		22:15	
Flickan som lekte med elden	3	15	Sub	129			12:30			15:10		17:50			20:30			
Piko	4	12	DOrig	80											20:10		22:05	
Alpha and Omega 3D	6	G	DD	88				13:00		15:00								
Despicable Me 3D	4	G	DD	95				13:50					18:05					
Despicable Me 3D	4	G	DOrig	95							16:00							
Eat Pray Love	7	G	Sub	140							16:20					21:20		
Legend of the Guardians: The Owls of Ga'Hoole 3D	1	G	DD	97				13:30										
Machete	9	18	Sub	105						15:05				19:40		21:55		
Občanský průkaz	10	12	CZ	137				13:20			16:10			19:00		21:50		
Přežít svůj život	9	12	Orig	105			12:45					17:20						
Red	5	12	Sub	115			12:00		14:25		16:50			19:20		21:45		
Román pro muže	8	12	CZ	100					14:00				18:10					
Saw 3D	1	15	DSub	90						15:40		17:45			20:00		22:10	
The Social Network	2	12	Sub	120			12:00		14:30			17:00		19:30			22:00	
Wall Street: Money Never Sleeps	6	12	DSub	127								17:00		19:50			22:35	
You Will Meet a Tall Dark Stranger	7	12	Sub	98			12:05		14:10					19:10				

Obrázok 14. Príklad riedko vyplnenej tabuľky - rozvrhu

5 Softvérová analýza

Táto časť diplomovej práce sa zaoberá analýzou z hľadiska vývoja softvérového diela. Zo štandardnej analýzy sú v nej zahrnuté: popis systému v podkapitole **5.1**, rámcová definícia stupňov požiadaviek na neho kladených v podkapitole **5.2 Požiadavky na systém** a ako posledná časť analýzy je uvedená **5.3 Štúdia uskutočniteľnosti**, ktorá má zodpovedať na otázku, či a za použitia ktorých komponentov, sa dá daný systém implementovať čo najefektívnejšie.

V tejto kapitole nie je uvedená časť prípadov použitia, detailné rozpracovanie funkčných a nefunkčných požiadaviek ani UML diagramy popisujúce systém. Príčinou ich absencie je experimentálny charakter tejto diplomovej práce, ako aj agilný postup vývoja softvéru zvolený pri vývoji tejto práce, ktorý je na tieto účely vhodnejší.

5.1 Popis systému

Úlohou systému je získavanie dát z dynamicky generovaných webových stránok technikou programovania demonštráciou (programming by demonstration). Takýto systém by mal umožniť užívateľovi ukázať, ako má systém postupovať pri zbieraní informácií z danej webovej stránky (s využitím obmedzenej sady operácií pomocou klávesnice, prípadne myši). Na základe takejto ukážky by systém mal odvodiť postup, ako získavať informácie na podobných stránkach. Implementovaný systém by mal dokázať pozbierať pre užívateľa relevantné informácie z podobných stránok, napríklad do tvaru jednoduchej tabuľky vhodnej na ďalšie strojové spracovanie.

5.2 Požiadavky na systém

Systém bude svojou funkcionalitou spĺňať základné, prípadne tiež pokročilé, požiadavky. Jedným z cieľov práce je snaha vyvinúť systém tak, aby sa pokiaľ možno čo najviac približoval ideálnym požiadavkám na neho kladeným.

Základné požiadavky na systém:

- spracovať validné stránky Strict verzie HTML neobsahujúce JavaScript ani AJAX
- jednoduchý spôsob naučenia sa - extrahovať dáta zo stránok majúcich rovnakú HTML štruktúru (rozvrhnutie)
- vytvoriť program na extrakciu dát za rozumnej užívateľovej pomoci

Pokročilé požiadavky na systém:

- spracovať validné stránky menej striktných verzií HTML (Transitional a Frameset)

- prispôsobivý spôsob naučenia sa - extrahovať dáta zo stránok majúcich veľmi podobnú HTML štruktúru (rozvrhnutie)
- učenie/predikcia už po prvom demonštrovanom zázname

Ideálne požiadavky na systém:

- spracovať nevalidné HTML stránky
- inteligentný spôsob naučenia sa - extrahovať dáta zo stránok majúcich značne odlišnú HTML štruktúru (rozvrhnutie)
- spracovať dynamické stránky obsahujúce JavaScript a AJAX
- naučenie funguje vždy pri minimálnej skúsenosti užívateľa

5.3 Štúdia uskutočniteľnosti

Pri práci s programom je potrebné pristupovať k HTML dokumentom uloženým na internete, zobrazovať ich a ponúknuť užívateľovi možnosť obmedzenej práce s textom v zobrazovanom HTML dokumente. Zakázané sú operácie mazania, či pridávania textu.

Implementovať za týmto účelom vlastný zobrazovač HTML dokumentov pri súčasnom množstve dostupných komponentov by bolo veľmi neefektívne. V systéme bude použitý a rozšírený nejaký už existujúci komponent, ktorý musí spĺňať nasledujúce požiadavky:

- použitie klávesnice a myši na označovanie textu v zabudovanom prehliadači,
- získať presný HTML podstrom dokumentu presne zodpovedajúci nejakému označenému textu v prehliadači a vedieť s ním pracovať pomocou DOM alebo HTML parsera,
- pre každý HTML podstrom dokumentu vedieť zobrazit' označený text v zabudovanom prehliadači.

5.3.1 Voľba vhodného prostredia

Hľadanie vhodných komponentov, ktoré budú použité pri vývoji systému, prebiehalo s ohľadom na požiadavky spomínané v predchádzajúcom odstavci. Základným krokom bola voľba platformy, ktorá poskytne vhodné API rozhranie tak, aby bolo možné zobrazovať HTML dokument a zároveň ľahko pristupovať ku kontextu užívateľských operácií a taktiež umožní simulovať obmedzenú prácu s textom za použitia kurzora v texte.

Medzi kandidátov na komponenty vhodné na rozšírenie, či priame použitie patrili:

1. Internetový prehliadač Mozilla Firefox, ktorý by bolo nutné rozšíriť vo forme pluginu.

2. Komponent WebBrowser, ktorý je štandardnou súčasťou .NET frameworku od spoločnosti Microsoft. Ide o wrapper nad internetovým prehliadačom Internet Explorer [14].
3. Opensource komponent C# WebBrowser wrapper tool⁶, ktorý je wrapperom pôvodného komponentu WebBrowser od spoločnosti Microsoft [15].
4. Platený komponent JxBrowser od spoločnosti TeamDev pre platformu Java⁷.

Ako najvhodnejšie riešenie, ktoré by malo umožniť dosiahnutie všetkých požiadaviek, bola zvolená druhá možnosť - rozšírenie komponentu WebBrowser, obsiahnutého v toho času najnovšom .NET frameworku od firmy Microsoft vo verzii 4.0. Dôvodov na jeho voľbu bolo hneď niekoľko:

- API rozhranie spĺňa väčšinu zadaných kritérií,
- nulové náklady na zaobstaranie a použitie komponentu,
- dobrá a prehľadná dokumentácia API,
- mnoho príkladov dostupných na internete.

Komponent JxBrowser nebol vybraný kvôli vysokým obstarávacím nákladom činiacim 1349 amerických dolárov [16]. Možnosť napísať vlastný plugin pre prehliadač Mozilla Firefox, by zahŕňala naučiť sa nový jazyk XUL a dôkladnejšie si naštudovať API rozhranie pre prácu s Firefoxom [17]. Možnosť číslo 3 je veľmi blízka zvolenému riešeniu. Nakoniec bol však uprednostnený štandardný WebBrowser, bez nutnosti inštalovať knižnice na podporu práce s pôvodným WebBrowser komponentom.

Súčasťou štúdie uskutočniteľnosti je overenie možnosti reálnej implementácie spomínaných požiadaviek vo forme prototypu programu. Až po splnení tejto prerekvizity je možné začať pracovať na implementácii požiadaviek systému.

⁶ dostupný na web stránke <http://www.codeproject.com/KB/miscctrl/csEXWB.aspx>

⁷ dostupný na web stránke <http://www.teamdev.com/store/jxbrowser/>

6 Algoritmy a dátové štruktúry

Táto kapitola sa zaoberá detailnejším návrhom a konkrétnou implementáciou algoritmov navrhnutých v kapitole **4.3 Základné schémy algoritmov**. Na začiatku je uvedený popis implementácie techniky programming by demonstration v podkapitole **6.1 PBD**, nasledovaný podkapitolami **6.2 DE-PBD** — extrakcia dát z webu technikou PBD a **6.3 UE-PBD** — extrakcia URL adries technikou PBD, v ktorých je objasnené ako využívajú techniku PBD na svoje účely.

V úvode každej z týchto podkapitol je vždy objasnený návrh konkrétneho algoritmu, nasledovaný zápisom dátových štruktúr a kódu algoritmu v pseudokóde so syntaxou podobnou jazyku C++. Na pochopenie tohto pseudokódu sa nevyžaduje znalosť jazyka C++, avšak znalosť základného konceptu objektovo orientovaného programovania je vítanou podmienkou na pochopenie. Ak v ňom nie sú definované niektoré procedúry, ich funkcia je považovaná za samovysvetľujúcu, alebo ozrejmenú v predošlom texte. Pre väčšiu názornosť sú algoritmy v tejto kapitole zapísané ako procedúra, ale v skutočnosti sú implementované asynchrónne — pomocou vyvolávania a spracovávanía udalostí. Táto forma je však ekvivalentná algoritmu zapísanému vo forme procedúry.

6.1 PBD

Táto podkapitola rozširuje základnú schému algoritmu PBD spomínanú v časti **4.3.1 Schéma techniky PBD**. V úvodnej časti **6.1.1** je priblížený návrh tohoto algoritmu. Nasleduje zápis algoritmu PBD a jeho dátových štruktúr v pseudokóde v časti **6.1.2 Dátové štruktúry algoritmu PBD**.

6.1.1 Návrh algoritmu

Základným problémom, ktorý vyvstáva pri návrhu algoritmu učenia technikou PBD, je, ako sa má systém naučiť vyberať vzory tak, aby bol potom schopný samostatne extrahovať dáta resp. URL. Odpoveďou na tento problém je heuristika.

Heuristika zodpovedá nejakej logike výberu (napríklad vyber každý n -tý odkaz zo vstupnej množiny). Heuristika po predložení vzoru dokáže určiť, či tento vzor (napríklad odkazy s indexmi 0, 3 a 6) zodpovedá jej logike výberu alebo nie. Ak áno, vypočíta si parametre (v tomto prípade $n=3$), a potom už bude schopná po predložení vstupnej množiny samostatne extrahovať dáta resp. odkazy. Ostáva však ešte dobre definovať sady heuristík tak, aby bol systém schopný efektívne extrahovať z čo najväčšieho množstva foriem dát. K tomuto účelu musí napomôcť silná znalosť domény.

PBD technika učenia sa prebieha nasledovne:

Po demonštrácii prvého vzoru systém učí všetky heuristiky rovnakého typu, ktoré má poznamenané v nejakom zozname. Učenie v každom kroku prebieha tak, že každej heuristike je predložený vzor a množina, z ktorej tento vzor bol vybraný.

Heuristika sa po predložení vzoru snaží vypočítať parametre popisujúce logiku výberu vzoru zo vstupnej množiny. Ak vzor nezapadá do logiky výberu, táto heuristika sa nedokáže naučiť, musí byť vyradená zo zoznamu heuristík a ďalej sa s ňou nepracuje. Pri vypočítavaní parametrov môže nastať prípad, kedy nie je schopná úplne overiť, že vzor bol vybraný touto logikou, pretože jej na to nebol predložený dostatočný počet vzorov. V takomto prípade nie je vyradená zo zoznamu heuristík, len je označená za zatiaľ nenaučenú. V prípade, že sa heuristike podarilo správne vypočítať parametre, je označená za naučenú.

V ďalšom kroku si heuristika po predložení ďalšieho vzoru vypočíta nové parametre, ktoré porovná s predošlými. Ak sa staré parametre líšia od nových, heuristika sa nedokáže prispôbiť a musí byť zo zoznamu logík odstránená.

Učenie sa končí:

- naučením heuristiky — ak po určitom počte krokov v zozname logík ostane nejaká nezamietnutá naučená logika, je označená za správnu a učenie logiky výberu sa končí.
- nemožnosťou sa naučiť — ak je zoznam heuristík prázdny, resp. žiadna v zozname neexistuje žiadna naučená heuristika počas niekoľkých po sebe nasledujúcich krokov.

Každej heuristike rovnakého typu je priradené nejaké ohodnotenie podľa jej obecnosti (napríklad heuristika vyberajúca HTML element podľa ID je menej obecná ako tá, ktorá vyhľadáva podľa relatívnej cesty). Toto ohodnotenie je v podkapitole **7.3 Obmedzenia** nazývané Poradie.

Po každej iterácii je zoznam naučených heuristík zotriedený podľa poradia obecnosti, kde na začiatku sú menej obecné heuristiky. Po úspešnom naučení má systém neprázdny zoznam heuristík, ktoré sa naučili demonštrovanej extrakcii a sú zotriedené začínajúc tou najmenej obecnou. Z nich si systém vyberie práve túto najmenej obecnú. To dáva výhodu pri extrakcii, kedy pri zlyhaní najmenej obcej heuristiky (t.j. nedokáže extrahovať), môže systém využiť záložné obcejšie heuristiky. V prípade, že sa v systéme táto záloha využívať nebude, je dobré zotriediť heuristiky opačne, pretože všeobecnejšia heuristika potom zlyhá pri extrakcii s menšou pravdepodobnosťou ako nejaká menej obecná.

Celý proces správy učenia (tým pádom aj zoznamu heuristík) vykonáva objekt v tejto práci nazývaný inštrukcia. Inštrukcia má dva zoznamy — jeden pre nenaučené heuristiky a druhý pre už naučené. Po inicializácii sú všetky heuristiky v zozname nenaučených. Po demonštrácii užívateľom inštrukcia predloží vzor a vstupnú množinu všetkým heuristikám v oboch zoznamoch. Agenti sa tak potom neskladajú z heuristík, ale inštrukcií a nemusia sa tak starať správu procesu učenia.

Algoritmy extrakcie dát a extrakcie URL technikou programovania demonštráciou využívajú túto techniku programovania demonštráciou. Každý z nich ju však mierne modifikuje podľa svojich potrieb.

6.1.2 Dátové štruktúry algoritmu PBD

6.1.2.1 Heuristika

Trieda `Heuristika` uvedená v tejto časti je najobecnejšou abstrakciou fungovania heuristiky v systéme WDE-PBD. Ostatné heuristiky implementované v systéme musia spĺňať a implementovať toto definované rozhranie.

```
class Heuristika
{
    object parametre;
    bool Naucena;
    bool NechopnaSaUcit;

    object VypocitajParametre(object vzor, object vstupnaMnozina);

    PrispodobSa(object vzor, object vstupnaMnozina)
    {
        parametre = VypocitajParametre(vzor, vstupnaMnozina);
    }

    object Extrahuj(object vstupnaMnozina);
}
```

6.1.2.2 Inštrukcia

Inštrukcia sa v programe chová rovnako ako heuristika s tým rozdielom, že obsahuje viacero heuristík. Pre algoritmus učenia PBD poskytuje správu zoznamu heuristík.

```
class Instrukcia
{
    Zoznam<Heuristika> vsetkyHeuristiky;
    Zoznam<Heuristika> nauceneHeuristiky;
    Heuristika vitaznaHeuristika;

    PrispodobSa(object vzor, object vstupnaMnozina)
    {
        nauceneHeuristiky.Vymaz();
        foreach (Heuristika heuristika in vsetkyHeuristiky)
        {
            heuristika.PrispobSa(vzor, vstupnaMnozina);
            if (heuristika.Naucena)
            {
                nauceneHeuristiky.pridaj(heuristika);
            }
            if (heuristika.NechopnaSaUcit)
            {
                vsetkyHeuristiky.vymaz(heuristika);
            }
        }
    }
}
```

```

        {
            vsetkyHeuristiky.odstran(heuristika);
        }
    }
    ZotriedPodlaPoradiaObecnosti(nauceneHeuristiky);
    vitaznaHeuristika = nauceneHeuristiky[0];
}

object Extrahuj(object vstupnaMnozina)
{
    return vitaznaHeuristika.Extrahuj(vstupnaMnozina);
}
}

```

6.1.3 Algoritmus PBD

```

Instrukcia PBD()
{
    Instrukcia instrukcia = new Instrukcia();

    do
    {
        object vzor = CakajNaUzivatelomDemonstrovanyVzor();
        Kontext kontext = VypocitajKontext(vzor);
        instrukcia.PrisposobSa(vzor, kontext);
    } while (!instrukcia.KoniecUcenia);

    if (instrukcia.Naucena)
    {
        return instrukcia;
    }
}

```

6.2 DE-PBD

Táto podkapitola sa venuje detailnejšiemu popisu algoritmu extrakcie dát technikou programovania demonštráciou DE-PBD. Podkapitola vychádza zo základnej schémy algoritmu uvedenej v časti **4.3.2 Schéma algoritmu extrakcie dát**. Úvodná časť podkapitoly **6.2.1 Návrh algoritmu** popisuje, ako tento algoritmus využíva techniku PBD a uvádza dôvody, ktoré viedli ku konkrétnym rozhodnutiam pri jeho implementácii. V ďalšej podkapitole **6.2.1.1 Dátové Heuristiky** je potom objasnený návrh dátových heuristik, ich skladba a možnosti kombinovania tak, aby bol systém schopný extrahovať dáta z čo najväčšieho počtu foriem.

