VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EXTRAKCE DAT Z POPISU ZBOŽÍ

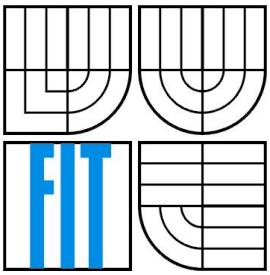
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. VOJTĚCH SLÁMA

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EXTRAKCE DAT Z POPISU ZBOŽÍ

DATA EXTRACTION FROM PRODUCT DESCRIPTIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. VOJTĚCH SLÁMA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. RADEK BURGET, PH.D.

BRNO 2008

Abstrakt

Tato práce se zabývá návrhem a implementací systému pro automatizovanou podporu sběru informací o zboží pro účely elektronických obchodů. Uvádí přehled existujících přístupů pro extrakci informací z HTML dokumentů, zejména se zaměřuje na wrappery a metody jejich automatické konstrukce. Zmíněn je i vizuální přístup k extrakci dat z dokumentů. V části zabývající se návrhem jsou formalizovány požadavky na systém a navrženy základní principy systému. Implementační část obsahuje podrobný popis algoritmu pro hledání cest ve stromu dokumentu. V závěru práce jsou zhodnoceny výsledky dosažené při experimentech.

Klíčová slova

Extrakce informací, wrapper, indukce wrapperu, elektronický obchod, e-shop, JavaScript, DOM.

Abstract

This work concentrates on the design and implementation of an automated support for data extraction from product descriptions. This system will be used for e-shop purposes. The work introduces present approaches to information extraction from HTML documents. It focuses chiefly at wrappers and methods for their induction. The visual approach to information extraction is also mentioned. System requirements and basic principles are described in the design part of the work. Next, a detailed description of a path tracing algorithm in document object model is explained. The last section of the work evaluates the results of experiments made with the implemented system.

Keywords

Information extraction, wrapper, wrapper induction, webshop, e-shop, JavaScript, DOM.

Citace

Sláma Vojtěch: Extrakce dat z popisu zboží. Brno, 2008, diplomová práce, FIT VUT v Brně.

Extrakce dat z popisu zboží

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D. Další informace mi poskytl Ing. Oto Škrkal.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Obsah

1 Úvod

Po úspěšném obhájení mé bakalářské práce, zaměřené na vývoj informačního systému elektronického obchodu, byl vytvořený systém nasazen a v současnosti je úspěšně používán na několika doménách. Díky zpětné vazbě od uživatelů, bylo možné systém dále upravovat a rozvíjet tak, aby co nejlépe odpovídal jejich potřebám. To umožnilo nahlédnout do procesů probíhajících při řízení a správě několika internetových obchodů a identifikovat tak činnosti, které patří mezi kritické, ať již z pohledu organizační, finanční, resp. časové náročnosti, nebo z hlediska kvalifikace pracovníků.

Jednou z časově nejnáročnějších činností při správě obchodu je tvorba a aktualizace databáze. Přidávání nových výrobků do sortimentu obchodu vyžaduje často zdlouhavý manuální proces. Informace o výrobcích jsou nejčastěji čerpány ze stránek výrobce, specifikací, recenzí výrobku, případně jiných obchodů prodávajících obdobný sortiment. V rámci webu tedy existuje dostatek zdrojů dat, z nichž většina obsahuje informace ve formě, která umožňuje strojové zpracování. Využitím metod data miningu, konkrétně metod určených pro extrakci informací z webu (web content mining), lze tyto zdroje využít a proces aktualizace tak může být alespoň částečně automatizován.

Vytvoření systému pro extrakci údajů o zboží by mělo praktický přínos nejen pro majitele a pracovníky elektronických obchodů, ale i pro jejich zákazníky. V současné době totiž velká část obchodů ukládá samostatně pouze základní parametry a ty, které souvisí s konkrétním druhem výrobku jsou ukládány samostatně mnohem méně často. Je to hlavně dáno vysokou časovou náročností. Využitím automatické extrakce by bylo možné získat větší počet parametrů výrobku mnohem rychleji, což by pro zákazníky obchodu znamenalo podrobnější a sofistikovanější možnosti při vyhledávání a filtrování výrobků.

Protože je extrakce informací perspektivní a pro mne osobně navíc zajímavou oblastí, zvolil jsem systém pro extrakci dat z popisu zboží jako téma své diplomové práce. Je potěšující, že vyvinutý systém bude mít významný praktický přínos pro uživatele.

Následující kapitola je zaměřena na problematiku extrakce informací obecně, sleduje vývoj v této oblasti v minulosti a detailněji se zabývá významnými směry současného výzkumu. Wrappery, které se používají zejména pro extrakci dat z HTML dokumentů, tvoří spolu s metodami pro jejich automatickou konstrukci hlaví téma kapitoly. Zmíněn je i vizuální přístup k extrakci dat, tvorba modelů rozvržení stránky a vlastností textu.

Ve třetí kapitole se podrobně zabývám zdroji dat, které lze pro vytvářený systém využít. Je provedena klasifikace podle jejich vlastností, kvality a formy poskytovaných dat. V další části jsou popsány používané formy publikace dat o výrobcích, včetně jejich modifikací a jsou doporučeny ty varianty, které jsou pro extrakci nejvhodnější.

Čtvrtá kapitola je věnována formalizaci požadavků na systém. Jsou zde rozebrány funkce a vlastnosti, které má systém splňovat. Na základě těchto požadavků je proveden návrh základních tříd a uživatelského rozhraní a je vytvořena sada diagramů modelujících systém.

V páté kapitole je popsána implementace klientské aplikace i serverové části systému. Hlavní část kapitoly je věnována algoritmu extrakce dat a práci s extrakčními pravidly. Poslední část kapitoly je zaměřena na zajímavé problémy, které bylo nutné v průběhu implementace vyřešit.

Šestá kapitola obsahuje stručné vyhodnocení extrakce prováděné v rámci experimentů a poukazuje na vlastnosti algoritmu i celého systému, které je možné v rámci dalšího rozvoje vylepšit.

2 Extrakce dat z dokumentů

Extrakce informace je technologie založená na analýze přirozeného jazyka a jejím účelem je z textových dokumentů (mohou být nestrukturované či polostrukturované) získat informace ve strukturované, předem dané, formě, kterou lze strojově lépe zpracovat. Výsledná data pak mohou být přímo zobrazena uživateli (např. automatická sumarizace textu), nebo uložena v databázi pro pozdější analýzu (např. indexace webových stránek internetovými vyhledávači).

Extrakce informací z elektronických dokumentů byla zkoumána již dlouho před rozmachem World Wide Webu. V 90.letech 20.století vznikla díky velkému nárůstu elektronické komunikace potřeba hromadně analyzovat a efektivně zpracovávat texty. Tato data byla reprezentována hlavně rozsáhlými databázemi news konferencí, emaily, vojenskými zprávami, novinovými články apod. [1] Hlavním cílem bylo tedy vytvořit metody pro efektivní zpracování přirozeného jazyka.