6.2.1 Návrh algoritmu

Pri štandardnom postupe WDE sa najprv získajú URL vstupných dokumentov, ktoré sú následne v druhom kroku spracovávané či už pomocou nejakého HTML parsera, DOMu alebo použitím textových funkcií, ako napríklad grep či regulárnych výrazov. Postup, ktorým sa tieto nástroje dopracujú k výstupnej množine atribútov z každej predloženej HTML stránky, je dvojaký:

1. buď sa pre každý atribút zvlášť vyhľadáva jeho výskyt v dokumente,
2. alebo sa priebežne narába s dokumentom, orezáva text, delí na dielčie časti, z ktorých sa ľahšie extrahujú jednotlivé atribúty.

Ak sa snažíme aplikovať PBD na tento problém zistíme, že je nutné zvoliť prvý uvedený spôsob, pretože nemáme zadefinované editačné operácie nad vizualizáciou HTML dokumentu. Aj podľa [2] je WDE označované ako Web Scraping, kde anglické slovo scraping v preklade znamená vystrihávanie — teda v istom prenesenom zmysle necháme užívateľa vystrihnúť zo stránky dáta. Pri postupe v bode 1. je pre každý atribút definovaná nejaká procedúra, ktorá extrahuje z HTML dokumentu jeho hodnotu. Algoritmus DE-PBD sa v podstate snaží nájsť túto procedúru⁸ za pomoci užívateľa a jeho silných znalostí domény WDE.

Výstupom učenia v algoritme DE-PBD je agent, ktorého heuristiky odpovedajú procedúram na extrakciu jednotlivých atribútov. Algoritmus sa tak snaží hľadať pre každý atribút procedúru extrakcie jeho hodnoty a jej vhodné parametre nasledovne:

Potom, čo užívateľ označil text a zvolil príkaz extrahovať, sa vykonáva technika programovania demonštráciou navonok bez nejakého väčšieho rozdielu oproti zvolenému postupu PBD. Systém má po označení k dispozícii pointer na presné miesto (HTML značka) v HTML strome, v ktorom sa tento označený text nachádza. Kľúčovým bodom návrhu je uvedomiť si, že kontext je treba rozdeliť na dve menšie časti — HTML kontext elementu

⁸ Táto procedúra je v tejto práci označovaná ako heuristika.

obsahujúceho označenie a kontext označeného textu vzhľadom k celému textu tohto elementu. Kontextu označenia tak je možné priradiť veľmi veľa informácií popisujúcich polohu označenia v kontexte celého HTML dokumentu. Návrh heuristík tesne súvisí s návrhom kontextu a je bližšie popísaný v nasledujúcej časti.

V ďalších iteráciách učenia, keď už existuje nejaká naučená heuristika, ktorá dokáže extrahovať dáta pre tento konkrétny atribút, ponúka systém užívateľovi možnosť zobrazíť si označenie, ktoré by táto heuristika vrátila po spustení na aktuálny HTML dokument.

Učenie sa končí štandardne podľa popísanej techniky PBD — naučením v prípade, že existuje naučená heuristika v niekoľkých po sebe idúcich krokoch.

6.2.1.1 Dátové Heuristiky

Návrh fungovania dátovej heuristiky je robený tak, aby bolo možné naučiť sa extrahovať všetky možné formy prezentovaných dát podľa analýzy v časti **4.4.4 Formy prezentácie dát**.

Pripomeňme si definíciu dátovej heuristiky vo WDE-PBD:

Dátová heuristika — heuristika, ktorá má za úlohu predpovedať označenie v HTML dokumente určené na extrakciu na základe znalosti predošlých extrakcií a ich kontextov. Jej vstupom je HTML dokument a výstupom je označenie.

Pri hlbšom zamyslení sa nad fungovaním dátovej heuristiky je zrejmé, že heuristika nemôže pracovať len s čistým textom HTML dokumentu (ako napríklad heuristika v systéme TELS, ktorá si všima vzdialenosť pozície kurzora/označenia vzhľadom k začiatku a koncu každého typu lexikálneho elementu [10]). Takáto heuristika by v prípade použitia na celý text HTML dokumentu nebola účinná, pretože súčasné webové stránky obsahujú množstvo dynamicky sa meniaceho obsahu ako je napríklad reklama, ktorý môže byť dokonca unikátny pre každú užívateľskú požiadavku, čo by mohlo spôsobiť veľké rozdiely v obsahu textu HTML dokumentu. Preto je pre HTML dokument nutné uvedenú lexikálnu štruktúru predefinovať:

1. Zaviesť nový lexikálny element — HTML značku.
2. Zrušiť lexikálny element odstavca (pretože v HTML dokumente existuje mnoho značiek za účelom oddeľovania blokov textu).
3. Predefinovať lexikálny element nového riadku ako značku `<br />`.
4. Ostatné elementy slovo a znak ostávajú nezmenené.

Táto nová lexikálna štruktúra zohľadňuje definíciu kontextu označeného textu. HTML kontext sa určuje vzhľadom k lexikálnym elementom z bodu 1 a textový kontext vzhľadom k lexikálnym elementom z bodov 3. a 4. Na text HTML elementu obsahujúceho označenie tak už je možné použiť spomínanú textovú heuristiku. Teraz je ešte potrebné zamerať sa na HTML štruktúru a vytvoriť HTML heuristiky, ktoré budú akousi HTML obdobou textových heuristík.

Každá dátová heuristika v systéme sa preto skladá z 2 čiastkových heuristík na seba nadväzujúcich. Heuristika použitá ako prvá v poradí vyhľadáva v HTML strome a vracia nejaký jeho element. Týmto sa priestor na hľadanie hodnoty atribútu v HTML dokumente výrazne zúži. Na tento "výrez" stromu už potom je možné použiť textovú heuristiku.

Otázkou je ako učiť dátovú heuristiku technikou PBD. V zásade sa núkajú dve alternatívy:

1. Učiť všetky HTML heuristiky, kde každá by v sebe obsahovala zoznam všetkých textových heuristík, ktoré by rekurzívne učila. V prípade neschopnosti textových heuristík naučiť sa by sa HTML heuristika tiež nedokázala naučiť a musela by byť zo zoznamu vyradená. Naučená HTML heuristika by sa potom stala výslednou dátovou heuristikou.
2. Učiť všetky HTML heuristiky oddelene od textových heuristík a v prípade, že jeden z typov nie je schopný naučiť sa, skončiť proces učenia

Variant č. 1. je ekvivalentný s variantom č. 2., čo vyplýva z návrhu kontextu. V systéme je zvolený variant č. 2., ktorý je ekvivalentom 1., ale je menej pamäťovo náročný.

Po naučení má dátová heuristika v sebe zakomponované správne čiastkové heuristiky so správnymi parametrami a v exekučnej časti algoritmu jej stačí poskytnúť iba HTML dokument, z ktorého dokáže samostatne extrahovať hodnotu atribútu, pre ktorý je určená.

V častiach **6.2.1.1.1 HTML heuristiky** a **6.2.1.1.2 Textové heuristiky** je popísané aké typy heuristík je možné použiť na nájdenie elementu resp. textu a v záverečnej časti **6.2.1.1.3 Kombinovanie heuristík** je popísané ako zložiť jednotlivé heuristiky do výsledných, aby bola vzniknutos sada heuristík, čo najuniverzálnejšia.

6.2.1.1.1 HTML heuristiky

Predpovedanie polohy resp. hodnoty ďalšieho atribútu sa v prvom kroku opiera o HTML heuristiky, ktoré sa snažia vo svojom návrhu zohľadniť skutočnosti zmienené v častiach **4.4.1 Typy webov a techniky generovania HTML dokumentov** a **4.4.2 Kompozícia webu a stránky**. Napríklad v ASP .NET má za určitých podmienok každý element unikátny identifikátor, ktorý je nemenný pre rovnakú šablónu.

HTML Heuristiky môžu vyhľadávať element, obsahujúci extrahovaný atribút nasledujúcimi spôsobmi:

1. "Zvrchu" podľa cesty⁹ k elementu, ktorá má:
 - 1.1. odsadenie každej značky od začiatku/konca rodičovského elementu definované zvlášť sprava alebo zľava,
 - 1.2. jeden pevný element odsadený o jeden viac ako v predošlej ceste.
2. "Zospodu" podľa názvu elementu a vlastností podstromu:
 - 2.1. celkový počet elementov,
 - 2.2. poradie značiek potomkov koreňa.

⁹ Cesta je postupnosť HTML značiek od elementu <body> k danému elementu.

3. "Priamo" podľa identifikátora elementu.

Jednoznačnosť nájdeného výsledku v postupoch 1. a 3. je v niektorých prípadoch aj na škodu. Napríklad, ak by sa systém spoliehal na vyhľadávanie podľa konkrétneho identifikátora a programátor webu by zmenil jeho názov, stala by sa heuristika neplatnou. Na to, aby sa systém dokázal vyrovnat' s malými zmenami v štruktúre šablóny stránky, ak už sa ju naučil rozpoznávať, by bolo dobré uprednostňovať heuristiky, ktoré sa napríklad nespoliehajú na jedinečnosť identifikátorov, ale preferujú skôr nájdenie elementu na základe nejakých obecných vlastností kontextu označenia.

Vezmime si do úvahy prípad, kedy sa agent naučí vyhľadávať podľa presnej cesty a po istom čase sa zmení štruktúra HTML stránky, napríklad by na webe pribudla reklama v záhlaví. Odsadenie elementu na začiatku cesty by sa zvýšilo a cesta k elementu s dátami taktiež. Táto heuristika by potom zlyhala vo svojej úlohe nájsť dáta. Ak by sme však mali špekulatívnu heuristiku, ktorá by síce vyhľadávala podľa cesty, ale v prípade nenájdenia výsledku by ignorovala odsadenie, jej výsledok by mohol byť správny.

6.2.1.1.2 Textové heuristiky

Textové heuristiky sa dostávajú vo vykonávaní celkovej heuristiky na rad ako druhé. Ich úlohou je z vyextrahovať správny reťazec zodpovedajúci cieľovému atribútu z textu elementu vráteného HTML heuristikou.¹⁰ Ďalším faktom, ktorý si treba uvedomiť je, aký typ textu si majú textové heuristiky všímať. Bud' sa zamerajú na prostý text elementu (všetko mimo HTML značiek), alebo môžu HTML značky v ňom vnorené považovať taktiež za text. Keďže vstupom textovej heuristiky je text a jej výstupom tiež text, je možné ich skladať do komplexnejších.

Textové heuristiky môžu vyhľadávať označenie v texte elementu rôznymi spôsobmi, tomu zodpovedá ich členenie:

1. Textové heuristiky

1.1. orezávajúce text:

1.1.1. zľava a sprava daným počtom znakov/slov/riadkov,

1.1.2. zľava/sprava daným počtom znakov/slov/riadkov a z orezaného textu vráti podreťazec dlhý zadaný počet znakov/slov/riadkov;

1.2. vyhľadávajúce zadaný reťazec podľa:

1.2.1. presnej hodnoty,

1.2.2. typu hodnoty (napríklad číslo, mena), alebo

1.2.3. regulárneho výrazu.

2. HTML textové heuristiky:

2.1. vráť obsah vymedzený koncom jednej a začiatkom nasledujúcej značky.

¹⁰ Fungovať by mohli aj samostatne na texte celého dokumentu, avšak vrátený výsledok v tomto prípade nemusí byť (a s veľkou pravdepodobnosťou ani nebude) správny.

Napríklad: v HTML texte "<div>parameter1 hodnota1
</div>" nájde HTML textová heuristika reťazec "hodnota1" vymedzenú značkami a </div>

6.2.1.1.3 Kombinovanie heuristík

Cieľom je vyvinúť, pokiaľ možno, čo najuniverzálnejšiu sadu heuristík tak, aby dokázali extrahovať dáta z čo najviac foriem. Nasledujúci zoznam ukazuje pre každú formu prezentácie dát typickú kombináciu výslednej heuristiky v systéme WDE-PBD:

a) Ploché dáta v tabuľke:

- **HTML heuristika:** vyhľadávajúca "Zvrchu" podľa cesty k elementu. (1)
- **textová heuristika:** orež text zľava a sprava daným počtom znakov/slov/riadkov. (1.1.1)

b) Štruktúrované dáta v tabuľke:

- **HTML heuristika:** vyhľadávajúca "Zvrchu" podľa cesty k elementu "Zvrchu" podľa cesty k elementu, ktorá má:. (1)
- **textová heuristika:** orež text zľava/sprava daným počtom znakov/slov/riadkov a z orezaného textu vráti podreťazec dlhý zadaný počet znakov/slov/riadkov (1.1.2)

c) Štruktúrované dáta:

- **HTML heuristika:** vyhľadávajúca "Zvrchu" podľa cesty k elementu. (1)
- **kombinovaná textová heuristika:**
 - **HTML textová heuristika:** vráť obsah vymedzený koncom jednej a začiatkom nasledujúcej značky. (2.1)
 - **textová heuristika:** orež text zľava/sprava daným počtom znakov/slov/riadkov a z orezaného textu vráť podreťazec dlhý zadaný počet znakov/slov/riadkov. (1.1.2) zľava/sprava daným počtom znakov/slov/riadkov a z orezaného textu vráti podreťazec dlhý zadaný počet znakov/slov/riadkov;

V prípadoch a) a b) je použitá HTML heuristika podľa toho, či jeden dokument obsahuje

- jeden záznam — odsadenie každej značky od začiatku/konca rodičovského elementu definované zvlášť sprava alebo zľava (odsadenie každej značky od začiatku/konca rodičovského elementu definované zvlášť sprava alebo zľava, 1.1) vyhľadaj element podľa cesty, ktorá má odsadenie každej značky od začiatku/konca rodičovského elementu definované zvlášť sprava alebo zľava,
- viacero záznamov — vyhľadaj element podľa cesty, ktorá má jeden pevný element odsadený o jeden viac ako v predošlej ceste (1.2).

6.2.2 Dátové štruktúry algoritmu DE-PBD

6.2.2.1 Heuristiky

Základné triedy, ktoré sa dokážu učiť a extrahovať text. Uvedené triedy obsahujú len rámcový zápis akú funkcionálnosť musí heuristika ponúkať. Implementované heuristiky tak musia dediť od týchto dvoch a ich zoznam si môžete pozrieť v **7.3 Obmedzenia**.

```
class HTMLHeuristika
{
    HTMLElement NajdiElement(HTMLDokument dokument);
    object PrispособSa(HTMLElement vzor, HTMLDokument vstupnaMnozina);
}

class TextovaHeuristika
{
    OznacenyText NajdiText(HTMLElement element);
    void PrispособSa(OznacenyText vzor, HTMLElement vstupnaMnozina);
}
```

6.2.2.2 Dátová inštrukcia

Dátová inštrukcia má za úlohu extrahovať dáta pre jeden atribút z HTML dokumentu. Skladá sa z HTML inštrukcie a z textovej inštrukcie, ktoré majú rovnakú funkciu ako [HTMLHeuristika](#) a [TextovaHeuristika](#).

```
class DatovaInstrukcia
{
    Instrukcia htmlInstrukcia;
    Instrukcia textovaInstrukcia;
    bool NeschopnaSaUcit;
    bool Naucena;

    public void PrispособSa(OznacenyText vzor, HTMLDokument dokument)
    {
        htmlInstrukcia.PrispособSa(vzor.kontext.RodicovskaHTMLZnacka, dokument);
        textovaInstrukcia.PrispособSa(vzor.text, vzor.kontext.RodicovskaHTMLZnacka.Text);
    }

    string Extrahuj(HTMLDokument dokument)
    {
        HTMLElement element = htmlInstrukcia.NajdiElement(dokument);
        OznacenyText oznacenie = textovaInstrukcia.NajdiText(element);
        return oznacenie.text;
    }
}
```

6.2.2.3 Data agent

Data agent sa skladá z inštrukcií, ktoré extrahujú hodnoty jednotlivých atribútov z HTML dokumentu. Učenie data agenta prebieha v algoritme v nasledujúcej časti, jeho činnosť extrakcie je popísaná tu.

```
class DataAgent
{
    public Zoznam<DatovaInstrukcia> Instrukcie;

    public void Extrahuj(HTMLDokument dokument, out Zoznam<Zoznam<string>> data)
    {
        int i = 0;
        foreach (DatovaInstrukcia instrukcia in Instrukcie)
        {
            string hodnota = instrukcia.Extrahuj(dokument);
            data[data.Pocet - 1][i] = hodnota;
            i++;
        }
    }
}
```

6.2.3 Algoritmus DE-PBD

Základná osnova algoritmu DE-PBD je:

1. Inicializuj pomocné premenné
2. Uč agenta/dátové inštrukcie pre každý atribút
3. V prípade naučenia agenta, ho spusti na zvyšok vstupu

```
DEPBD(  
    Zoznam<string> URL  
    , bool jedenZaznamVDokumente  
    , int dlzkaUcenia  
    , out Zoznam<Zoznam<string>> data  
    , out DataAgent dataAgent  
)  
{  
    // zoznam premennych vyuzivanych v algoritme  
    Prehliadac prehliadac = new Prehliadac();  
    int aktualnyAtribut = 0;  
    int pocetAtributov = 0;  
    int aktualnyZaznam = 0;  
    Zoznam<DatovaInstrukcia> instrukcie = new Zoznam<DatovaInstrukcia>();  
    OznacenyText oznacenie;  
    bool koniecUcenia = false;  
    bool uspesnaPredpovedCelehoZaznamu = false;  
    int pocetSpravnychPredpovediCelehoZaznamu = 0;  
  
    // pocas demonstracie 1. zaznamu sa zistuje pocet atributov a vytvaraju instrukcie  
    prehliadac.nacitaj(URL[aktualnyZaznam]);  
    while (!UzivatelHlasiKoniecZaznamu())  
    {  
        oznacenie = CakajNaOznacenieTextuUzivatelom();  
        data[aktualnyZaznam][aktualnyAtribut] = oznacenie.text;  
        instrukcie.pridaj(new Instrukcia());  
        instrukcie[aktualnyAtribut].PrisposobSa(oznacenie, prehliadac.HTMLDokument);  
        aktualnyAtribut++;  
    }  
    pocetAtributov = aktualnyAtribut;  
  
    // proces ucenia  
    do  
    {  
        aktualnyZaznam++;  
        if (jedenZaznamVDokumente) prehliadac.nacitaj(URL[aktualnyZaznam]);  
        uspesnaPredpovedCelehoZaznamu = true;  
  
        for (int atribut = 0; atribut < pocetAtributov; atribut++)  
        {  
            DatovaInstrukcia instrukcia = instrukcie[aktualnyAtribut];  
            oznacenie = CakajNaOznacenieTextuUzivatelom();  
            data[aktualnyZaznam][aktualnyAtribut] = oznacenie.text;  
            instrukcia.PrisposobSa(oznacenie, prehliadac.HTMLDokument);  
            if (instrukcia.NeschopnaSaUcit) return;  
            if (!instrukcia.Naucena) uspesnaPredpovedCelehoZaznamu = false;  
        }  
  
        if (uspesnaPredpovedCelehoZaznamu) pocetSpravnychPredpovediCelehoZaznamu++;  
        else pocetSpravnychPredpovediCelehoZaznamu = 0;  
  
        koniecUcenia = !(uspesnaPredpovedCelehoZaznamu
```

```

        && pocetSpravnychPredpovediCelehoZaznamu > dlzkaUceni);
    } while (!koniecUceni);

    // uloz agenta
    dataAgent.Instrukcie = instrukcie;

    // spusti agenta na zvysoke vstupu
    for (int i = aktualnyZaznam; i < URL.Pocet; i++)
    {
        prehliadac.naviguj(URL[i]);
        dataAgent.Extrahuj(prehliadac.HTMLDokument, out data);
    }
}

```

6.3 UE-PBD

Táto podkapitola sa venuje širšiemu popisu algoritmu extrakcie URL adres technikou programovania demonštráciou UE-PBD, ktorý vychádza z časti **4.3.3 Schéma algoritmu**. Úvodná časť podkapitoly **6.3.1 Návrh algoritmu** sa opiera o existenciu a vlastnosti navigačných stránok plynúcu z analýzy v **4.4.3 Navigácia užívateľa k dátam** a popisuje ako konkrétne využíva algoritmus UE-PBD techniku programovania demonštráciou v častiach **6.3.1.1 Získavanie navigačných stránok** a **6.3.1.2**

Získavanie zoznamu URL z navigačnej stránky za účelom automatizovať extrakciu URL adries.

6.3.1 Návrh algoritmu

Proces získavania URL vedúcich na extrahované záznamy je činnosť, ktorá si v niektorých prípadoch vyžaduje opakované použitie. Napríklad v internetovom obchode sa mení ponuka tovaru v nepravidelných intervaloch. Užívateľ, ktorý chce získať dáta o aktuálnej ponuke nových produktov, tak musí získať aktuálny zoznam URL dokumentov, ktoré tieto informácie obsahujú. Preto je vhodné zber URL adries taktiež automatizovať.

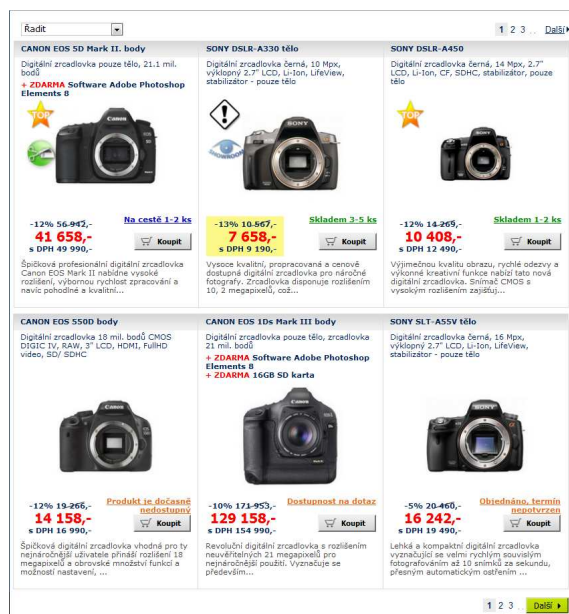
Navigačné stránky (viď Obrázok 15. Časť navigačnej stránky) sú už zo svojej podstaty prirodzeným zdrojom, z ktorého možno získať všetky URL vedúce na extrahované záznamy. Pri štandardnom riešení WDE sa používajú nasledujúce kroky:

1. Získanie zoznamu všetkých navigačných stránok.
2. Pre každú navigačnú stránku sa z nej potom extrahujú URL odkazujúce na stránky so vstupnými záznamami.

Algoritmus UE-PBD bude fungovať obdobne, na oba kroky použije užívateľa, od ktorého sa naučí, ako získať zoznam všetkých navigačných stránok a ako z každej stránky získať spomínané URL záznamy technikou PBD.

Vstupom učiaceho algoritmu je východisková stránka a demonštrácia užívateľom. Výstupom sú URL a agent, ktorý je tvorený dvoma časťami: heuristikou výberu navigačných stránok a heuristikou výberu URL z navigačnej stránky.

Pri automatizovanom získavaní URL je potom použitý tento naučený agent, ktorého vstupom je východiskový HTML dokument a výstupom aktuálny zoznam URL.



Obrázok 15. Časť navigačnej stránky

6.3.1.1 Získavanie navigačných stránok

Ak nehľadáme na technológie, ktoré tieto štandardné WDE riešenia používajú, môžeme určiť viacero postupov, ktorými získavajú zoznamy navigačných stránok. Najpoužívanejšie tri sú:

1. Odpozorovaním existencie šablóny v URL navigačnej stránky je možné po získaní prvej a poslednej URL vygenerovať rad URL, ktoré sú navigačnými stránkami.
Napríklad: navigačné stránky pre fotoaparáty sú:
<http://www.alza.cz/digitalni-foto/18843129-p1.htm>
<http://www.alza.cz/digitalni-foto/18843129-p2.htm>
...
<http://www.alza.cz/digitalni-foto/18843129-p31.htm>
2. Extrahovaním URL z HTML odkazu, s funkciou "Ďalšia stránka", ktorý vedie na navigačnú stránku nasledujúcu za aktuálnou dovtedy, až kým sa na aktuálnej stránke tento odkaz nevyskytuje.
3. Ručným definovaním zoznamu stránok, v prípade malého počtu stránok.

Metóda z bodu 1. sa nemôže použiť v prípade, že URL navigačných stránok nemá túto šablónu, a musí sa použiť metóda z bodu 2. Tá však má tiež zásadný nedostatok, ktorým je fakt, že odkaz resp. tlačidlo s funkciou "Ďalšia stránka" nemusí existovať, pretože existujú (viď Obrázok 16. Príklad stránkovacej oblasti) minimálne dva priame¹¹ spôsoby ako môže užívateľ prejsť použitím stránkovacej oblasti z aktuálnej navigačnej stránky na nasledujúcu¹²:

- použitím odkazu resp. tlačidla s funkciou "Ďalšia stránka"

¹¹ tých nepriamych je veľa, napríklad z prvej stránky užívateľ naviguje na tretiu a tak na druhú atď.

¹² ak pomineme počet stránkovacích oblastí v navigačnej stránke

- použitím číslovaného odkazu, ktorý má číslo nasledujúce za číslom priradeným aktuálnej



Obrázok 16. Príklad stránkovacej oblasti

Ďalšou nevýhodou pri použití štandardného riešenia WDE je, že pre každý web je nutné naprogramovať osobitý skript, keďže HTML štruktúra ich navigačných stránok ako aj logika stránkovania môžu byť a aj zvyčajne sú úplne odlišné.

UE-PBD sa inšpiruje štandardnou WDE technikou z bodu 2., ale jeho ideou je použiť užívateľa na navigáciu na ďalšiu stránku a postup, ktorým ju demonštruje sa naučiť. Systém sa snaží odvodiť, ktorý typ prechodu užívateľ zvolil, nájsť v ňom nejakú logiku.

Algoritmus učenia sa výberu ďalšej navigačnej stránky funguje nasledovne:

Po tom čo užívateľ klikol na odkaz na ďalšiu stránku, musí v prvej iterácii systém detekovať stránkovaciu oblasť. Cestu značiek k oblasti si uloží agent, aby ju potom pomocou tejto cesty vedel vyhľadať v ďalšom procese učenia a extrakcie. Heuristikám je potom predložený kliknutý odkaz a zoznam odkazov v stránkovacej oblasti na prispôsobenie si svojich parametrov. Každá heuristika si zároveň ukladá index kliknutého odkazu a jeho kontext (počet odkazov v stránkovacej oblasti) do histórie. Po učení heuristik systém naviguje na tento kliknutý odkaz, kde pokračuje učenie ďalšou iteráciou.

Problémom pri realizácii tejto metódy je, ako detekovať stránkovaciu oblasť v prvom kroku. Po tom, čo užívateľ klikol na odkaz, má systém len informáciu o mieste v HTML strome, kde sa tento odkaz nachádza a vie, že tento odkaz sa určite nachádza v stránkovacej oblasti¹³. Z týchto informácií neexistuje presný postup, ako sa k tejto oblasti dopracovať a navrhované riešenie je skôr heuristikou, ktorá sa opiera fakt, že počet odkazov v stránkovacej oblasti navigačnej stránky je číslo v rozsahu 3-50 a všetky tieto odkazy sa nachádzajú v HTML strome veľmi blízko seba. Systém určí stránkovaciu oblasť prechodom v strome od kliknutého linku smerom k rodičovi, až kým nie je počet odkazov v rodičovi v tomto stanovenom rozmedzí.

¹³ Pretože užívateľ nemá dôvod učiť systém nejaký úplne iný odkaz, ktorý vedie na nasledujúcu stránku ale nenachádza v stránkovacej oblasti, lebo takýchto je na stránke málo a za týmto účelom vývojári webov umiestňujú na navigačné stránky stránkovacie oblasti.

6.3.1.2 Získavanie zoznamu URL z navigačnej stránky

V druhom kroku získavania cieľových URL odkazujúcich na vstupné záznamy sa pri štandardnom WDE z každej navigačnej stránky extrahujú tieto URL z náhľadov záznamov. Pre každý element odpovedajúci náhľadu položky (viď Obrázok 17.) sa extrahuje odkaz na stránku s cieľovým záznamom, podľa cesty v HTML strome od elementu odpovedajúcom náhľadovej oblasti. Ide o pomerne jednoduchý spôsob, ktorý nemá nejaké závažnejšie nedostatky, keďže náhľady záznamov v oblasti náhľadov sú typicky generované pomocou jednej šablóny.

Algoritmus UE-PBD sa však nesústreďuje na polohu odkazu v HTML strome, ale funguje na rovnakom princípe ako učenie sa odkazu na ďalšiu stránku - systém sa snaží odvodiť logiku výberu odkazov na detaily záznamu zo všetkých odkazov nachádzajúcich sa v oblasti náhľadov.



Obrázok 17. Náhľad záznamu

Algoritmus učenia sa výberu odkazov na detaily záznamov funguje veľmi podobne ako učenie sa prechodu na ďalšiu stránku. Systém taktiež potrebuje určiť náhľadovú oblasť, aby mohol poskytnúť heuristikám adekvátny kontext. Rozdiel v tomto prípade však nastáva v tom, že heuristikám je predkladaný ako vzor nie jeden, ale zoznam odkazov, ktoré boli vybrané zo všetkých odkazov v náhľadovej oblasti.

Táto voľba súvisí s najzásadnejším problémom tohto algoritmu — v prvých troch až štyroch krokoch sa nedá náhľadová oblasť zaručene a spoľahlivo identifikovať. Heuristiky sa nemôžu priebežne učiť po jednej predkladanom URL, pretože v týchto krokoch majú k dispozícii iba neúplný kontext.

Navrhované riešenie, ako určiť náhľadovú oblasť, je opäť skôr heuristikou, ktorá sa opiera o fakt, že náhľady záznamov v stránkovacej oblasti sa v HTML strome nachádzajú veľmi blízko seba. Systém určuje náhľadovú oblasť metódou "obalovania". Od obdržania druhej demonštrovanej URL zisťuje spoločného rodiča všetkých predošlých URL, až kým sa tento rodič nemení (t.j. "obal" sa ustáli). Potom však z tohto rodiča nevyberá všetky odkazy v ňom, pretože ten už môže byť dosť veľký a môže obsahovať aj odkazy zo stránkovacej oblasti. Preto sa vyberajú iba linky v tých elementoch, ktoré sú priamym potomkom spoločného rodiča, majú rovnakú html značku a nasledujú priamo za sebou.