2.1 Historie extrakce informací

O velký přínos v oblasti extrakce informací se zasloužila série konferencí MUC (*Message Understanding Conference*)[1] a MET (*Multilingual Entity Task Evaluation)*. Účast na MUC resp. MET byla pro vývojové týmy podmíněna vytvořením funkčního systému pro extrakci informací z dokumentů. Zadání je zveřejněno vždy s dostatečným předstihem před samotnou konferencí. Zadání specifikuje *tématickou oblast* (*doménu*) vstupních dokumentů, jejich formát a informace, které je třeba extrahovat. Každá vývojová skupina také dostane ukázkové zprávy, určené pro testování systému, včetně jejich vzorového, ručně zpracovaného řešení.

Před začátkem konference obdrží skupiny sadu testovacích dokumentů, které nechají systémem zpracovat. Po obdržení testovacích zpráv již na systému nemohou provádět žádné dodatečné úpravy. Výsledky jednotlivých skupin organizátoři zpracují, vyberou několik nejúspěšnějších týmů, které poté přizvou k účasti. Na konferenci proběhne oficiální vyhodnocení a týmy prezentují své řešení. Diskutují se dosažené výsledky, možný budoucí vývoj a stanovují se také cíle úloh pro další konferenci.

MUC-1 se konala v roce 1987 a jejími organizátory byly skupiny *Naval Research and Development Group*[2] *(NRaD) a Defense Advanced Research Projects Agency*[3] *(DARPA)*. Program konference byl zaměřen na výzkum systému pro automatické získávání informací z vojenských zpráv.

[1] MUC - http://en.wikipedia.org/wiki/Message_Understanding_Conference
[2] NRaD - skupina pro námořní výzkum a vývoj
[3] DARPA - vládní organizace pro podporu vývoje projektů obrany státu

Protože zadání úloh nedefinovalo formu výsledných dat a každý z účastníků navrhnul vlastní formát, neproběhlo ani oficiální vyhodnocení výsledků.

Na konferenci MUC-2 v roce 1989, jejíž doménou byly opět vojenské zprávy, byly zavedeny *šablony (templates)*. Šablona obsahovala 10 volných *oddílů (slots)*, do kterých se ukládaly extrahované informace z dokumentu (typ události, místo, čas, apod.). Vyplnění oddílu nebylo povinné. Zavedením šablon došlo ke sjednocení formátu výsledků a bylo možné přesně definovat míru schopnosti nalézt informaci. Následují vzorce použité pro vyhodnocení [2]:

$$\text{recall} = \frac{N_{correct}}{N_{key}}$$

$$\text{precision} = \frac{N_{correct}}{N_{correct} + N_{incorrect}}$$

N_{key} = počet vyplněných pozic
$N_{correct}$ = počet správně vyplněných pozic
$N_{incorrect}$ = počet nesprávně vyplněných pozic

Vzorec č. 1 - Vzorce použité pro hodnocení systémů na MUC

Konference MUC-3 (1991) a MUC-4 (1992) byly zaměřeny na dokumenty informující o teroristických útocích ve Střední a Jižní Americe. Složitost šablon se postupně zvyšovala (18, resp. 24 oddílů).

V pořadí pátá Message Understanding Conference (MUC-5, 1993) byla organizována jako součást vládního programu Tipster[4] a významně zvyšovala složitost zadaných úloh. Systémy musely být připravené kromě angličtiny také na japonštinu. Byla také zavedena hierarchická struktura šablon, protože jediná šablona s mnoha oddíly nedokázala dostatečně zachytit strukturu stále složitějších dokumentů. To také umožnilo lepší návrh databáze, do kterých byly extrahované informace ukládány. Počet šablon pro zadané úlohy byl celkem 11 a počet oddílů 47. Doména analyzovaných dokumentů byla výroba elektronických obvodů a podniky se zahraniční účastí

Konference MUC-1 až 5 se zaměřovaly na analýzu a identifikaci specifických událostí v přirozeném textu. Šablony pro hledané informace byly však stále složitější a navíc bylo zřejmé, že doménová závislost vyvinutých systémů není žádoucí. Bylo nutné zvolit pro extrakci informací obecný přístup a vyvinout takové technologie, které budou použitelné nezávisle na typu úkolu.

Pro konferenci MUC-6 (1995), konané opět v rámci programu Tipster, bylo proto zadáno několik úloh[5]:

- *named entity recognition* – jazykově nezávislá úloha pro rozpoznávání jmenných entit a jejich typu (jména osob a organizací, názvy geografických míst, časové a množstevní údaje)

[4] Tipster - vládní program USA pro výzkum a vývoj v oblasti získávání a vyhledávání informací
[5] MUC-6 - http://cs.nyu.edu/cs/faculty/grishman/muc6.html

- *coreference* – nalezení a propojení výrazů odkazujících na stejnou jmennou entitu
  template elements - nalezení dostupných informací v textu o entitě a jejich vyplnění do šablony prvků
- *scenario templates (traditional information extraction)* – identifikace událostí v textu a jejich napojení na participující entity uložené v šablonách prvků

Doménami MUC-6 byly změny ve výkonném managementu společností, objednávky letadel a vyjednávání mezi odbory a vedením společnosti. Témata byla zveřejněna pouze měsíc před samotnou konferencí a to proto, aby vývojové týmy nemohly systém příliš zoptimalizovat na danou problematiku.

V roce 1998 se konala zatím poslední konference MUC-7, která zavedla novou úlohu *template relations*[6]. Ta byla zaměřena na vztahy mezi entitami, jejichž nalezení a rozpoznání by umožnilo hlubší pochopení textu. Pro účely testování byly brány v úvahu pouze vztahy vázané k entitě společnosti (je umístěna, je produktem, zaměstnává). Za domény konference byly zvoleny havárie letadel a starty raket, resp. řízených střel.

Přestože zadání MUC-5 bylo definováno pro angličtinu a japonštinu, systémy pro MUC-6 a 7 byly vyvíjeny opět pouze pro dokumenty v anglickém jazyce. Pro analýzu textů v jiných jazycích byla vytvořena samostatná konference s názvem *Multilingual Entity Task Evaluation*, jejíž organizace a průběh byl podobný jako u MUC. Jak již z názvu konference vyplývá, MET jsou zaměřeny pouze na jeden krok získávání informací a to na rozpoznávání jmenných entit. MET-1 (1996) byla zacílena na čínštinu, japonštinu a španělštinu, u MET-2 (1998), probíhající paralelně s MUC-7, byla španělština odebrána.