Spôsob ukončenia procesu učenia je v tomto algoritme trochu pozmenený. V momente, keď už systém má naučenú logiku, ktorá nezmenila svoje parametre istý počet krokov, predpovedá užívateľovi všetky odkazy, o ktorých si myslí, že sa ich užívateľ snaží naučiť. Ak užívateľ tieto odkazy potvrdí v dialógovom okne ponúknutým systémom, systém označí logiku výberu URL detailov za schválenú a ďalej sa ju neučí. Ak užívateľ zamietne, systém pokračuje v učení a čaká na ďalšiu demonštrovanú URL. Ak v systéme nezostala žiadna logika, proces učenia sa končí, systém sa nedokáže naučiť.

6.3.2 Dátové štruktúry algoritmu UE-PBD

6.3.2.1 Heuristiky

Základné triedy, ktoré sa dokážu učiť a extrahovať odkazy. Uvedené triedy obsahujú len rámcový zápis, akú funkcionálnosť musí heuristika ponúkať. Implementované heuristiky tak musia dediť od týchto dvoch a ich zoznam si môžete pozrieť v podkapitole **7.3 Obmedzenia**.

```
class HeuristikaNahladu
{
    bool Naucena;
    bool NechopnaUcitSa;
    PrispособSa(Zoznam<HTMLOdkaz> vybraneOdkazy, Zoznam<HTMLOdkaz> vsetkyOdkazy);
    Zoznam<HTMLOdkaz> Filtruj(Zoznam<HTMLOdkaz> vsetkyOdkazy);
}

class HeuristikaDalsejStranky
{
    bool Naucena;
    bool NechopnaUcitSa;
    void PrispособSa(HTMLOdkaz odkaz, Zoznam<HTMLOdkaz> vybraneOdkazy);
    HTMLOdkaz Filtruj(Zoznam<HTMLOdkaz> vsetkyOdkazy);
}
```

6.3.2.2 URL agent

URL sa skladá z *InstrukciaNahladu* a *InstrukciaDalsejStranky*, ktoré majú rovnakú funkciu ako *HeuristikaNahladu* a *HeuristikaDalsejStranky*.

```
class URLAgent
{
    string vychodiskovaStranka;
    InstrukciaNahladu instrukciaNahladov;
    InstrukciaDalsejStranky instrukciaDalsejStranky;
    HTMLElement nahladovaOblast;
    HTMLElement strankovaciaOblast;

    Extrahuj(string url, out Zoznam<HTMLOdkaz> vystupneUrlDetailov)
    {
        Prehliadac prehliadac = new Prehliadac();
        Zoznam<HTMLOdkaz> odkazyNahladovejOblasti;
        HTMLOdkaz dalsiaStranka;

        prehliadac.naviguj(url);
        do
        {
            odkazyNahladovejOblasti = UEBPD.ZistiOdkazyNahladovejOblasti(
                nahladovaOblast
            );

            vystupneUrlDetailov.Pripoj(
                instrukciaNahladov.Filtruj(odkazyNahladovejOblasti)
            );
            dalsiaStranka = instrukciaDalsejStranky.Filtruj(
                strankovaciaOblast.HTMLOdkazy
            );
            if (dalsiaStranka != null) prehliadac.naviguj(dalsiaStranka.url);
        } while (dalsiaStranka != null);
    }
}
```

6.3.3 Algoritmus UE-PBD

Základná osnova algoritmu:

1. Inicializácia pomocných premenných.
2. Učenie sa/extrakcia URL náhľadovej oblasti.
3. Učenie sa prechodu na ďalšiu stránku.
4. Spustenie agenta na zvyšok vstupu.

```
UEPBD(  
    string vychodiskovaStranka  
    , out Zoznam<HTMLOdkaz> vystupneUrlDetailov  
    , out URLAgent urlAgent  
)  
{  
  
    // 1.inicializacia premennych  
    Prehliadac prehliadac = new Prehliadac();  
    HTMLOdkaz aktivovanyOdkaz;  
    Zoznam<HTMLOdkaz> urlDetailov;  
    Zoznam<HTMLOdkaz> urlDalsichStranok;  
    HTMLElement nahladovaOblast;  
    HTMLElement strankovaciaOblast;  
    Zoznam<HTMLOdkaz> predpovedaneNahlady;  
    HTMLOdkaz predpovedanaDalsiaStranka;  
    urlAgent = new URLAgent();  
    prehliadac.nacitaj(vychodiskovaStranka);  
    prehliadac.zakazNavigaciuPoKlikuNaOdkaz = true;  
    urlAgent.vychodiskovaStranka = vychodiskovaStranka;  
  
    do  
    {  
        // 2.uc heuristiku nahladov  
        if (urlAgent.instrukciaNahladov.Naucena)  
        {  
            aktivovanyOdkaz = CakajNaKlikUzivatel();  
            urlDetailov.pridaj(aktivovanyOdkaz);  
            vystupneUrlDetailov.pridaj(aktivovanyOdkaz);  
            urlAgent.nahladovaOblast = ZistiNahladovuOblast(urlDetailov);  
            nahladovaOblast = urlAgent.nahladovaOblast. ZistiOdkazyNahladovejOblasti(  
                urlDetailov[0]  
            );  
            if (Ustalena(nahladovaOblast))  
            {  
                predpovedaneNahlady = urlAgent.instrukciaNahladov.PrisposobSa(  
                    urlDetailov, nahladovaOblast  
                );  
                if (predpovedaneNahlady.Pocet > 0  
                    && UzivatelSchvalil(predpovedaneNahlady))  
                {  
                    urlAgent.instrukciaNahladov.Naucena = true;  
                }  
                if (urlAgent.instrukciaNahladov.NechopnaUcitSa) return;  
            }  
        }  
    }  
}
```

```

// 2.heuristika nahladov je schvalena - extrahuj nahlady
else
{
    vystupneUrlDetailov.Pripoj(
        urlAgent.instrukciaNahladov.Filtruj(nahladovaOblast)
    );
}

// 3.ucenie heuristiky prechodu na dalsiu stranku
urlDalsichStranok.pridaj(aktivovanyOdkaz);
strankovaciaOblast = ZistiStrankovaciOblast(aktivovanyOdkaz);
urlAgent.strankovaciaOblast = strankovaciaOblast;
urlAgent.instrukciaDalsejStranky.PrisposobSa(aktivovanyOdkaz, strankovaciaOblast);
if (urlAgent.instrukciaDalsejStranky.NechopnaUcitSa) return;
prehliadac.naviguj(aktivovanyOdkaz);
}
while (!urlAgent.instrukciaDalsejStranky.Naucena);

// 4.spusti agenta na zvsok vstupu
urlAgent.Extrahuj(aktivovanyOdkaz.url, out vystupneUrlDetailov);
}

```

6.3.4 Algoritmus Zisti stránkovaciú oblasť

Po kliknutí na `aktivovanyOdkaz`, ktorý sa nachádza v stránkovacej oblasti stránky, systém prechádza HTML strom smerom nahor a hľadá ju.

```

HTMLElement ZistiStrankovaciOblast(HTMLOdkaz aktivovanyOdkaz)
{
    HTMLElement aktualny = aktivovanyOdkaz;
    HTMLElement predosly;
    int minPocet = 3;
    int maxPocet = 15;
    while (aktualny != null && aktualny.HTMLOdkazy.Pocet < minPocet)
    {
        aktualny = aktualny.rodic;
        predosly = aktualny;
    }
    return aktualny != null && aktualny.HTMLOdkazy.Pocet < maxPocet
        ? aktualny
        : predosly
    ;
}

```

6.3.5 Algoritmus Zisti náhľadovú oblasť

Tento algoritmus sa schováva v metóde `HTMLOdkaz.ZistiOdkazyNahladovejOblasti` a je volaný v cykle učenia sa náhľadovej oblasti algoritmu UE-PBD. Zjednodučený cyklus je:

```
HTMLOdkaz prvý = odkazyNahladov[0];
foreach (HTMLOdkaz odkaz in odkazyNahladov)
{
    odkazyNahladovejOblast = odkaz.ZistiOdkazyNahladovejOblasti(prvý);
}

Zoznam<HTMLOdkaz> HTMLOdkaz.ZistiOdkazyNahladovejOblasti(HTMLOdkaz prvýNahlad)
{
    if (this == prvýNahlad)
    {
        return prvýNahlad;
    }
    else
    {
        Zoznam<HTMLOdkaz> odkazy = new Zoznam<HTMLOdkaz>();

        HTMLHtmlElement obal = ZistiObal(this,prvýNahlad);
        string rozlisovaciaZnacka = prvýNahlad.HTMLCesta[obal.HTMLCesta.Pocet].Znacka;
        int poradie = -1;
        foreach (HTMLHtmlElement potomok in obal.deti)
        {
            if (potomok.Znacka == rozlisovaciaZnacka)
            {
                if (    poradie == -1
                    || potomok.Poradie == poradie
                    || potomok.Poradie == poradie + 1)
                {
                    odkazy.Pripoj(potomok.HTMLOdkazy);
                }
                poradie = potomok.Poradie;
            }
        }
        return odkazy;
    }
}
```

7 Implementácia

Nasledujúca kapitola sa venuje detailom týkajúcim sa implementácie systému WDE-PBD.

Na úvod je v popísaný systém z hľadiska zvolenej architektúry v podkapitole **7.1 Popis systému**. Podkapitola **7.2. Členenie systému** približuje kompozíciu systému a podkapitola **7.4 Rozšíriteľnosť** popisuje možnosti ďalšieho rozšírenia systému.

7.1 Popis systému

Systém WDE-PBD je samostatná okienková aplikácia implementovaná v jazyku C# 4.0 určená pre platformu Microsoft .NET vo verzii 4.0. V systéme je používaná jediná nesystémová externá knižnicu MSHTML, ktorá je určená na obohatenie funkcionality komponentu WebBrowser — zabudovaného internetového prehliadača.

Tento zabudovaný prehliadač v aplikácii je takmer identický ako Internet Explorer a zdieľa s ním rovnaké nastavenia. Dôvodom je fakt uvedený v [13]: "Knižnica MSHTML bola uvedená v internetovom prehliadači Microsoft Internet Explorer 4.0. Je hlavným komponentom prehliadača Internet Explorer a môže byť použitá v iných aplikáciách".

7.2 Členenie systému

7.2.1 Jadro

Jadro systému WebDataExtractor.Core je založené na knižnici MSHTML. Funkcionalita, ktorú táto knižnica ponúka, je v ňom rozšírená o novú použitím extensions metód. Tie dovoľujú rozširovať funkcionality už existujúcich knižníc, bez nutnosti zásahu do nich resp použitia dedenia.

7.2.2 Vlastné Komponenty

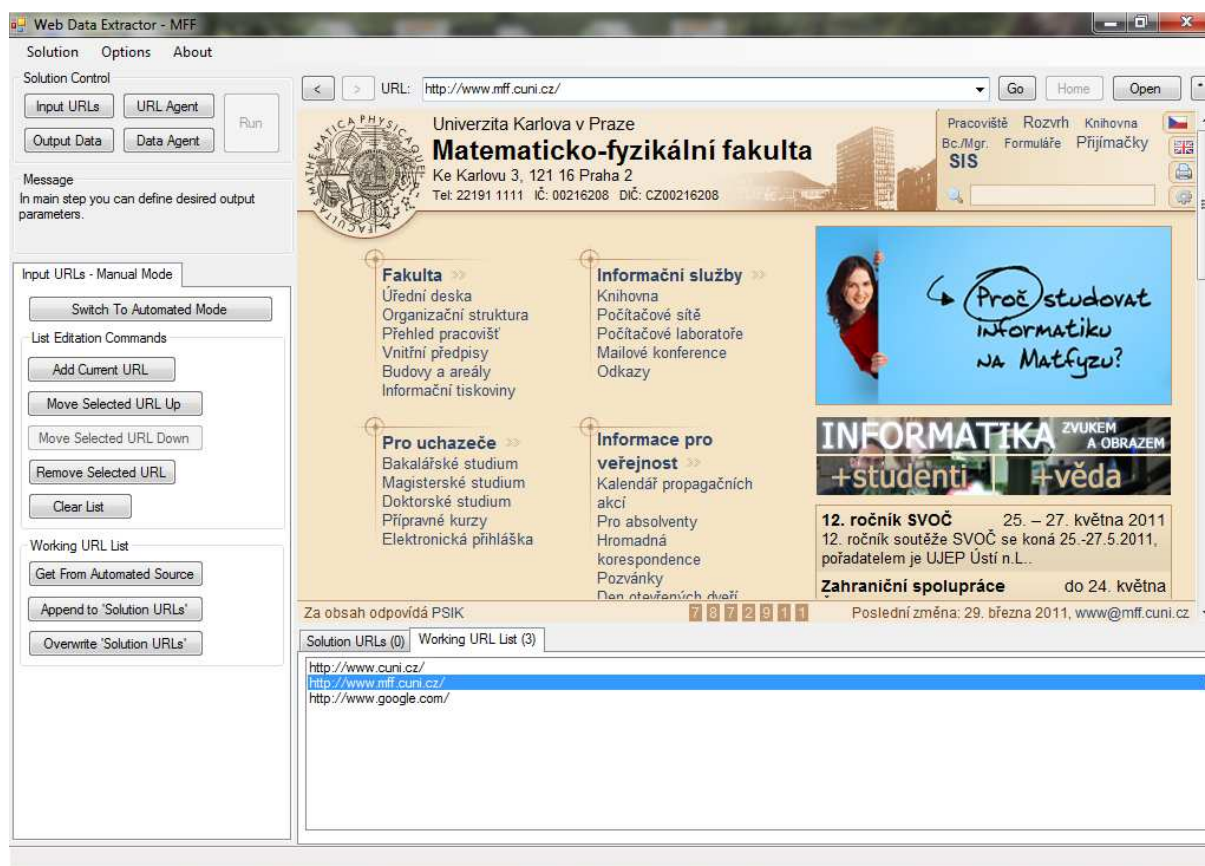
Ide o podpornú assembly WebDataExtractor.Controls pre okienkovú aplikáciu, ktorá obsahuje vlastné rozšírenia štandardných vizuálnych komponentov .NET-u, ktoré sú použité v okienkovej aplikácii. Použité však môžu byť v ľubovoľnej inej aplikácii.

Jedným z vlastných komponentov je HTML vizualizér, ktorý vizualizuje strom HTML dokumentu. Ďalším vlastným komponentom je VariableExplorer, ktorý umožňuje zobraziť a

prehliadať štruktúru zadanej premennej. Oba tieto komponenty sú využívané v pokročilom móde učenia dátového agenta na uľahčenie práce vývojára heuristiky pre systém WDE-PBD.

7.2.3 Okienková aplikácia

Aplikácia pozostáva z hlavného okna a niekoľkých ďalších pomocných okienok. Hlavné okno programu obsahuje menu, panel Solution umožňujúci prístup k jednotlivým nástrojom používaným pri učení resp. extrakcii, panel so systémovými odkazmi pre užívateľa, panel pre prístup k akciám pre jednotlivé nástroje. Ďalej, hlavné okno obsahuje zabudovaný internetový prehliadač v centrálnej časti a pod ním panel zobrazujúci dáta v aktuálnom kroku. Bližší popis jednotlivých častí nájdete v prílohe **11.4 Popis hlavného okna programu**.



Obrázok 18. Screenshot systému WDE-PBD

7.3 Obmedzenia

Implementácia systému WDE-PBD nezohľadňuje všetky body návrhu, odklonila sa od nich v týchto bodoch:

1. Po skončení učenia agenta sa nespustí práve naučený agent na zvyšok vstupu. Užívateľ však môže opustiť obrazovku pre učenie agenta a spustiť naučeného agenta zvlášť na celý vstup.
2. Nie je použitá klávesnica na označovanie textu HTML dokumentu z toho dôvodu, že štruktúra HTML dokumentu je veľmi zložitá a na extrakciu dát sa oveľa lepšie hodí myš.
3. Ťažisko tejto práce spočíva v implementácii heuristik. Sada heuristik implementovaná v systéme WDE-PBD (viď Tabuľka 1) je postačujúca na bezproblémovú extrakciu dobre štruktúrovaných dát.