V následujících letech navázalo na sérii konferencí MUC a MET několik dalších projektů, zabývajících se vyhledáváním informací a jejich extrakcí z dokumentů. Byl to např. *Information Retrieval and Extraction Exercise (IREX*[7]*, 1998-1999*), série konferencí *Language Resources and Evaluation Conference* (*LREC*[8], od roku 1998), nebo *Conference on Computational Natural Language Learning* (*CoNLL*[9], od roku 1997). Jejich cílem v oblasti extrakce informací je především zlepšit výsledky metod pro identifikaci jmenných entit a to specializací na konkrétní jazyk, nebo tématickou doménu.

[6] MUC-7 - http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html
[7] IREX - http://nlp.cs.nyu.edu/irex/index-e.html
[8] LREC - http://www.lrec-conf.org/
[9] CoNLL - http://www.cnts.ua.ac.be/conll/

2.2 Extrakce informace z HTML dokumentů

Příchod WWW a jazyka HTML a jeho masivní rozvoj na konci 90. let 20. století, umožnil nový pohled na extrakci informací z dokumentů. Vstup pro předchozí metody tvořil většinou nestrukturovaný text a od toho se také odvíjela jeho analýza a zpracování. U dokumentů v jazyku HTML však lze jednoduše definovat jejich strukturu a vizuální podobu. Díky tomu lze při jejich analýze použít nejen text, ale i informace o jeho vizuálním formátování. Takové dokumenty nazýváme polostrukturované.

Postupy extrakce informací z textů v přirozeném jazyce, na které byl do té doby vývoj zaměřen, neměly pro HTML dokumenty uspokojivé výsledky. Je to dáno tím, že webové stránky obsahují ve větší míře izolované útržky informací, jejichž kontext je získán až dosazením do vizuálního toku dokumentu.

Bylo tedy nutné vyvinout takové postupy pro extrakci informací, které budou připraveny na jiný charakter HTML dokumentů. Hlavní snahy byly tehdy zaměřeny na použití *wrapperů*, které jsou podrobně rozebrány v sekci 2.2.1. Časté používání wrapperů bylo umožněno hlavně slabou podporou kaskádových stylů (CSS) ze strany webových prohlížečů. Pro tvůrce HTML stránek bylo běžné definovat strukturu a formátování dokumentu HTML značkami, protože při použití CSS neměli jistotu, že bude dokument na cílovém zařízení korektně zobrazen.

Jednoduchost jazyka HTML, spolu s jeho možnostmi vizuálně ovlivňovat tok textu, byla pro tvůrce dokumentů vždy velikou výhodou. Pro účely strojového zpracování však HTML není ideální a to i přesto, že obsahuje značky, z jejichž definice přímo vyplývá sémantika obsahu. Mezi takové značky můžeme zařadit např. nadpisy <h1> až <h7>, odstavce <p>, seznamy <ul>, <ol> apod. S rozvojem technologií kaskádových stylů (CSS), již nejsou tvůrci webových stránek na používání těchto značek vázaní. Uzavírají obsah do sémanticky neutrálních značek , resp. a vzhled dokumentu se tím přesouvá mimo HTML kód. Struktura takového dokumentu je pak bez dalších informací nerozlišitelná a jeho zpracování wrapperem již není tak efektivní.

```
<h1>Nadpis</h1>
<h2>Podnadpis</h2>
<p>Text odstavce a <strong>zvýrazněný text.</strong></p>

<div id='nadpis'>Nadpis</div>
<div class='text-velky'>Podnadpis</div>
<div>Text odstavce a <span class='barevny'>zvýrazněný text.</span></div>
```

Obrázek č. 1 - Přesun formátování dokumentu z HTML do CSS

2.2.1 Wrappery

Slovo wrapper znamená v angličtině obal resp. obálka, což dobře charakterizuje princip jeho činnosti. Wrapper můžeme chápat jako proceduru, která vstupní dokument transformuje na n-tice, vyhovující množině *extrakčních pravidel*. Při průchodu dokumentem jsou vybírány ty části, které jsou obklopeny řetězci z množiny extrakčních pravidel.

Všechny dokumenty vyhovující jedné množině pravidel nazýváme *třída dokumentů*. Ve většině případů se jedná o stránky v rámci jednoho informačního zdroje (např. e-shop), které jsou generovány stejným skriptem. Pro každou třídu dokumentů je třeba vytvořit příslušná extrakční pravidla.

HTML kód:

```
<html>
<head><title>Výsledky běhu 100m</title></head>
<body>
<b>Pavel</b><i>11,40 s</i><br />
<b>Vojta</b><i>11,59 s</i><br />
<b>Martin</b><i>11,80 s</i><br />
<b>Radek</b><i>12,21 s</i><br />
</body>
</html>
```

Wrapper:

```
W = {[<b>, </b>], [<i>,</i>]}
```

Výstup:

```
O = {[“Pavel”, “11,40 s”],
     [“Vojta”, “11,59 s”],
     [“Martin”, “11,80 s”],
     [“Radek”, “12,21 s”]}
```

Obrázek č. 2 – Ukázka jednoduchého LR wrapperu a jeho výstupu

Výše uvedený příklad je ukázkou wrapperu typu LR (zkratka slov left-right). Podle [3] lze wrappery rozdělit do 6 tříd:

- **LR** (left-right) – Pro každou extrahovanou hodnotu existuje pravidlo, složené z dvojice řetězců, z nichž první odděluje hodnotu od zbytku dokumentu zleva, druhý zprava. Algoritmus prochází dokument a vybírá hodnoty obsažené mezi oddělovači.
- **HLRT** (head-left-right-tail) – Tato třída je velmi podobná třídě LR. Používá stejný algoritmus, pouze s omezením na specifickou část dokumentu, což přináší vyšší efektivitu při zpracování rozsáhlých stránek a snížení množství redundantních dat. Část dokumentu použitou pro extrakci vymezují oddělovače *head* a *tail*.
- **OCLR** (open-close-left-right) – U této třídy jsou zavedeny oddělovače *open* a *close*, které určují specifickou oblast extrakce pro jednotlivé n-tice. Tím se liší od třídy HLRT, kde je specifická oblast nastavena v rámci celého dokumentu.
- **HOCLRT** (head-open-close-left-right) – Kombinace tříd HLRT a OCLR, u kterých je extrakce prováděna z části dokumentu ohraničené řetězci *head*, *tail*, ve které jsou vynechány místa, ležící mimo oddělovače *open* a *close*.

- **N-LR** (nested-left-right) – Předchozí třídy jsou vhodné pro data s opakující se strukturou, u kterých má každá n-tice získaných dat stejnou délku. Příkladem vhodných dat jsou např. tabulky. N-LR wrappery berou v úvahu data, ve kterých jsou n-tice s rozdílnou délkou. Typickým příkladem jsou víceúrovňové seznamy.
- **N-HLRT** (head-left-right-tail) – analogická kombinace tříd N-LR a HLRT.