HTML heuristiky (WebDataExtractor.Core.Agents.Data.Heuristics.HTML)		
Názov	Poradie¹⁴	Popis
FindByIterativePathHeuristic	20	Hľadá cestu odlišnú v jednom elemente ako predchádzajúca, je využívaná pri extrakcii viacerých záznamov z jedného dokumentu.
FindByPathHeuristic	10	Hľadá podľa presnej cesty s odsadením značiek od začiatku rodičovského elementu.
FindByPathReversedOffsetHeuristic	11	Hľadá podľa presnej cesty s odsadením značiek od konca rodičovského elementu.
FindByPathAdaptedOffsetHeuristic	50	Hľadá podľa presnej cesty s odsadením každej značky zvlášť, buď od konca rodičovského elementu alebo od začiatku.
Textové heuristiky (WebDataExtractor.Core.Agents.Data.Heuristics.Text)		
Názov	Poradie	Popis
AdvancedHTMLHeuristic	5000	Hľadá text za zadanou HTML značkou
CharsCropBothHeuristic	1000	Odreže zo začiatku text m znakov a z konca n znakov.
CharsCropLeftHeuristic	2000	Odreže zo začiatku textu m znakov a z výrezu vráti prvých n znakov.
CharsCropRightHeuristic	3000	Odreže z konca textu m znakov a z výrezu vráti posledných n znakov.
WordsCropBothHeuristic	100	Odreže zo začiatku textu m slov a z konca n slov.
WordsCropLeftHeuristic	200	Odreže zo začiatku textu m slov a z výrezu vráti prvých n slov.
WordsCropRightHeuristic	300	Odreže z konca textu m slov a z výrezu vráti posledných n slov.
LinesCropBothHeuristic	10	Odreže zo začiatku textu m riadkov a z konca n riadkov.
LinesCropLeftHeuristic	20	Odreže zo začiatku textu m riadkov a z výrezu vráti prvých n riadkov.
LinesCropRightHeuristic	30	Odreže z konca textu m riadkov a z výrezu vráti posledných n riadkov.

¹⁴ Poradie je definované v časti **6.1.1 Návrh algoritmu**.

URL Heuristiky náhľadov (WebDataExtractor.Core.Agents.URL.Heuristics.Detail)		
Názov	Poradie	Popis
SelectDetailByOffsetHeuristic	10	Vyberie odkazy, ktoré majú nejaký pevný odstup.
SelectDetailByOffsetGroupsHeuristic	100	Vyberie skupiny odkazov, ktoré majú nejaký pevný odstup.
URL Heuristiky ďalších stránok (WebDataExtractor.Core.Agents.URL.Heuristics.Pager)		
Názov	Poradie	Popis
SelectNextPageByIndexHeuristic	10	Kým nie je počet odkazov a index vybraného odkazu ustálený vyberaj podľa zapamätaných indexov, potom podľa ustáleného indexu.
SelectNextNotVisitedPageHeuristic	100	Vyber vždy ešte nenavštívený odkaz zo stránkovacej oblasti.

Tabuľka 1 Prehľad heuristik implementovaných v systéme WDE-PBD

Problém s extrakciou dát z voľne štruktúrovaného textu sa snaží riešiť pokročilá textová heuristika. Tá funguje správne, avšak problémom je, že pri extrakcii dát z voľného textu zlyhajú HTML heuristiky v úlohe poskytnúť jej správny HTML element. Príčinou je voľnosť textu, kedy sa často stane, že pred aj za značkou, z ktorej bol extrahovaný text, pribudnú ďalšie značky podobného charakteru. HTML heuristika by tak musela byť schopná vyhľadávať ten správny element "niekde uprostred", čo sa veľmi ťažko dá popísať nejakou logikou.

Návrhov na odstránenie tohto problému je viacero, to je už ale ponechané na prípadné budúce rozšírenie systému pomocou sprístupneného návodu na implementáciu vlastných heuristik.

Z heuristik neboli implementované:

- HTML heuristiky vyhľadávané "zospodu" podľa vlastností podstromu elementu,
- HTML heuristiky vyhľadávané "priamo" podľa identifikátora,
- textové heuristiky vyhľadávané zadaný reťazec podľa: presnej hodnoty, typu hodnoty (napríklad číslo, mena), alebo regulárneho výrazu,
- textové heuristiky skladané do zložitejších.

7.4 Rozšíriteľnosť

Systém je navrhnutý a implementovaný tak, aby bol pokiaľ možno čo najjednoduchšie rozšíriteľný o nové heuristiky. Programátor, ktorý by ho chcel rozšíriť novou heuristikou musí podediť heuristiku od základnej definovanej heuristiky, implementovať jej funkcie a priradiť jej určité atribúty. Následne túto novú heuristiku je nutné vložiť do assembly jadra projektu a novo zbuildovaný systém ju už potom automaticky načíta a použije v procese učenia. Podrobný návod ako tak môže urobiť sa nachádza v programátorskej dokumentácii (viď **Príloha B. CD nosič.**)

8 Testovacie scenáre

Táto kapitola má za cieľ overiť schopnosť použitia systému implementovaného v tejto práci v reálnych prípadoch pomocou testovacích scenárov. Tie sú primárne dizajnované podľa časti **4.4.4 Formy prezentácie dát** tak, aby overili schopnosť extrakcie dát z čo najväčšieho počtu foriem. Každý z nich obsahuje časť s popisom testovacieho scenára, za ktorou je uvedená tabuľka s výsledkami testu. Za tabuľkou s výsledkami testovacieho scenára nasleduje vždy rozbor výsledkov. Testovacie scenáre používané v tejto kapitole je možné nájsť na priloženom CD nosiči (v adresári viď **Príloha B. CD nosič**). Na záver kapitoly sú v podkapitole **8.5 Zhrnutie výsledkov** uvedené spriemerované výsledky testovacích scenárov spolu so zhodnotením celkových výsledkov programu.

Zoznam sledovaných parametrov počas jednotlivých testovacích scenárov:

- URL agent:
 - o Učenie:
 - počet HTML dokumentov potrebných na naučenie,
 - počet extrahovaných URL potrebných na naučenie.
 - čas učenia v sekundách.
 - o Extrakcia:
 - počet spracovaných HTML dokumentov,
 - počet extrahovaných URL,
 - počet duplicitných URL vo výsledku,
 - čas extrakcie v sekundách.
- Data agent:
 - o Učenie:
 - počet HTML dokumentov potrebných na naučenie,
 - počet extrahovaných záznamov potrebných na naučenie,
 - počet atribútov naučeného data agenta.
 - čas učenia v sekundách.
 - o Extrakcia:
 - počet spracovaných HTML dokumentov,
 - počet extrahovaných záznamov,
 - počet extrahovaných atribútov,
 - počet nesprávne extrahovaných záznamov,
 - počet nesprávne extrahovaných atribútov,
 - čas extrakcie v sekundách.

Pre každý objekt (dokument, záznam, URL, atribút) je vo výsledkoch testu uvedené tiež:

- rýchlosť spracovania — počet objektov spracovaných za sekundu,
- čas v sekundách potrebný na spracovanie jedného objektu.

8.1 Testovací scénár č. 1

Názov: Extrakcia dát o mobilných telefónoch z webu www.mobil.idnes.cz
Zameranie: Extrakcia dát zoskupených v tabuľke
Úloha: Extrahovať atribúty zodpovedajúce skupine "Základní informace"
URL: http://mobil.idnes.cz/katalog.asp?znacka=3&page=1

Výsledky:

Testovací scénár č. 1 – výsledky			dokument	záznam	atribút	celkový čas [s]	celkový čas [%]
URL Agent	Učenie	celkom	4	81	-	70	12,41
		čas na jednotku [s]	17,50	0,86	-	-	-
		rýchlosť [jedn. x s ⁻¹]	0,06	1,16	-	-	-
	Extrakcia	celkom	12	204	-	11	1,95
		čas na jednotku [s]	0,92	0,05	-	-	-
		rýchlosť [jedn. x s ⁻¹]	1,09	18,55	-	-	-
		duplicitných	-	84	-	-	-
		duplicitných [%]	-	41,18	-	-	-
Data Agent	Učenie	celkom	3	3	19	382	67,73
		čas na jednotku [s]	127,33	127,33	20,11	-	-
		rýchlosť [jedn. x s ⁻¹]	0,01	0,01	0,05	-	-
	Extrakcia	celkom	140	140	2660	101	17,91
		čas na jednotku [s]	0,72	0,72	0,04	-	-
		rýchlosť [jedn. x s ⁻¹]	1,39	1,39	26,34	-	-
		nesprávnych	-	12	228	-	-
		nesprávnych [%]	-	8,57	8,57	-	-
Prehľad	Celkom	čas	-	-	-	564	100,00
	Učenie	čas	-	-	-	452	80,14
	Extrakcia	čas	-	-	-	112	19,86

Tabuľka 2. Výsledky testovacieho scénára č.1

Rozbor výsledkov:

Učenie URL agenta prebiehalo rýchlo a bezproblémovo, takisto extrakcia. Agent extrahoval všetky záznamy sprístupnené pomocou stránkovania a žiadne naviac. Duplicita 41,18% bola spôsobená nekvalitou dátového zdroja.

Učenie agenta prebiehalo pomerne rýchlo a bezproblémovo. Jedinou nevýhodou pri extrakcii dát, ktoré obsahujú veľa atribútov je náročnosť kladená na koncentráciu užívateľa. Ak by sa niekde pomýlil, musel by celý proces učenia opakovať znova, pretože systém neponúka možnosť undo.

8.2 Testovací scénár č. 2

Názov: Extrakcia dát o fotoaparátoch z webu www.alza.cz
Zameranie: Extrakcia dát zoskupených textovo
Úloha: Extrahovať atribúty Obrazový snímač, Hmotnosť, Rozmery
URL: http://www.alza.cz/digitalni-foto/18843129.htm

Výsledky:

Testovací scénár č. 2– výsledky			dokument	záznam	atribút	celkový čas [s]	celkový čas [%]
URL Agent	Učenie	celkom	4	81	-	37	9,61
		čas na jednotku [s]	9,25	0,46	-	-	-
		rýchlosť [jedn. x s ⁻¹]	0,11	2,19	-	-	-
	Extrakcia	celkom	30	625	-	58	15,06
		čas na jednotku [s]	1,93	0,09	-	-	-
		rýchlosť [jedn. x s ⁻¹]	0,52	10,78	-	-	-
		duplicitných	-	0	-	-	-
		duplicitných [%]	-	0,00	-	-	-
Data Agent	Učenie	celkom	3	3	4	106	27,53
		čas na jednotku [s]	35,33	35,33	26,50	-	-
		rýchlosť [jedn. x s ⁻¹]	0,03	0,03	0,04	-	-
	Extrakcia	celkom	100	100	400	184	47,79
		čas na jednotku [s]	1,84	1,84	0,46	-	-
		rýchlosť [jedn. x s ⁻¹]	0,54	0,54	2,17	-	-
		nesprávnych	-	70	70	-	-
		nesprávnych [%]	-	70,00	17,50	-	-
Prehľad	Celkom	čas	-	-	-	385	100,00
	Učenie	čas	-	-	-	143	37,14
	Extrakcia	čas	-	-	-	242	62,86

Tabuľka 3. Testovací scénár č.2 – výsledky

Rozbor výsledkov:

Učenie a extrakcia URL prebehla rýchlo a v poriadku. Extrahované boli všetky URL, žiadne navyše, bez duplicitných záznamov.

Učenie data agenta prebiehalo komplikovane. Stránky, na ktorých bola vykonávaná extrakcia neboli validné (chýbala koncová značka div), a preto heuristiky počas učenia nepracovali správne. Po správnom preusporiadaní vstupných URL (v manuálnej editácii zoznamu URL) tak, aby obsahovali podobné (hoci nevalidné) dokumenty, sa podarilo doviesť proces učenia k úspechu.

Data agent extrahoval trochu pomalšie (približne 10-krát oproti testovaciemu scenáru č 1.). Príčinou bola väčšia výpočtová náročnosť pokročilej textovej heuristiky v ňom použitej/naučenej. Agent extrahoval iba 30% všetkých záznamov správne — chyba nastávala iba v atribúte Hmotnosť, ktorý mal overiť použitie pokročilej textovej heuristiky. Príčin bolo hneď niekoľko:

1. Stránky boli nevalidné a systému sa javilo, že pred aj za elementom, v ktorom mala byť vykonávaná extrakcia, pribudla značka rovnakého typu, a teda vyhľadávanie HTML značky podľa cesty zlyhalo (pretože hľadaný element mal byť "niekde uprostred").
2. Na stránkach, ktoré nezodpovedali bodu 1. nebol vôbec prítomný text "Hmotnosť:" resp. na nich bola prítomná nejaká jeho modifikácia (napr "Hmotnosť (g):").

8.3 Testovací scénár č. 3

Názov: extrakcia aukcií automobilov z webu www.ebay.com
Zameranie: Extrakcia štruktúrovaných dát zoskupených v tabuľke
Úloha: Extrahovať atribúty Názov modelu, aktuálna ponuka
URL: http://motors.shop.ebay.com/Cars-Trucks-
 /6001/i.html?LH_ItemCondition=1&_dmpt=US_Cars_Trucks&_fcid=9&_jgr=0&_loc
 alstpos=&_sop=7&_sticky=1&_stpos=&gbr=1

Výsledky:

Testovací scénár č. 3 – výsledky			dokument	záznam	atribút	celkový čas [s]	celkový čas [%]
URL Agent	Učenie	celkom	3	25	-	74	10,93
		čas na jednotku [s]	24,67	2,96	-	-	-
		rýchlosť [jedn. x s ⁻¹]	0,04	0,34	-	-	-
	Extrakcia	celkom	26	1291	-	85	12,56
		čas na jednotku [s]	3,27	0,07	-	-	-
		rýchlosť [jedn. x s ⁻¹]	0,31	15,19	-	-	-
		duplicitných	-	78	-	-	-
		duplicitných [%]	-	6,04	-	-	-
	Data Agent	celkom	3	3	2	70	10,34
		čas na jednotku [s]	23,33	23,33	35,00	-	-
		rýchlosť [jedn. x s ⁻¹]	0,04	0,04	0,03	-	-
		celkom	40	40	80	448	66,17
		čas na jednotku [s]	11,20	11,20	5,60	-	-
		rýchlosť [jedn. x s ⁻¹]	0,09	0,09	0,18	-	-
		nesprávnych	-	28	28	-	-
		nesprávnych [%]	-	70,00	35,00	-	-
Prehľad	Celkom	čas	-	-	-	677	100,00
	Učenie	čas	-	-	-	144	21,27
	Extrakcia	čas	-	-	-	533	78,73

Tabuľka 4. Testovací scénár č.3 – výsledky

Rozbor výsledkov:

URL agent sa naučil štandardne rýchlo. Pri extrakcii extrahoval 78 duplicitných záznamov, čo predstavuje 6,04% z celkového počtu. Agent extrahoval nadmnožinu všetkých záznamov sprístupnených pomocou stránkovania, čo znamená, že extrahoval nesprávne URL. Duplicita bola spôsobená z väčšej časti nekvalitou zdroja (obsahoval duplicitné záznamy) a z menšej časti zlým výberom URL náhľadov URL agentom, ktorý extrahoval z každej stránky jednu pevnú URL naviac. Tento omyl je však ospravedlniteľný, pretože po odfiltrovaní duplikátov predstavuje chybu rádove v promile. Pri extrakcii sa potom z neho neextrahujú dáta a

vo výsledku sa znovu dá odfiltrovať prázdny výstupný záznam. Chyba URL agenta je tak vo výstupe úplne eliminovaná.

Data agent extrahoval relatívne pomaly (v porovnaní s výsledkami obdobného testovacieho scenáru č. 1 rádovo až 100-krát viac). Dôvodom bola prítomnosť veľkého počtu obrázkov konkrétneho automobilu na stránke, ktoré predĺžili dobu na stiahnutie jedného dokumentu aj na 10 sekúnd. Data agent extrahoval iba 30% záznamov korektne. Dôvodom je, že vstupné HTML dokumenty nemali rovnakú štruktúru, a len tretina z nich obsahovala položku aktuálna ponuka (ostatné obsahovali inú HTML štruktúru s uvedenou cenou a pod.).