Obrázek č. 3 - Zobrazení průchodu dokumentem pro 6 základních tříd wrapperů

Jak je vidět, hlavní nevýhodou wrapperů je úzká vazba mezi extrakčními pravidly a strukturou dokumentu. Pokud je dokument byť i nepatrně změněn, nemusí mu již množina pravidel odpovídat. Extrahovaná data v takovém případě nebudou korektní, nebo proces extrakce zcela selže. Poté je třeba pravidla aktualizovat, a to buď ručně (což je v současnosti nejčastější varianta), nebo za pomoci některého z automatických postupů strojového učení.

Pro proces tvorby wrapperu pomocí strojového učení je nutné mít k dispozici množinu dokumentů patřících do stejné třídy a aplikovat na ni algoritmus, který se z nich extrakční pravidla naučí. Ze získaných pravidel lze vygenerovat wrapper pracující nad touto třídou dokumentů. V reálné situaci je často pro naučení pravidel dostupný malý počet dokumentů (často však jen jeden), což lze řešit svépomocí a vytvořit testovací množinu vlastní.

Pravidla pro extrakci mohou být reprezentována několika způsoby. Pokud zdrojový kód dokumentu bereme jako řetězec znaků, jsou pravidla obvykle ve tvaru *řetězce*, případně *regulárního výrazu* [4]. Tento způsob je obvyklý u manuální tvorby pravidel. V případě automatického přístupu je vhodnější nahlížet na dokument jako na sekvenci *slov [5][6]*. Slova jsou tvořena částmi zdrojového kódu dokumentu, což jsou např. HTML elementy, čísla, textové řetězce apod. Těmto základním jednotkám jsou dále přiřazeny vlastnosti, na základě jejich lexikálních vlastností (délka slova, velikost písmen apod.) i v závislosti na jejich kontextu v dokumentu (nadpis, poznámka v závorce apod.). Jedním z nejčastějších způsobů, jak lze na HTML dokument pohlížet je Document Object Model (DOM), ve které je každý element HTML stránky reprezentován jedním uzlem ve stromové struktuře. Extrakční pravidla pak můžeme modelovat např. jako *podstromy* v této struktuře [7].

2.2.2 Automatická konstrukce wrapperů

Manuální vytváření wrapperů je zdlouhavá práce, u které je nutné provést analýzu zdrojového kódu dokumentů. Je třeba nalézt části kódu dostatečně specifické, které lze použít pro vytvoření pravidel. Takový postup je náchylný na chyby a navíc jsou vytvořené wrappery úzce vázány na zdrojový kód. Pokud se změní struktura zpracovávaných dokumentů, je nutné wrapper upravit.

Proto vnikla potřeba mít možnost získat extrakční pravidla automaticky, nejlépe pak na takové úrovni abstrakce, která není tolik závislá na struktuře dokumentu. V současnosti je již k dispozici celá řada postupů pro automatickou analýzu dokumentu, ze kterých budou podrobněji zmíněny metody založené na *gramatické inferenci* a *Markovových modelech*. Mezi další přístupy lze zařadit *konceptuální modelování* [8], vyvinuté primárně pro nestrukturované dokumenty, může být však použito i pro HTML stránky.

2.2.2.1 Gramatická inference

Gramatická inference je úloha využívající postupů strojového učení a umělé inteligence. Jejím cílem je vytvoření konzistentní gramatiky pro neznámý jazyk na základě konečné množiny příkladů.

Vytvoření wrapperu pro určitou třídu HTML stránek odpovídá úloze odvození gramatiky, která tuto třídu generuje. Takto je možné zachytit strukturu dokumentu na obecnější úrovni, která není tak úzce svázaná s jeho zdrojovým kódem. Vstupem algoritmu je množina pozitivních příkladů (stránky z přijímané třídy) a negativních příkladů. Výstupem algoritmu je gramatika a to buď regulární, nebo bezkontextová.

Tento přístup není bohužel pro webové stránky použitelný přímo. Pokud je totiž k dispozici pouze množina pozitivních příkladů (to jsou již existující stránky), nelze požadovanou gramatiku vytvořit v konečném čase [9]. Tento problém lze vyřešit dodáním umělé množiny negativních příkladů člověkem, což je nepraktické, protože automatická tvorba wrapperu opět vyžaduje manuální práci. Druhou možností je omezit jazyk vytvářené gramatiky na podtřídu regulárních jazyků, které lze v konečném čase odvodit. V tomto případě není množina negativních příkladů potřeba, avšak pro vstupní dokument nemusí gramatika existovat.

Postupy založené na gramatické inferenci můžeme nalézt např. v kombinaci s použitím Bayesova klasifikátoru [10], stochastických bezkontextových gramatik [5] nebo stromových modelů dokumentu [11].

2.2.2.2 Skryté Markovovy modely

Markovův model je konečný automat, u kterého jsou přechody mezi jednotlivými stavy dány přechodovou maticí. Při přechodu je vygenerován s určitou pravděpodobností symbol, po průchodu automatem je tedy vytvořen celý vektor symbolů (řetězec). Pokud známe výsledný vektor symbolů, ale ne sekvenci stavů, nutnou pro jeho generování, nazýváme model skrytý (*Hidden Markov Model*,

zkráceně *HMM*). Charakter generovaných symbolů je různý, záleží na typu úlohy. Může to být znak, řetězec, symbol definované abecedy, uzel v grafu apod.

Použití HMM při extrakci informací je založeno na myšlence, že vstupní dokument lze vytvořit stochastickým procesem a my se snažíme nalézt Markovův model tohoto procesu. Stavy použité pro konečný automat jsou rozděleny na ty, které generují symboly určené pro extrakci (*target*) a symboly pro nás nezajímavé (*background*). Nancy R. Zhang (2001) používá navíc další stavy *prefix* a *suffix*, které generují symboly vyskytující se před, resp. za symboly pro extrakci.

Při učení známe výstupní vektor symbolů (text dokumentu) a snažíme se určit hodnoty přechodové matice takové, aby výsledný automat co nejlépe generoval výstupní dokument. Extrakce informace probíhá na stejném modelu za použití naučených parametrů. Nejdříve je třeba nalézt posloupnost přechodů, která nejlépe odpovídá dokumentu. Poté jej můžeme rozdělit na části generované stavy *target* a *background* a tak rozlišit extrahované informace.

2.2.3 Vizuální přístup k extrakci dat

S nástupem CSS se stávají tradiční přístupy využívané wrappery neefektivní. Informace, které popisují formátování dokumentu, se přesouvají z jazyka HTML do CSS stylů, často v externích souborech. Proto je vhodnější zvolit odlišný přístup a postup založený na vizuálním modelu dokumentu se jeví jako perspektivní řešení této situace [7]. Postup je navíc nezávislý na použití (resp. nepoužití) CSS stylů, protože bere v úvahu až finální vzhled dokumentu a v principu lze použít na dokumenty v jakémkoli formátu (PDF, RTF apod.)