8.4 Testovací scenár č. 4

Názov: Extrakcia kurzového lístka z webu www.csob.cz
Zameranie: Extrakcia viacerých detailov záznamov v jednom HTML dokumente
Úloha: Extrahovať atribúty kurzu : mena, nákup, predaj, stred pre celý týždeň
URL: <http://www.csob.cz/cz/Csob/Kurzovni-listky/Stranky/kurzovni-listek.aspx>

Výsledky:

Testovací scenár č. 4 – výsledky			dokument	záznam	atribút	celkový čas [s]	celkový čas [%]
Data Agent	Učenie	celkom	1	4	4	81	96,43
		čas na jednotku [s]	81,00	20,25	20,25	-	-
		rýchlosť [jedn. x s ⁻¹]	0,01	0,05	0,05	-	-
	Extrakcia	celkom	5	80	320	3	3,57
		čas na jednotku [s]	0,60	0,04	0,01	-	-
		rýchlosť [jedn. x s ⁻¹]	1,67	26,67	106,67	-	-
		nesprávnych	-	0	0	-	-
		nesprávnych [%]	-	0,00	0,00	-	-
Prehľad	Celkom	čas	-	-	-	84	100,00
	Učenie	čas	-	-	-	81	96,43
	Extrakcia	čas	-	-	-	3	3,57

Tabuľka 5. Testovací scenár č.4 – výsledky

Rozbor výsledkov:

URL agenta v tomto prípade nebolo nutné učiť ani spúšťať.

Učenie data agenta i extrakcia prebehli hladko. Čas extrakcie jedného záznamu bol rádovo 10 až 100-krát menší ako v ostatných prípadoch. Dôvodom je charakter dát, kde je v jednom HTML dokumente mnoho záznamov, ktoré sa z hľadiska heuristik extrahujú ľahko, pretože sú väčšinou umiestnené v tabuľke.

8.5 Zhrnutie výsledkov

Nasledujúca tabuľka obsahuje spriemerované výsledky jednotlivých testovacích scenárov.

Priemerné výsledky			dokument	záznam	atribút	celkový čas [s]	celkový čas [%]
URL Agent	Učenie	celkom	3,67	62,33	N/A	60,33	13,25
		čas na jednotku [s]	16,45	0,97	N/A	N/A	N/A
		rýchlosť [s ⁻¹]	0,06	1,03	N/A	N/A	N/A
	Extrakcia	celkom	22,67	706,67	N/A	51,33	11,27
		čas na jednotku [s]	2,26	0,07	N/A	N/A	N/A
		rýchlosť [s ⁻¹]	0,44	13,77	N/A	N/A	N/A
		duplicitných	N/A	54,00	N/A	N/A	N/A
		duplicitných [%]	N/A	7,64	N/A	N/A	N/A
Data Agent	Učenie	celkom	2,5	3,25	7,25	159,75	35,08
		čas na jednotku [s]	63,90	49,15	22,03	N/A	N/A
		rýchlosť [s ⁻¹]	0,02	0,02	0,05	N/A	N/A
	Extrakcia	celkom	71,25	90	865	184	40,40
		čas na jednotku [s]	2,58	2,04	0,21	N/A	N/A
		rýchlosť [s ⁻¹]	0,39	0,49	4,70	N/A	N/A
		nesprávnych	N/A	27,5	81,5	N/A	N/A
		nesprávnych [%]	N/A	30,56	9,42	N/A	N/A
Prehľad	Celkom	čas	N/A	N/A	N/A	455,417	100,00
	Učenie	čas	N/A	N/A	N/A	220,083	48,33
	Extrakcia	čas	N/A	N/A	N/A	235,333	51,67

Tabuľka 6. Zhrnutie výsledkov

Učenie a extrakcia v prípade URL agenta boli bezproblémové a rýchle. Učenie URL agenta trvá približne rovnako dlho ako extrakcia URL. Na naučenie agenta je však nutné mať istý cvik.

Dáta o učení data agenta v tejto tabuľke nie sú veľmi výpovedné z toho dôvodu, že učenie bolo nutné v niektorých prípadoch opakovať znova a celkový čas, ktorý užívateľ strávil učením nie je zaznamenaný.

Celkový čas extrakcie dát bol v scenároch č. 2 a č. 3 veľký. V prvom prípade dôvodom bolo, že data agent nespracováva HTML stránku hneď po obdržaní kódu, ale až po obdržaní všetkých objektov (čo v niektorých prípadoch môže predĺžiť čas extrakcie až 10 až 100 násobne). V druhom prípade bol dôvodom fakt, že zložitejšie heuristiky nie sú optimalizované na rýchlosť (v porovnaní s predošlou príčinou sú však rádovo 10-krát rýchlejšie). Preto bola extrakcia dát v týchto prípadoch obmedzená na počet vstupných dokumentov tak, aby bol dosiahnutý rozumný čas celkovej extrakcie.

Celkový pomer času učenia a extrakcie je tak v konečnom dôsledku iný. Na učení agenta strávi užívateľ menej času ako naučený data agent extrakciou. Ak porovnáme čas, ktorý potrebuje užívateľ na extrakciu jedného atribútu (6,77s) a data agent (0,21s) dostávame vcelku prijateľný pomer ušetrenia času automatizáciou — rádovo v desiatkach. Tento pomer sa samozrejme dá vylepšiť vyriešením problémov spomínaných v predchádzajúcom odstavci.

Data agentovi sa podarilo dosiahnuť chybovosť 9,42%, čo nie je zlý výsledok, avšak tieto chyby spôsobili až 30,56% chybovosť záznamov.

Príčin tejto chybovosti je viacero:

- chybné stránky, ktoré sa z výstupu URL agenta môžu objaviť (napríklad stránka so starou štruktúrou, alebo neexistencia HTML dokumentu),
- stránky obsahujú dáta vo forme, ktorá je ťažko extrahovateľná (napríklad hľadaný HTML element je vždy "niekde v strede"),
- každá stránka obsahuje mierne modifikovanú HTML štruktúru,
- sada heuristík nie je postačujúca na extrakciu dát z voľne štruktúrovaného textu.

Celkovo sa dá povedať, že systém WDE-PBD dosahuje dobré (nie výborné) výsledky. URL agent je veľmi užitočným a nápomocným nástrojom pri celkovom ušetrení času použitím automatizácie. Je však škoda, že systém si nedokázal poradiť s niektorými nástrahami v reálnych prípadoch extrakcie dát. Na dosiahnutie lepších výsledkov však existuje priestor — odstránenie čakania na načítanie celého dokumentu, kvalitnejšia implementácia heuristík, ktoré by si dokázali poradiť aj s nevalidnými dokumentmi, implementácia nových a chytrejších heuristík.

9 Záver

Diplomová práca splnila hlavný cieľ navrhnuť a implementovať systém na získavanie dát z dynamicky generovaných webových stránok technikou programovania demonštráciou.

Užívateľ má možnosť nielen demonštrovať zber dát, ale aj zber URL odkazujúcich na HTML extrahované dokumenty, čím sa výrazne automatizuje celý proces extrakcie dát z internetu. Na získanie aktuálnych extrahovaných dát systému WDE-PBD stačí jeden súbor, v ktorom sú obsiahnuté len tri nutné základné vstupy a to: východisková stránka pre extrakciu, naučený URL agent a naučený dátový agent.

V úvode diplomovej práce boli položené nasledujúce otázky:

1. Dá sa implementovať uvedený systém za použitia techniky programovania demonštráciou tak, aby bol použiteľný v praxi?
2. Ak áno, aké sú jeho výhody a nevýhody oproti ostatným nástrojom venujúcim sa extrakcii dát z Internetu?
3. Ak nie, aké sú hlavne obmedzenia vyplývajúce z použitia tejto techniky?

1. Výsledky testov systému WDE-PBD v kapitole **8 Testovacie scenáre** ukazujú, že sa dá implementovať uvedený systém za použitia techniky programovania demonštráciou. Pre použitie v praxi by bolo potrebné ho však rozšíriť o niekoľko ďalších vhodných heuristik a zlepšiť komunikáciu medzi systémom a užívateľom.

2. Najväčšou výhodou systému WDE-PBD oproti ostatným nástrojom riešiacim problematiku extrakcie dát z webu je, že jeho užívatelia nemusia mať žiadnu znalosť programovacieho jazyka na to, aby pomocou neho dokázali extrahovať dáta. Ďalšou z výhod je možnosť pomerne rýchleho naučenia agenta užívateľom respektíve preučenia agenta v prípade zmeny v štruktúre stránky, na ktorú je naučený.

S úlohou extrahovať dáta z Internetu som sa osobne mal možnosť neraz stretnúť ako počas štúdia na vysokej škole, tak v praxi ako programátor. Na univerzite počas práce na semestrálnej úlohe z predmetu Dobývanie znalostí sme s kolegom použili jazyk Perl. Vývoj a odladenie skriptu na extrakciu trval jeden deň a samotné získanie stránok na extrakciu zabralo taktiež nemalé úsilie. V praxi som modifikoval už existujúci skript, ktorý sa stal nefunkčným po úpravách extrahovaného webu a jeho odladenie a sfunkčnenie trvalo tiež približne jeden deň. V oboch prípadoch zabralo vyriešiť túto úlohu neporovnateľne väčšie úsilie ako použitím systému vyvinutého v tejto práci.

3. Použitá technika PBD v tejto práci si vynútila existenciu zložitejšieho užívateľského rozhrania, na ktoré si užívateľ nemusí zvyknúť. Všeobecne nevýhodou PBD je, že sa systém nie vždy dokáže naučiť užívateľov zámer pri učení agenta.

Systém WDE-PBD splňuje všetky základné požiadavky:

- spracováva validné stránky Strict verzie HTML neobsahujúce JavaScript ani AJAX,

- implementuje jednoduchý spôsob učenia extrakcie dát zo stránok majúcich rovnakú HTML štruktúru (rozvrhnutie) a
- umožňuje vytvoriť program na extrakciu dát za rozumnej užívateľovej pomoci.

Z pokročilejších požiadaviek systém splňuje požiadavku učenie/predikcia už po prvom demonštrovanom zázname, vďaka použitiu algoritmu, ktorý sa opiera o silnú znalosť domény na základe analýzy v časti **4.4 Analýza problematiky WDE**.

Ideálne požiadavky na systém sa nepodarilo naplniť. Knižnica MTHML, používaná v systéme, funguje nekorektne na nevalidných HTML stránkach. Požiadavku inteligentný spôsob naučenia sa — extrahovať dáta zo stránok majúcich značne odlišnú HTML štruktúru (rozvrhnutie) sa nepodarilo zohľadniť. Spracovanie dynamických stránok, ktoré obsahujú extrahované dáta pomocou JavaScriptu či AJAXu, nie je možné, pretože systém pri extrakcii nesimuluje kurzor myši, ale strojovo vyberá cieľové dáta z HTML dokumentu.

Systém WDE-PBD je v súčasnosti možné celkom jednoduchým spôsobom rozširovať o nové heuristiky postupom uvedeným v programovej dokumentácii, ktorá sa nachádza na priloženom CD.

Medzi možné budúce vylepšenia patrí URL agent druhej úrovne, ktorý by dokázal extrahovať odkazy nielen pre jednu skupinu entity asociovanú s nejakou východiskovou stránkou, ale všetky URL stránok webu pre všetky typy entít. Najprv by boli získané prvým URL agentom všetky URL východiskových stránok pre každý typ entity, a potom by sa na každú východiskovú stránku aplikoval URL agent druhej úrovne.

V systéme mohla byť použitá iná technika programovania demonštráciou. Systém by sa pre užívateľa javil ako nástroj, kde sa ručne vykonáva extrakcia dát, ale na pozadí by obsahoval nahrávač užívateľských akcií, z ktorých by sa snažil odvodzovať procedúru extrakcie. Užívateľ by počas extrakcie navigoval z východiskovej stránky až na detail prvého záznamu, kde by vykonal extrakciu. Následne by sa vrátením v histórii prehliadania dostal na východiskovú stránku, odkiaľ by navigoval na druhý záznam atď. Po skončení extrakcie na prvej navigačnej stránke by prešiel na druhú a celý tento postup by opakoval, kým by sa systém (ne)naučil.

To už je ale námetom na ďalšie pokračovanie predloženej práce. Aplikácia v pokročilom režime umožňuje detailnejšie sledovať fungovanie používaných heuristik, čo by mohlo zohrať významnú rolu pri budúcom rozširovaní systému.

10 Zoznam použitej literatúry

- [1] **Wikipedia**. Data Scraping. [Online] [Dátum: 3. December 2010.] http://en.wikipedia.org/wiki/Data_scraping.
- [2] **Wikipedia**. Web Scraping. [Online] [Dátum: 3. December 2010.] http://en.wikipedia.org/wiki/Web_scraping.
- [3] **Admarket**. Projekt Heureka.cz. [Online] [Dátum: 3. December 2010.] <http://www.admarket.cz/projekty/heureka-cz/>.
- [4] **QualityUnit**. Unit Miner Software. [Online] [Dátum: 3. December 2010.] <http://www.qualityunit.com/unitminer/unit-miner-software/>.
- [5] **Lixto**. Visual Web Information Extraction with Lixto. [Online] [Dátum: 3. December 2010.] <http://www.dbai.tuwien.ac.at/staff/baumgart/lixto/vldb.pdf>.
- [6] **Lixto**. Declarative Information Extraction, Web. [Online] [Dátum: 3. December 2010.] <http://www.dbai.tuwien.ac.at/staff/baumgart/lixto/lpnmr.pdf>.
- [7] **Lixto**. Lixto Software | Tools & Middleware. [Online] [Dátum: 3. December 2010.] http://www.lixto.com/?page_id=50.
- [8] **Mozenda**. Videos - Capturing text. [Online] [Dátum: 3. December 2010.] <http://www.mozenda.com/video02-capture-text>.
- [9] **Mozenda**. Pricing Plans. [Online] [Dátum: 3. December 2010.] <http://www.mozenda.com/pricing>.
- [10] **A. Cypher, D. C. Halbert, D. Kurlander, H. Lieberman, D. Maulsby, B. A. Myers, A. Turransky (Eds.)**. *Watch what I do: programming by demonstration*. Cambridge, MA, USA : MIT Press, 1993.
- [11] **Lieberman, H., Paternò, F. a Wulf, V.** *End User Development*. 2006.
- [12] **Wikipedia**. End-user development. [Online] [Dátum: 7. December 2010.] http://en.wikipedia.org/wiki/End-user_development.
- [13] **Wikiedia**. Programming by Demonstration. [Online] [Dátum: 3. December 2010.] http://en.wikipedia.org/wiki/Programming_by_demonstration.
- [14] **MSDN**. WebBrowser. [Online] [Dátum: 3. December 2010.] <http://msdn.microsoft.com/en-us/library/system.windows.forms.webbrowser.aspx>.
- [15] **Code Project**. The most complete C# Webbrowser wrapper control. [Online] [Dátum: 3. December 2010.] <http://www.codeproject.com/KB/miscctrl/csEXWB.aspx>.

[16] **TeamDev.** Purchase JXBrowser. [Online] [Dátum: 3. December 2010.]
<http://www.teamdev.com/store/jxbrowser/>.

[17] **Mozilla.** How to develop a Firefox extension. [Online] [Dátum: 3. December 2010.]
<http://blog.mozilla.com/addons/2009/01/28/how-to-develop-a-firefox-extension/>.

[18] **MSDN.** MSHTML Overview. [Online] [Dátum: 3. December 2010.]
<http://msdn.microsoft.com/en-us/library/bb498651%28VS.85%29.aspx>.

11 Úvod

11.1 Účel programu Web Data Extractor

Program Web Data Extraktor slúži na automatizovanú extrakciu dát z webu. Užívateľ má možnosť extrahovať dáta z HTML dokumentov jednak on-line, priamo z webu, tak aj offline z lokálneho disku svojho počítača.

Vytvorenie vlastného riešenia na extrakciu dát je jednoduché, užívateľ nemusí ovládať žiadny programovací jazyk, stačí ak ukáže programu ako na to, a o zvyšok sa postará program sám. Na naučenie programu sa používa technika programovania demonštráciou. Jej výhodou je rýchlosť učenia, takto výsledné riešenie je vysoko automatizovateľné. Web Data Extraktor umožňuje užívateľovi riešenie extrakcie vytvoriť, prispôbovať, uložiť ho a následne ľubovoľne spúšťať.