Abychom byli schopni pracovat s vizuální informací v dokumentu, je třeba ji nejdříve formálně popsat a vytvořit takový model, který umožní s těmito informacemi snadno manipulovat. [7] řeší úlohu jejím rozdělením na několik částí. Nejdříve je sestaven *model rozvržení stránky (page layout model)*, poté *model vlastností textu (text features model)*. Jejich kombinací pak vznikne model logické struktury dokumentu (*logical document strucuture*).

2.2.3.1 Model rozvržení stránky

Při tvorbě stránek je autory kladen důraz hlavně na vizuální atraktivitu stránky, její přehlednost a dobrou orientaci návštěvníků v obsahu, nikoliv na to, jak snadno lze analyzovat její zdrojový kód. Stránky jsou většinou vizuálně rozděleny na několik oblastí, z nichž každá má v dokumentu jiný význam. Takovými oblastmi mohou být např. hlavička nebo patička stránky, navigace, reklama apod., přičemž tyto oblasti se mohou vnořovat. Klasifikovat takový dokument jako celek nedává příliš uspokojivé výsledky a je vhodnější zpracovávat obsah jednotlivých oblastí zvlášť. Obrázek č. 4 zobrazuje ukázku základního členění oblastí pro úvodní stránku serveru Novinky.cz.

Obrázek č. 4 - Logické členění dokumentu na vizuální celky

Segmentaci stránky na vizuálně související oblasti lze provést vytvořením stromu objektů v HTML dokumentu takových, které formují vlastní vizuální oblast. Mezi ně lze zařadit tabulky, seznamy, odstavce, rámce (jinak také framy), a nespecifické oblasti tvořené elementem . Kořen stromu reprezentuje oblast celého dokumentu, listy stromu nejmenší uvažované oblasti, např. buňky tabulky. Takový model neobsahuje konkrétní informace o vizuální podobě segmentů, bere však v úvahu jejich strukturu a zanoření.

Vytvořený model je většinou tvořen velkým množstvím uzlů. Dalším krokem proto může být spojování oblastí, které na stránce leží blízko sebe, nebo u kterých rozpoznáme, že nejsou z vizuálního hlediska významné (např. neobsahují žádný text). Redukce počtu uzlů tedy probíhá směrem zdola-nahoru, ale existují i opačné postupy, které stránku dělí na stále menší oblasti.

2.2.3.2 Model vlastností textu

Aby se čtenář dokázal v textu dokumentu a jeho hierarchii jednoduše orientovat, jsou jednotlivé části textu vizuálně odděleny. Tučné písmo, kurzívu a podtržení používáme k odlišení významných částí textu od zbytku dokumentu. Pro zvýraznění lze použít i vhodně zvolenou barvu. Velikost písma, souvisí s jeho zařazením ve struktuře dokumentu. Důležité nadpisy jsou tedy vysázeny velkým písmem, hlavní sdělení normálním písmem a případné poznámky pouze drobným

písmem. Díky těmto běžně používaným typografickým pravidlům lze na jakýkoli text pohlížet jednotným způsobem.

Pro účely automatického zpracování dokumentu je nutné vytvořit model vlastností textu. Model by však neměl popisovat konkrétní grafické atributy textu jako je jeho velikost a barva. Je žádoucí, aby interpretoval jejich vizuální postavení v rámci stránky. [7] pro tyto účely zavádí metriky *markedness* (zřetelnost) a *weight* (váha), které jsou přiřazeny každému textovému řetězci v HTML dokumentu a popisují atributy textu na obecné úrovni. Hodnota markedness udává, jak je element vůči svému okolí výrazný, weight oproti tomu bere v úvahu postavení elementu v hierarchii nadpisů.

$$\text{markedness} = (F \cdot \Delta f + b + o + u + c) \cdot (1 - z)$$
$$\text{weight} = [(F \cdot \Delta f) + (b + o + u + c) \cdot l + W \cdot p] \cdot (1 - z)$$

Vzorec č. 2 - Výpočet hodnot markedness (zřetelnost) a weight (váha)

Hodnoty b, o, u, c a z jsou rovny 1, pokud je daný text tučným písmem, kurzívou, podtržený, barevně zvýrazněný, nebo přeškrtnutý. V opačném případě je jejich hodnota 0. Parametr Δf udává rozdíl mezi velikostí písma textu a standardní velikostí v dokumentu. Pokud za textem následuje zalomení řádku, nebo dvojtečka, je hodnota l (resp. p) rovna 1, v opačném případě 0. Konstanty F a W definují váhu velikosti písma, resp. dvojtečky.

2.2.3.3 Předzpracování HTML kódu

Informace potřebné pro vytvoření výše uvedených modelů z HTML dokumentu nelze získat přímo. Zdrojový kód stránky je třeba upravit, než je možné přistoupit k vizuální segmentaci stránky a výpočtům hodnot markedness a weight. Při předzpracování jsou významné zejména tyto kroky:

- **Konverze značek na styly.** Převod takových HTML značek, které ovlivňují vzhled písma na ekvivalentní inline CSS styl. Do zdrojového kódu dokumentu jsou také dopočítány hodnoty CSS vlastností nastavené přes atributy *id* a *class*.
- **Sjednocení zápisu jednotek.** Některé CSS vlastnosti umožňují definovat ekvivalentní hodnoty několika různými způsoby. Příkladem mohou být ekvivalentní zápisy „color: red;" a „color: #f00;". Podobný problém existuje i u definice rozměrů, kde je navíc třeba číselné hodnoty při změně jednotky přepočítat.

2.2.3.4 Extrakce informací z logického modelu dokumentu

Protože již máme vytvořeny modely vlastností textu a rozvržení stránky, lze jejich kombinací získat logický model dokumentu. Ten bude reprezentován stromovou strukturou, jehož uzly tvoří hierarchii nadpisů dokumentu a listy obyčejný text dokumentu. Zanoření uzlů stromu odpovídá váze elementu

v rámci dokumentu. Díky tomu dokážeme na stránku pohlížet způsobem, jakým stránku prohlíží člověk. Nečte jej od začátku do konce, ale zaměřuje pozornost podle hierarchie nadpisů a dalších elementů podle jejich vizuálního významu v rámci celé stránky.

K úloze extrakce informace z logického modelu dokumentu lze tedy také přistoupit způsobem, jakým informace vyhledává člověk. Ten většinou přibližně zná formu, kterou hledaná informace má a prochází dokument po oblastech, dokud odpovídající úsek nenalezne. Pokud máme tento úkol vykonat automaticky, lze jej převést na vyhledání určitého podstromu v logickém modelu dokumentu. Při prohledávání používá [7] regulární výrazy pro navigaci v hierarchii stromu a algoritmus pro porovnávání stromů *pathfix*.

2.3 Document Object Model

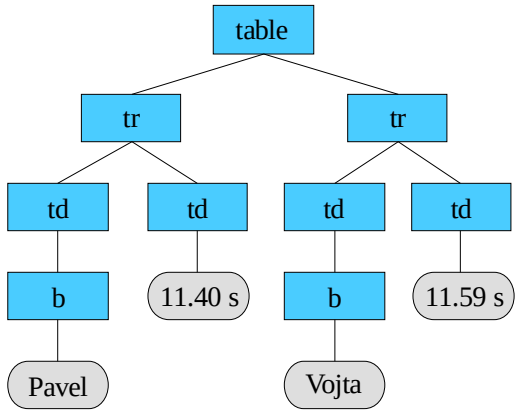
Abychom mohli snadno přistupovat k prvkům HTML dokumentu a efektivně s nimi pracovat, je potřeba mít možnost tyto prvky na stránce jednoduše a jednoznačně identifikovat. V počátcích rozvoje WWW měl každý webový prohlížeč své vlastní JavaScriptové rozhraní pro manipulaci s HTML elementy, což však znamenalo jejich vzájemnou nekompatibilitu. Dynamická práce s dokumentem byla obtížná a tvůrci webových stránek museli ladit skripty pro každý prohlížeč zvlášť.

Konsorcium W3C, které se zabývá tvorbu webových standardů, proto vytvořilo Document Object Model (DOM), což je v podstatě pouze API (rozhraní) pro práci s dokumentem, které musí každý program podporující tento standard obsahovat. Celý DOM standard je obecný, nezávislý na platformě i jazyku a lze jej použít pro správu HTML i XML dokumentů, grafiky SVG a dalších technologií zakládajících se na XML.

Na správně formátovaný HTML (resp. XML) dokument lze skrz DOM pohlížet jako na stromovou strukturu. Elementy, texty, komentáře a další prvky HTML jazyka tvoří v DOM objekty, odvozené od základního objektu *Node*. Každý z nich má své specifické rozhraní. Každý dokument obsahuje žádný nebo jeden uzel <doctype>, jeden kořenový uzel (u HTML dokumentu <html>) a libovolný počet komentářů, či procesních instrukcí. Některé uzly mohou mít pod sebou potomky, či další uzly specializovaných typů. Některé prvky jsou koncové a nemohou pod sebou mít nic. Díky stromové struktuře můžeme jednoduše rozlišit nadřazené, podřízené, nebo rovnocenné elementy.

Historie DOM je poměrné krátká. W3C vytvořilo první pracovní návrh DOM v polovině 90. let 20. století pod názvem DOM Level 0. Přestože se nejednalo o specifikaci, byl tento model zahrnut do standardu HTML 4. V roce 1998 byl uvolněn DOM Level 1, o dva roky později DOM Level 2. Ten již počítal s podporou kaskádových stylů a transformacemi dokumentu. Specifikace DOM Level 3, která je zatím poslední verzí, byla vydána v roce 2004.

```
<table>
  <tr>
    <td><b>Pavel</b></td>
    <td>11.40 s</td>
  </tr>
  <tr>
    <td><b>Vojta</b></td>
    <td>11.59 s</td>
  </tr>
</table>
```



Obrázek č. 5 - Fragment HTML kódu a jeho DOM reprezentace

2.3.1 DOM v PHP

DOM API implementuje i jazyk PHP a od verze 5.0 není k používání tohoto rozhraní potřeba žádné rozšíření, protože je přímo součástí jazyka. Funkce tohoto rozhraní lze použít pro generování, parsování XML a HTML dokumentů a manipulaci s jejich elementy, což je pro implementaci extrakčního systému velmi vhodné. Jde zejména o funkce pro zpracování HTML souborů, které nemusí být správně strukturovaným dokumentem. DOM automaticky doplní resp. vynechá značky tak, aby byl dokument v souladu s (X)HTML resp. XML normou. Taková data poté lze použít jako vstup XSLT procesoru, nebo nad nimi použít Xpath dotazy.

2.4 Extrakce dat z popisu zboží

Při volbě konkrétní metody, která bude extrahovat data z popisu zboží, je nutné brát v úvahu několik hledisek:

- **Spolehlivost extrakce** - algoritmus by měl dávat kvalitní výsledky minimálně pro takový formát dat, který se ve vstupních dokumentech vyskytuje nejčastěji.

- **Implementační složitost** - zvolené řešení by nemělo vést k příliš složitému a rozsáhlému kódu, který by později mohl způsobit problémy s udržovatelností aplikace. Zároveň by řešení mělo být dostatečně robustní s možností pozdějších rozšíření.

- **Požadavky na odbornost uživatele** - vytváření i použití extrakčních pravidel musí být pro uživatele snadno pochopitelné a nemělo by vyžadovat žádné zvláštní odborné znalosti. Obsluha systému by měla být intuitivní.

- **Rychlost definice pravidel** - proces definice a úprav extrakčních pravidel nesmí být časově náročný, jinak bude používání systému neefektivní.

Po bližším prostudování problematiky extrakce dat z webových dokumentů přichází v úvahu několik variant, jak lze k extrakci přistupovat. Níže jsou uvedeny dva přístupy, které byly při návrhu systému zvažovány.

2.4.1 Hledání cest ve stromu

Základní myšlenkou tohoto přístupu je využití znalosti o umístění parametru v rámci DOM webové stránky. Hledaná hodnota parametru je v rámci DOM stránky dána pozicí počátečního uzlu (s názvem parametru) a jednoznačnou cestou ve stromu uzlů. Tato cesta v dokumentu tvoří podstrom ve tvaru obráceného písmene ‘V‘, jehož kořen je nejbližší společný předek elementů s názvem, resp. hodnotou parametru. Počáteční element cesty obsahuje řetězec identifikující název parametru, cílový element text s hodnotu parametru. Přesný formát hodnoty může být dále extrahován pomocí dodatečných informací, jakými jsou např. pozice v textu, počet slov před a za hodnotou.

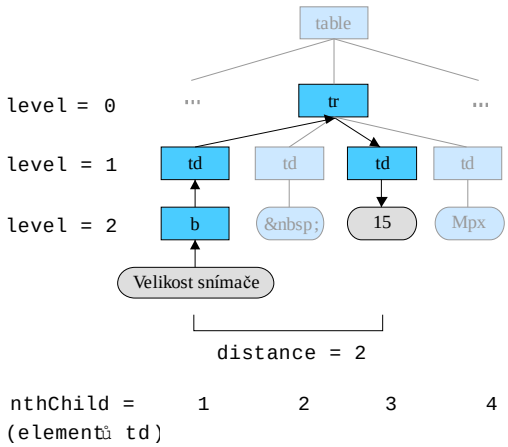
```
<table>

  ...

  <tr>
    <td><b>Velikost snímače</b></td>
    <td> </td>
    <td>15</td>
    <td>Mpx</td>
  </tr>

  ...

</table>
```



Obrázek č. 6 - Ukázka průchod DOM stromem pro tabulková data

Z principu hledání cest ve stromu vyplývá, že metoda není vhodná pro takové formáty dat, u kterých je počáteční i cílový element totožný (název i hodnota parametru jsou v jedné textové oblasti). Strom je v tomto případě redukován na kořenový uzel, takže algoritmus musí procházet velké množství nerelevantních elementů. Dobré výsledky lze naopak předpokládat pro data uložená ve strukturované formě, tedy např. pro tabulky (jednoduché i vnořené), seznamy a další složené struktury (data generovaná do elementů div).