Program umožňuje ukladať extrahované dáta do jednoduchého textového formátu CSV, ktorý je vhodné spracovať napríklad v tabuľkovom editore. Pre pokročilejších užívateľov je v programe možnosť uložiť výsledky vo forme SQL skriptu, ktorým je možno naimportovať resp. zaktualizovať obsah databázy.

11.2 Základné pojmy

Solution — predstavuje Vaše riešenie na extrakciu dát, ktoré vytvárate pomocou tohto programu. Ukladá sa doň všetko, čo je potrebné na vykonanie extrakcie dát.

Agent — agent pre Vás vykonáva automatizovanú činnosť a je súčasťou Solution. Vaše Solution môže obsahovať 2 typy agentov:

- **URL agent** — agent, ktorý extrahuje URL dokumentov, na ktorých chcete vykonať extrakciu,
- **Data agent** — agent, ktorý za Vás extrahuje dáta z HTML dokumentov.

Každého agenta musíte najprv vždy naučiť ako má extrahovať URL dokumentov resp. dáta z nich. Ďalej ho potom môžete ľubovoľne spúšťať. Ak Vám z nejakých dôvodov nebude vyhovovať, môžete naučiť nového.

11.3 Základné workflow programu

Proces extrakcie v programe Web Data Extractor pozostáva z troch základných krokov:

1. Získanie zoznamu URL dokumentov, na ktorých bude extrakcia vykonávaná.
2. Extrakcia dát zo získaného zoznamu URL.
3. Uloženie extrahovaných dát do textového súboru.

1. Na vytvorenie kompletného riešenia, ktoré môže byť opätovne spúšťané, je nutné aby užívateľ špecifikoval zdroj URL, na ktorých sa bude vykonávať extrakcia. Tento zdroj môže byť:

- URL agent (je nutné ho naučiť ako extrahovať vstupné URL),
- stránky uložené v adresári na lokálnom PC,
- textový súbor so zadanými URL,
- zoznam URL uložený v solution.

2. Extrakcia dát sa vykonáva pomocou naučeného data agenta. Pred spustením data agenta na dokumenty je nutné ho najprv naučiť. Učenie je možné v prípade, že vo svojom riešení máte nejaký zoznam URL, ktorý obsahuje viac ako 3 stránky.

3. Po skončení extrakcie dát data agentom z dokumentov načítaným zo vstupného URL zoznamu môžete uložiť svoje dáta do dvoch rôznych formátov:

- CSV — dáta uložené v textovom súbore v podobe tabuľky,
- SQL — umožňuje vytvoriť databázové skripty typu INSERT a UPDATE.

11.4 Popis hlavného okna programu

Aplikácia **Web Data Extractor** obsahuje jedno hlavné okno (viď Obrázok 19).



Obrázok 19. Aplikácia Web Data Extraktor

Priblížme si jednotlivé časti hlavného okna zobrazeného na obrázku:

1. **Menu** — umožňuje Vám kontrolovať aplikáciu. Obsahuje 3 položky:
 - o **Solution** — dovoľuje Vám ovládať solution a celý program:
 - **New** — slúži na vytvorenie nového solution.
 - **Open** — slúži na otvorenie už existujúceho solution.
 - **Save as...** — slúži na uloženie solution do konkrétneho umiestnenia na Vašom PC.
 - **Close** — zatvára práve otvorené solution.
 - **Exit** — zatvára program.
 - o **Options** — umožňuje Vám nastaviť rôzne nastavenia programu.
 - o **About** — po kliknutí sem sa Vám zobrazia informácie o programe Web Data Extractor.
2. **Panel Solution** — obsahuje tlačidlá pre prístup k základným nástrojom na prácu so Solution:

- o **Input URLs** — umožňuje získavať URL a meniť nastavenia získavania URL.
 - o **URL Agent** — poskytuje prístup k URL agentovi.
 - o **Output Data** — slúži na uloženie výstupu extrakcie, nastavovanie formátu a vlastností výstupu.
 - o **Data Agent** — poskytuje prístup k data agentovi.
 - o **Run** — týmto tlačidlom môžete spustiť celé solution na extrakciu dát v prípade, že máte špecifikovaný zdroj URL a naučeného Data agenta.
3. **Panel Messages** — v tomto paneli s Vami systém komunikuje, zobrazuje Vám hlásenia a preto by ste mu pri práci s programom mali venovať zvýšenú pozornosť.
4. **Panel Akcií** — obsahuje nástroje a tlačidlá pre prácu s časťami Solution.
5. **Ovládacie prvky prehliadača:**
- o **<** — naviguje prehliadač na predchádzajúcu stránku v histórii.
 - o **>** — naviguje prehliadač na nasledujúcu stránku v histórii.
 - o **Pole URL** — slúži na zadávanie URL adresy.
 - o **Go** — naviguje na adresu.
 - o **Home** — naviguje na domovskú stránku.
 - o **Open** — otvára stránku uloženú na disku.
6. **Okno prehliadača.**
7. **Informačné záložky** — v tejto časti okna sa Vám zobrazujú rôzne dáta, či už sú to zozbierané URL, dáta alebo log aplikácie. Záložky sa zobrazujú vždy podľa toho, na akej časti Solution práve pracujete.

12 Popis funkcií

12.1 Manipulácia so Solution

Na manipuláciu s Vaším solution slúži položka **Solution** v **menu** lište programu.

Ak chcete:

- **Vytvoriť nové solution** — kliknite na položku menu **Solution** → **New**.
- **Otvoriť existujúce solution** — kliknite na položku menu **Solution** → **Open**.
- **Uložiť aktuálne solution niekam na disk** — kliknite na položku menu **Solution** → **Save as...**
- **Zatvoriť aktuálne solution** — kliknite na položku menu **Solution** → **Close**.

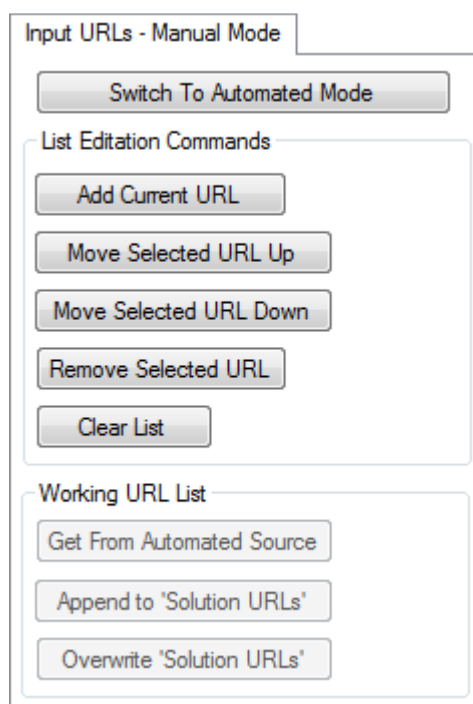
12.2 Získanie URL adries ručne

Pre ručné získavanie URL adries slúži panel akcií s názvom **Input URLs - Manual Mode** (viď Obrázok 20), ktorý si sprístupníte ak v paneli **Solution** kliknete na tlačidlo **Input URLs**.

V tomto paneli máte možnosť pracovať s dvoma zoznamami URL, ktorým odpovedajú záložky v informačnom paneli:

- **Solution URLs** — zoznam URL uložený vo Vašom solution, ktorý môže byť jedným zo zdrojov pri automatizovanom zbere dát.
- **Working URLs** — pracovný zoznam URL, ktorý je využívaný pri výučbových častiach programu.

Tento panel akcií Vám umožňuje vykonávať správu oboch zoznamov URL, podľa toho ktorú informačnú záložku máte zobrazenú.



Obrázok 20. Panel akcií Input URLs - Manual Mode

12.2.1 Pridanie URL do zoznamu

1. Naviguje na stránku, ktorú chcete pridať pomocou **poľa adresy** a tlačidla **Go** resp. kliknutím na tlačidlo **Home** alebo **Open**.
2. V paneli **List Editation Commands** kliknete na tlačidlo **Add Current URL**.

12.2.2 Odobratie vybranej URL zo zoznamu

1. V **informačnom paneli** v záložke URL zoznamu kliknite na URL, ktorú chcete z neho odobrať.
2. V paneli **List Editation Commands** kliknite na tlačidlo **Remove Selected URL**.

12.2.3 Zmazanie celého zoznamu URL

1. V paneli **List Editation Commands** kliknite na tlačidlo **Clear**.

12.2.4 Preusporiadanie poradia v URL zozname

1. V **informačnom paneli** v záložke **Solution's URLs** kliknite na URL, ktorú chcete posunúť.
2. V paneli **List Editation Commands** kliknite na tlačidlo:
 - **Move Up** — pre pohyb hore,
 - **Move Down** — pre pohyb dole.

12.2.5 Získanie zoznamu URL zo automatizovaného zdroja

V prípade, že ste zadali pri automatizovanej extrakcii iný zdroj URL ako zo Solution máte možnosť získať URL z tohto zdroja do zoznamu pracovných URL.

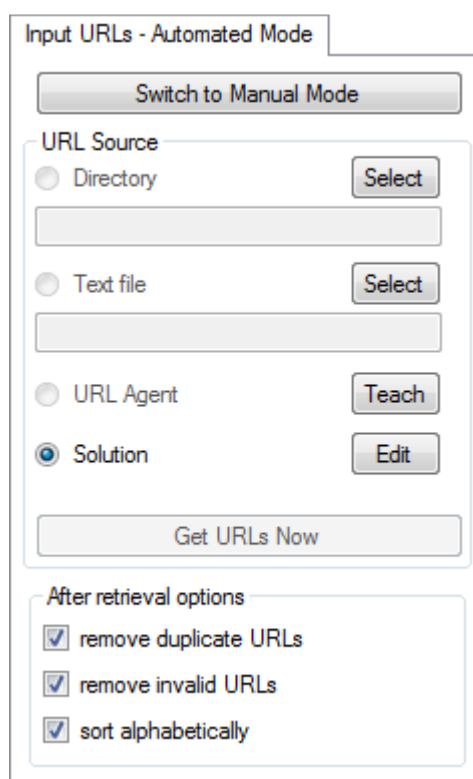
1. V paneli **Working URL List** kliknite na tlačidlo **Get From Automated Source**.

12.2.6 Uloženie zoznamu pracovných URL do solution

1. V paneli **Working URL List** kliknite na tlačidlo:
 - o **Append to Solution URLs**, ak chcete pripojiť zoznam pracovných URL na koniec zoznamu URL uložených v solution,
 - o **Overwrite Solution URLs**, ak chcete prepísať existujúce URL v solution záznamami z pracovného zoznamu URL.

12.3 Automatizované získavanie URL adries

Pre automatizovanú extrakciu dát je nutné špecifikovať jej zdroj URL v paneli akcií **Input URLs - Automated Mode** (viď Obrázok 21). Na jeho zobrazenie kliknite v paneli Solution na tlačidlo **Input URLs** a potom na tlačidlo **Switch to Automated Mode**.



Obrázok 21. Panel akcií Input URLs - Automated Mode

12.3.1 Špecifikovanie zdroja URL

1. V paneli **URL Source** vyberte zdroj:
 - o **Directory** — slúži na načítanie URL dokumentov s príponou *.htm a *.html zo špecifikovanej zložky na disku počítača:
 1. Zložku špecifikujete kliknutím na tlačidlo **Select**.
Po vybraní zložky sa Vám zobrazí cesta k zložke v príľahlom poli.
 - o **Text file** — slúži na načítanie URL z textového súboru:
 1. Súbor vyberiete kliknutím na tlačidlo **Select**.
Po vybraní súboru sa Vám zobrazí cesta k súboru
 - o **URL agent** — slúži na načítanie URL adries prostredníctvom URL agenta:
 - Nového agenta naučíte po kliknutí na tlačidlo **Teach** (viď ďalejv **12.4.1 Učenie URL Agentu**).

- o **Solution** — program nenačíta URL adresy, použije zoznam uložený v Solution.

12.3.2 Získanie zoznamu URL adries zo zadaného zdroja

1. V paneli **URL Source** kliknite na tlačidlo **Get URLs Now**.

URL adresy sa začnú získavať (ich zoznam sa zobrazuje v **Informačnom** paneli v záložke **Working URL List**).

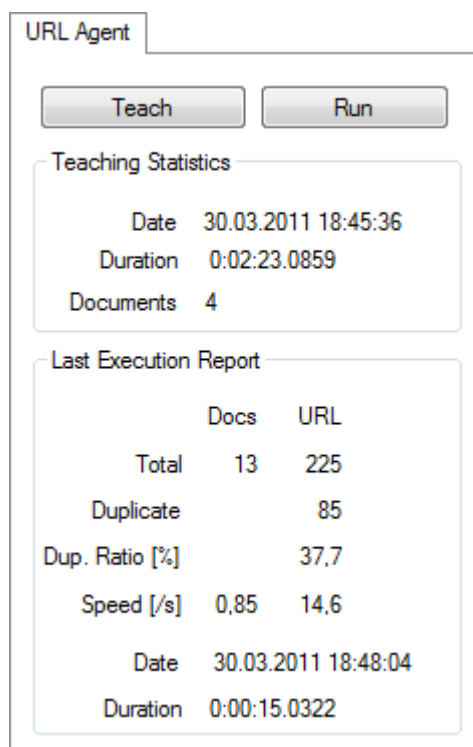
12.3.3 Špecifikovanie nastavení po získaní URL

Pre automatizovaný mód je možné zadať, čo má program po získaní URL vykonať. V paneli **After Retrieval Options** zaškrtnite podľa uváženia:

- **remove duplicate URLs** — vyradí duplicitné URL zo zoznamu získaných URL,
- **remove invalid URLs** — vyradí URL záznamy, ktoré majú nesprávny tvar, alebo neexistujú na disku lokálneho počítača,
- **sort alphabetically** — zotriedi URL vzostupne podľa abecedy.

12.4 URL agent

Panel akcií **URL Agent** je prístupný po kliknutí v paneli **Solution** na tlačidlo **URL Agent** (viď Obrázok 22.).



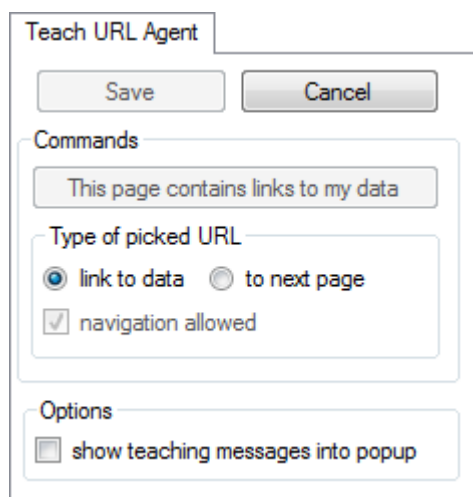
Obrázok 22. Panel akcií URL Agent

URL agent je nástroj, ktorý Vám umožní rýchlo a jednoducho získavať URL dokumentov z webov, ktoré ponúkajú vyhľadávanie položiek, ktoré chcete extrahovať. URL agent extrahuje URL odkazy na tieto položky zo stránok, ktoré Vám web ponúka na ich vyhľadanie. Príklad takejto stránky vidíte na Obrázku 23. Zoznam týchto položiek je vypisovaný do oblasti B, kde súčasťou každej položky je aj URL. Užívateľ môže pristupovať k zvyšným stránkam pomocou odkazov v časti A1 alebo A2.

URL agent sa dokáže naučiť extrahovať linky z uvedených typov stránok. Pri učení sa sústreďí na oblasti A1, A2 a B, v ktorých mu postupne musíte demonštrovať odkazy. Pri demonštrovaní sa, prosím, zamerajte vždy len na jednu z oblastí A1, A2.

12.4.1 Učenie URL Agenta

Učenie URL agenta prebieha v paneli akcií **Teach URL Agent** (viď Obrázok 24.), ktorý zobrazíte kliknutím v paneli **Solution** na tlačidlo **URL Agent** a potom v paneli akcií na tlačidlo **Teach**.