2.4.2 Syntaktický analyzátor

Zcela odlišným přístupem k extrakci dat je vytvoření syntaktického analyzátoru jazyka, který by v popisu zboží hledal řetězce vyhovující určité šabloně. Z testovací množiny dat podrobně rozebrané v kapitole 3 vyplynulo, že se na stránkách každý parametr objevuje v několika mírně odlišných zápisech. Pro každý parametr by pak bylo nutné vytvořit sadu šablon (gramatiku), které by daný parametr přijímaly.

Identifikace jazykových konstrukcí hledaných parametrů by probíhala v textovém popisu produktu. Ten by mohl být zadán přímo v administračním rozhraní, případně by byl označen uživatelem na načtené stránce. Systém by provedl syntaktickou analýzu textu a zkusil by identifikovat ty lexémy, které tvoří název a hodnotu parametru. Součástí systému by byl index obsahující synonymní názvy pro jednotlivé parametry.
Šablona pro parametr rozměry (resp. gramatika přijímající tento parametr) by pak mohla vypadat následovně:

<parametr> → <název> <jednotka>? | <název> <hodnota> <jednotka>? <upřesnění>?
<název> → <řetězec>
<jednotka> → <párový_oddělovač_start><jednotka2><párový_oddělovač_konec> | <jednotka2>
<párový_oddělovač_start> → „ | ' | { | (| [
<párový_oddělovač_konec> → “ | ' | } |) |]
<jednotka2> → mm | cm (hodnoty jednotek načítány z databáze)
<hodnota> → <číslo> | neuvedeno | ε
<upřesnění> → (Max) | (Min)

Výška: neuvedeno
Šířka [mm] 430
Width (max) 101.6 mm (4.010 inches)
Height 1.028 Inches (Max)
Height 0.37" (9.5mm)
Šířka (mm) 88,9

Výše uvedený přístup však nebyl pro řešení diplomové práce shledán vhodným a to ze dvou hlavních důvodů:

- **Šablony** - pro každý parametr je třeba definovat velký počet šablon. Některé formáty parametrů mají mnoho podobných variant, což vede k nedeterministickým šablonám. Uživatel by měl mít možnost vytvářet nové šablony, aby byl systém schopen pracovat s novými formáty parametrů. Definice šablony v tomto případě však není triviální, což by bylo pro uživatele zbytečnou zátěží.

- **Rychlost** - protože lze některé parametry definovat pouze nedeterministicky, je třeba zavést deterministickou simulaci, tj. zásobníkový automat s návraty. Při velkém počtu šablon, které musí automat zpracovat, existuje reálné riziko zpomalení celého procesu natolik, že se stane neefektivním.

3 Zdroje vstupních dat

Abychom mohli navrhnout postup pro efektivní extrakci dat o výrobcích a jejich parametrech, je nutné provést několik kroků:

- **Vytvořit soubor zdrojů dat.** Systém není navrhován pro specifickou doménu a je tedy vhodné nalézt zdroje dat takové, které pokryjí širokou oblast výrobků. Přesto je při výběru zdrojů dat možné zohlednit takovou oblast, u které je předpoklad, že bude využívána nejčastěji.
- **Klasifikovat zdroje.** Provést rozdělení zdrojů podle charakteru a vlastností poskytovaných dat. Porovnat výhody a nevýhody jednotlivých tříd zdrojů a doporučit takové, které jsou pro extrakci nejvýhodnější.
- **Prostudovat formu prezentovaných dat.** V dostupné množině zdrojů identifikovat typické formy prezentace dat. Zaměřit se také na jejich varianty a modifikace. Vybrat takovou formu dat, která bude pro účely extrakce nejlépe použitelná a zároveň dostatečně frekventovaná.

Mezi nejčastější prodávané zboží v internetových obchodech patří stabilně elektronika (fotoaparáty, mobily, hardware), bílé zboží (chladničky, pračky) a knihy. Právě na tyto oblasti byl brán zřetel při vytváření souboru zdrojů dat, který lze nalézt v příloze č.1. V souboru jsou zdroje zastoupeny odkazem na stránku s konkrétním výrobkem a daty určenými k extrakci. Jako zdroj dat však chápeme množinu všech stránek s výrobky na daném serveru. Zároveň je žádoucí, aby byl zdroj dat zastoupen pouze jednou stránkou. V opačném případě by byly poměry zastoupení formátů dat zkreslené.

U sestavené množiny typických zdrojů dat byla provedena analýza a následně klasifikace dat.

3.1 Klasifikace zdrojů dat

Datové zdroje můžeme rozdělit podle původu obsažených informací na *primární*, které poskytují nová data, a *sekundární*, které informace pouze přejímají z primárních zdrojů.

3.1.1 Primární zdroje

Mezi primární zdroje řadíme ty, které jsou poskytovateli informací pro ostatní zdroje. Jedná se většinou o webové stránky výrobců, nakladatelství apod. Skutečnost, že se může současně jednat i o internetový obchod, který je typickým zástupcem sekundárních zdrojů, nehraje roli. Důležité je, zda jsou data publikovaná na zdroji původní.

3.1.1.1 Stránky výrobců

Výhody

- **Nejspolehlivější zdroj dat.** Stránky výrobců sice také mohou obsahovat chyby, protože specifikace se mezi výrobky často kopírují, ale přesto je korektnost dat jedna z nejvyšších.

Nevýhody

- **Variabilita dat.** Parametry jsou v rámci různých zdrojů zcela jinak pojmenované, jsou zapsány v jiných jednotkách. Někdy dokonce stejně pojmenované parametry nevyjadřují stejnou vlastnost, jako např. u rozměrů DVD přehrávačů (někteří výrobci rozměry uvádí bez výstupků a jiní včetně výstupků).
- **Nevyrovnaná kvalita.** Častým jevem je velký podíl balastních informací na stránkách produktu. Suchý výčet parametrů není z hlediska marketingu vhodnou propagací produktu a proto se na stránkách objevují popisné texty s množstvím hodnot a zkratek, které však nelze efektivně strojově zpracovat.
- **Formát dat.** Ne vždy jsou informace o zboží dostupné ve formátu HTML a pokud máme k dispozici pouze PDF dokument, nemůžeme jeho zdrojový kód tak jednoduše zpracovat. Další překážkou je, pokud výrobci vydávají série výrobků a uzpůsobí tomu i formu publikace dat. Jedna stránka pak obsahuje agregované informace o několika výrobcích, případně pouze rozdílové informace mezi výrobky v rámci série.