Obrázok 24. Panel akcií Teach URL Agent

Proces učenia URL agenta:

1. Navigujte na prvú stránku, ktorá obsahuje zoznam záznamov, ktoré chcete extrahovať.
2. V paneli **Commands** kliknite na tlačidlo **This page contains links to my data** - tým oznámite agentovi skutočnosť, že táto stránka je prvá, ktorá obsahuje zoznam záznamov, ktoré chcete extrahovať.
3. V **okne prehliadača** kliknite na prvý odkaz zo zoznamu záznamov (z oblasti podobnej oblasti B – viď Obrázok 23).
4. Opakujte predošlý bod, až kým ste neoznačili všetky odkazy na stránke výsledkov, alebo až kým Vám program neponúkne voľbu:

Sú označené odkazy, tie ktoré hľadáte?

- a. Ak áno, kliknite na tlačidlo **Yes**. Systém sa práve naučil ako získavať URL záznamov zo stránky výsledkov. Na ďalších stránkach výsledkov už bude extrahovať URL samostatne. Systém automaticky prepel voľbu typu linku v v paneli **Type of picked URL** na voľbu **to next page**.
 - b. Ak nie, kliknite na tlačidlo **No** a pokračujte krokom 3.
5. Ak nie je v paneli **Akcií** v paneli **Type of picked URL** zvolená voľba **to next page**, učiňte tak. Tým dávate agentovi najavo, že mu idete demonštrovať odkaz na ďalšiu stránku výsledkov.
 6. V **okne prehliadača** kliknite (v oblasti podobnej A1 resp. A2 z Obrázka 23.) na URL, ktorá vedie na ďalšiu stránku výsledkov.

System prejde na túto ďalšiu stránku.

7. Ak systém nezbiera automaticky URL záznamov, pokračujte krokom 3.
8. Ak Vám systém ohlási: "URL Agent je naučený. Chcete ho uložiť?"
 - a. Ak áno, kliknite na tlačidlo **Yes**. URL agent sa práve naučil ako získavať ďalšiu stránku výsledkov a učenie je na konci- URL agent je uložený vo Vašom solution.
 - b. Ak nie, kliknite na tlačidlo **No** a pokračujte krokom 6.

12.4.2 Spustenie URL Agentu

Ak je v Solution uložený URL agent spustíte ho nasledovne:

1. V paneli **Solution** kliknite na tlačidlo **URL Agent**.
2. V paneli akcií kliknite na tlačidlo **Run**.

System Vám zobrazí panel akcií **Input URLs - Manual Mode** a spustí naučeného URL agenta. Ten uloží získané URL do zoznamu pracovných URL.

12.5 Data Agent

Data agent je nástroj, ktorým program Web Data Extractor extrahuje dáta z HTML dokumentov. Na narábanie s ním slúži panel akcií **Data Agent** (vid' Obrázok 25). Umožňuje Vám dve základné činnosti data agenta, a to učenie a extrakciu dát. Panel si zobrazíte kliknutím v paneli **Solution** na tlačidlo **Data Agent**.

Data Agent

Teach

Run

Teaching Statistics

Date

30.03.2011 18:50:51

Duration

0:02:48.0812

Records

3

Last Execution Report

	Docs	Recs	Atts
Total	140	140	280
Incorrect		12	12
Fail Ratio [%]		8,57	4,29
Speed [/s]	1,44	1,44	2,88
Date	30.03.2011 18:54:00		
Duration	0:01:37.0168		

Obrázok 25. Panel akcií Data Agent

12.5.1 Učenie Data Agenta

Ak vo Vašom solution existujú aspoň 3 URL, sprístupní sa Vám možnosť učiť data agenta. Na to slúži panel akcií **Teach Data Agent** (viď Obrázok 26.). Ten sa Vám sprístupní po kliknutí v paneli **Solution** na tlačidlo **Data Agent** a potom v **paneli akcií** na tlačidlo **Teach**.

Teach Data Agent

Save Cancel

Commands

Extract NextRecord

Show Suggestion

Teaching Progress

	Attribute	Record
Current	2/2	3/140
Guessed	2/2	1/2

Your Agent is considered as learned after 2 records guessed OK.

Show advanced information

☐ Context ☐ Agent Details

☐ Previous Record ☐ HTML

Options

☒ require column name at first extract

Obrázok 26. Panel akcií Teach Data Agent

Agentu naučíte veľmi jednoducho. Načíta sa Vám prvý HTML dokument z Vášho solution a na prvom zázname budete označovať jednotlivé dáta. Systém bude po celú dobu pozorne sledovať, ktoré časti dokumentu označujete a tie analyzovať. Ak budete hotoví, dáte učiacemu sa data agentovi najavo, že prechádzate na druhý záznam. Agent Vám na požiadanie môže označovať časti HTML dokumentu, o ktorých si myslí, že sú dátami, ktoré chcete extrahovať. Ak nie sú, nič sa nedeje, poučíte agenta tak, že mu ukážete extrakciu dát, ktoré mal označiť. Ak sa nevyskytne problém a agent sa dokáže naučiť extrahovať všetky Vami zadané vlastnosti záznamov z HTML dokumentu, učenie sa končí a Vy môžete pokračovať ďalej spustením tohto naučeného data agenta na všetky URL záznamy zo vstupu.

Proces učenia data agenta:

1. Systém Vám zobrazí možnosť voľby. Vyberte, prosím, jednu z možností, ktorá najlepšie zodpovedá situácii:
 - a. **1 URL has 1 record** — v jednom spracovávanom HTML dokumente sa nachádza 1 extrahovaný záznam.

- b. **1 URL has N records** — v jednom spracovávanom HTML dokumente sa nachádza viacero extrahovaných záznamov.

V prípade, že sa rozhodnete nepokračovať kliknite na tlačidlo na **Cancel** - pre vrátenie sa späť. V opačnom prípade, systém zobrazí prehliadač a načíta do neho prvú URL.
2. V okne prehliadača označte prvú hodnotu, ktorú chcete extrahovať.
3. V paneli kliknite na tlačidlo **Extract**, tým uložíte dáta do výsledku.
 - a. Ak spracováвате prvý záznam a máte zaškrtnutú voľbu **require column name at first extract**, systém Vás vyzve na zadanie názvu atribútu záznamu, ktorý práve extrahujete.
 - b. Zozbierané dáta počas učenia sa Vám zobrazujú v **informačnom paneli** v záložke s názvom **Data**.
 - c. Ak je už data agent dostatočne naučený, je pripravený ponúknuť Vám svoju predpoveď. Tú si zobrazíte stlačením tlačidla **Show Suggestion**.
4. Opakujte kroky 2-3 so všetkými hodnotami, ktoré chcete extrahovať pre aktuálny záznam.
5. Na spracovávanie ďalšieho záznamu sa posuniete tlačidlom **Next Record**,
6. Opakujte kroky 2 až 5 pre ďalšie záznamy, kým Vám systém neoznámí, výsledok učenia agenta:
 - a. Data agenta sa podarilo naučiť a bude uložený. Odteraz môžete spúšťať data agenta na extrakciu dát z dokumentov zo vstupného zoznamu URL.
 - b. Data agenta sa nepodarilo naučiť, proces učenia je nutné opakovať znova.

Poznámky:

- Data agent robí na pozadí predpoveď dát, ktorú porovnáva a analyzuje s Vami označenými dátami.
- Priebeh učenia môžete sledovať v paneli **Akcií** v paneli **Teaching Progress**, kde sa nachádza jednoduchá tabuľka:
 - V jednotlivých stĺpcoch sú zobrazené hodnoty pre:
 - **Record** — záznam,
 - **Attribute** — atribút.
 - V jednotlivých riadkoch sú zobrazené hodnoty pre:
 - **Current** — aktuálny záznam/atribút z celkového počtu,
 - **Guessed** — počet správne predpovedaných záznamov resp. atribútov z celkového počtu.
- Učenie sa končí úspechom v prípade, že v riadku **Record** v stĺpci **Predicted** všetky predpovedané záznamy v rade sú predpovedané správne a dosahujú stanovenú hranicu počtu záznamov, ktorá je uvedená v paneli **Teaching Progress**, a ktorú je možné nastaviť posuvníkom v pravej časti.

Zrušenie procesu učenia:

1. V paneli **Akcií** kliknite na tlačidlo **Cancel**.

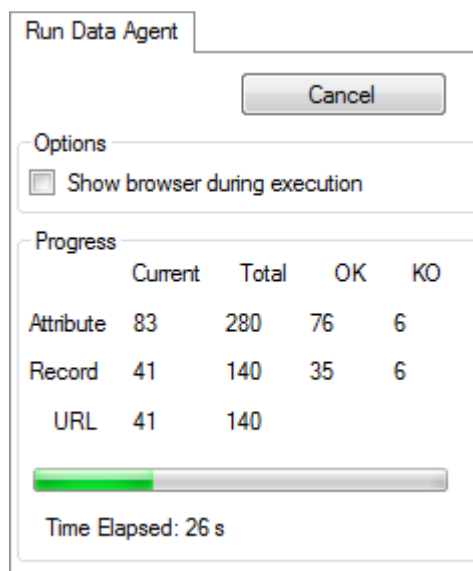
Možnosti zobrazenia počas procesu učenia:

Pomocou volieb v paneli **Show Advanced Information** Počas procesu učenia môžete zobraziť rôzne doplňujúce informácie:

- **Context** — zobrazí podrobnosti o kontexte aktuálneho označenia.
- **Agent Details** — v informačnom paneli zobrazí záložku **Agent Details**, ktorá obsahuje podrobnosti o skladbe práve učeného data agenta.
- **Previous Record** — zobrazí predošlý záznam v prehliadači hneď vedľa aktuálneho.
- **HTML** — v informačnom paneli zobrazí záložku s názvom HTML, v ktorej je štruktúrované zobrazený zdrojový kód aktuálneho HTML dokumentu.

12.5.2 Spustenie Data Agent

V prípade, že je v Solution uložený naučený data agent a solution obsahuje tiež neprázdny zoznam URL sa Vám sprístupní panel **Run Data Agent** (viď Obrázok 26) po kliknutí v **Solution** paneli na tlačidlo **Data Agent** a následne v **paneli akcií** na tlačidlo **Run**. Extrahované dáta máte možnosť vidieť v informačnom paneli v záložke **Data**.



Obrázok 27. Panel akcií Run Data Agent

Zobrazenie prehliadača počas extrakcie:

1. V paneli **Akcií** v paneli **Options** zaškrtnite voľbu **Show browser during execution**.

12.6 Uloženie extrahovaných dát

Extrahované dáta máte možnosť uložiť v požadovanom formáte v paneli akcií **Output Data**, (viď Obrázok 28). Panel zobrazíte, ak v paneli **Solution** kliknete na tlačidlo **Output Data**.

Output

Save

Destination

C:\Users\Tommy\Desktop\test Pick

Options

☒ remove empty rows

Format

☒ CSV

☐ quote cells " Delimiter : ;

☐ include column names

☐ include URLs

☐ SQL

SQL Command :

INSERT INTO Table (Col1,Col2)
VALUES ({0});

Regenerate from data description

1. Obrázok 28. Panel akcií Output Data

12.6.1 Základné nastavenia výstupu

Ak chcete uložiť výstup do súboru:

V prípade, že data agent skončil svoje vykonávanie, zanechal po sebe nejaké dáta a je špecifikovaná cesta k výstupnému súboru:

1. V paneli **Akcií** kliknite na tlačidlo **Save**.
2. Systém uloží extrahované dáta do súboru na disku, ktorý ste zadali za použitia ostatných nastavení výstupu.

Ak chcete špecifikovať cestu k výstupnému súboru:

1. V paneli **Akcií** kliknite na tlačidlo **Pick**.
2. Systém Vás vyzve na špecifikovanie cesty, kam sa majú uložiť výsledky extrakcie.
2. Po potvrdení cesty systém uloží extrahované dáta do súboru, ktorý ste zadali za použitia nastavení výstupu.

Ak nechcete vo výsledku zahrnúť prázdne riadky:

1. V paneli **Akcií** v paneli **Options** zaškrtnite voľbu **remove empty rows**.

12.6.2 Nastavenia výstupu v CSV formáte

Ak chcete vybrať formát CSV:

1. V paneli **Akcií** v paneli **Format** zaškrtnite voľbu **CSV**.

Ak chcete uviesť dáta v úvodzovkách:

1. V paneli **Akcií** v paneli **Format** zaškrtnite voľbu **quote cells**.
Systém vypíše každú bunku výstupných dát v úvodzovkách.

Ak chcete špecifikovať znak pre úvodzovky:

1. V paneli **Akcií** v paneli **Format** do políčka za voľbou **quote cells** vpíšte Vami požadovaný znak pre oddelovač.
Systém vypíše každú bunku výstupných dát uvedenú Vami špecifikovanými úvodzovkami.

Ak chcete špecifikovať znak pre oddelovač buniek:

1. V paneli **Akcií** v paneli **Format** do políčka za **Delimiter** vpíšte Vami požadovaný znak.
Systém vypíše každú bunku výstupných dát oddelenú Vami špecifikovaným oddelovačom.

Ak chcete zahrnúť názvy stĺpcov:

1. V paneli **Akcií** v paneli **Format** zaškrtnite voľbu **include column names**.

Systém zahrnie do výstupného súboru na prvý riadok názvy extrahovaných polí.

Ak chcete URL odkiaľ bola vykonaná extrakcia záznamu:

1. V paneli **Akcií** v paneli **Format** zaškrtnite voľbu **include URLs**.
Systém zahrnie do výstupného súboru na prvý stĺpec URL, z ktorej bol záznam extrahovaný.

12.6.3 Nastavenia výstupu v SQL formáte

Ak chcete vybrať formát SQL:

1. V paneli **Akcií** v paneli **Format** zaškrtnite voľbu **SQL**.

Ak chcete definovať formát výstupného SQL súboru:

2. V paneli **Akcií** v paneli **Format** definujte políčko **SQL command** nasledovne.
Systém bude v nasledujúcom kroku generovať textový súbor s príponou .sql kde na každom riadku budú uvedený SQL command, v ktorom bude podreťazec '{0}' nahradený hodnotami atribútov uvedenými v úvodzovkách oddelenými čiarkou.

Príklad:

SQL Command = "INSERT INTO Table(Column1,Column2,Column3) VALUES ({0});"

Výsledok= "INSERT INTO Table(Column1,Column2,Column3) VALUES ('ABC','DEF','GHI');"

SQL Command môžete zadať nasledovnými dvoma spôsobmi:

- automatickým vygenerovaním hodnoty - kliknutím na tlačidlo **Regenerate from data description**,
- ručne.

12.7 Spustenie celého Solution

V prípade, že je v Solution uložený naučený data agent a je nastavený zdroj URL:

1. V paneli **Solution** kliknite na tlačidlo **Run**.
Systém Vás vyzve na špecifikovanie cesty, kam sa majú uložiť výsledky extrakcie.

2. Po potvrzení cesty systém spustí získavanie URL a následne data Agentu a extrahované dáta uloží do súboru, ktorý ste zadali. V paneli **Akcií** Vám postupne zobrazia panely **URL Input - Automated Mode**, **Run Data Agent** a **Output Data**.

Príloha B. CD nosič

Popis obsahu priloženého CD:

- **Documents** — obsahuje elektronické verzie dokumentov týkajúcich sa systému:
 - o **WebDataExtraction.MasterThesis.pdf** — elektronická podoba tohto dokumentu,
 - o **WDE-PBD.Userguide.pdf** — užívateľská príručka,
 - o **ProgramDocumentation** — adresár obsahujúci programovú dokumentáciu:
 - **index.html** — úvodná stránka programovej dokumentácie.
- **Install** — zložka inštalácie systému obsahuje:
 - o **WDE-PBD.InstallationManual.pdf** — inštalačná príručka systému,
 - o **WDE-PBD.0.9.setup.exe** — inštalačný súbor systému.
- **TestData** — testovacie dáta
- **Project** — adresár projektu Microsoft Visual Studio 2010 systému WDE-PBD
- **README.txt** — súbor s popisom obsahu CD