3.1.2 Sekundární zdroje

Mezi sekundární zdroje dat řadíme takové stránky, které obsahují data zpracovaná z primárních zdrojů. Je přirozené, že tato třída je mnohem obsáhlejší, protože obchodů prodávajících zboží od jednoho výrobce je obvykle více. Navíc, výrobci sami se často snaží, aby se informace o jejich produktech co nejvíce rozšířily.

3.1.2.1 Internetové obchody

Výhody

- **Konzistence dat v rámci zdroje.** Data již někdo před námi zpracoval, odstranil nepodstatné informace a extrahoval důležité parametry. Ty bývají v rámci zdroje stejně pojmenované.

Nevýhody

- **Správnost dat.** I přes to, že korektnost údajů u internetových obchodů je většinou na dobré úrovni, u neověřeného zdroje si nemůžeme být správností údajů jisti a je vhodné porovnání s daty z primárního zdroje.

3.1.2.2 Recenze produktů

Výhody

- **Správnost dat.** Na rozdíl od internetových obchodů, kde jsou výrobky vkládány po desítkách a stovkách, je při recenzi jednomu výrobku věnováno mnohem více času. Recenzent je většinou znalý oboru a produktu se podrobně věnuje, takže si i leccos ověří. Proto lze spolehlivost údajů v recenzích srovnat s primárními zdroji.

Nevýhody

- **Nedostatek kvalitních zdrojů**. Právě protože jsou recenze časově náročnou činností, je počet kvalitních a rozsáhlých zdrojů omezen.

3.2 Formát dat

Ať už se jedná o primární nebo sekundární zdroje, data jsou v nich publikována v několika nejčastějších formách.

3.2.1 Tabulka

Technické parametry uložené ve struktuře tabulky jsou formou dat, která je vhodná pro jejich analýzu i získávání. Tabulky jsou určeny pro přehledné shrnutí funkcí, které produkt poskytuje a obsahem buněk jsou často atomické hodnoty. Hustota využitelné informace je u tabulek vysoká i proto, že není obvyklé vkládat do tabulek souvislé texty.

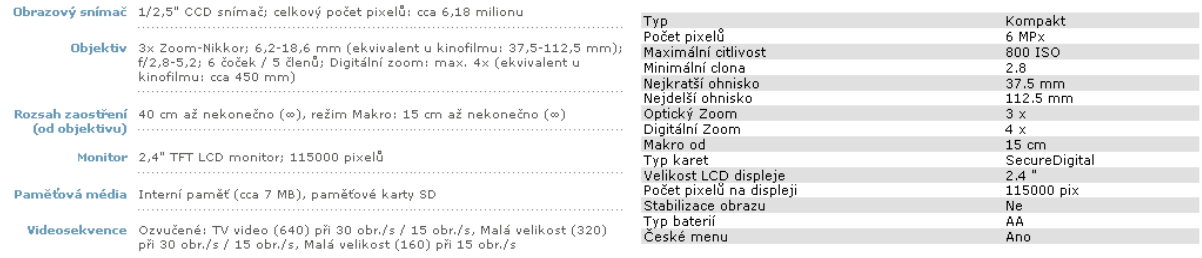
Nejčastější rozvržení tabulky je struktura 2 sloupců, kdy první obsahuje názvy parametrů (resp. skupiny parametrů), druhý jejich hodnoty. Někdy jsou skupiny logicky svázaných parametrů odděleny nadpisem ve zvýrazněném řádku, nebo naopak prázdným řádkem. Horizontální rozvržení tabulky se nevyskytuje, protože by obsahovala velké množství sloupců a taková struktura není pro publikaci na webových stránkách vhodná.

U primárních zdrojů je formátování tabulek více variabilní. Tabulky nemají často jednoduchou strukturu parametr/hodnota, mají více úrovní, případně jsou do sebe zanořené. Ve větší míře obsahují informace v takové formě, kterou nelze přímo využít. Jako příklad lze uvést specifikace více produktů v jedné tabulce, rozdílové tabulky mezi jednotlivými produkty. Překážkou bývá často i samotný formát dat, kdy je dokument uložen ve formátu PDF. Zdrojový kód takového dokumentu nelze tak snadno analyzovat jako (X)HTML.

U sekundárních zdrojů mají tabulky parametrů v drtivé většině jednoduchou strukturu parametr/hodnota. Je to dáno tím, že tato data nejsou primárně určena pro prezentaci, ale jsou do databáze vložena za účelem porovnávání a vyhledávání produktů. Na to jsou parametry s atomickými hodnotami nejvhodnější, a to jak pro stroj, tak i pro člověka.

Výhodou tabulek u sekundárních zdrojů je také skutečnost, že jsou v nich vynechány nepodstatné, těžko klasifikovatelné údaje, případně takové údaje, které se nevyskytují u typu výrobku obecně a jsou specifické pouze pro daný produkt. Jejich vhodnost je tedy dána tím, že parametry byly

již předem vybrány a jsou v rámci celého datového zdroje konzistentní. To u primárních zdrojů sice platí také, ale tam máme k dispozici data pouze od jednoho výrobce. Níže je uveden příklad typické tabulky parametrů pro primární a sekundární zdroj.



Obrazový snímač	1/2,5" CCD snímač; celkový počet pixelů: cca 6,18 milionu
Objektiv	3x Zoom-Nikkor; 6,2-18,6 mm (ekvivalent u kinofilmu: 37,5-112,5 mm); f/2,8-5,2; 6 čoček / 5 členů; Digitální zoom: max. 4x (ekvivalent u kinofilmu: cca 450 mm)
Rozsah zaostření (od objektivu)	40 cm až nekonečno (∞), režim Makro: 15 cm až nekonečno (∞)
Monitor	2,4" TFT LCD monitor; 115000 pixelů
Paměťová média	Interní paměť (cca 7 MB), paměťové karty SD
Videosekvence	Ozvučené: TV video (640) při 30 obr./s / 15 obr./s, Malá velikost (320) při 30 obr./s / 15 obr./s, Malá velikost (160) při 15 obr./s

Typ	Kompakt
Počet pixelů	6 MPx
Maximální citlivost	800 ISO
Minimální clona	2.8
Nejkratší ohnisko	37.5 mm
Nejdelší ohnisko	112.5 mm
Optický Zoom	3 x
Digitální Zoom	4 x
Makro od	15 cm
Typ karet	SecureDigital
Velikost LCD displeje	2.4 "
Počet pixelů na displeji	115000 pix
Stabilizace obrazu	Ne
Typ baterií	AA
České menu	Ano

a) www.europe-nikon.com b) www.cybex.cz

Obrázek č. 7 - Srovnání typické tabulky parametrů u primárního (a) a sekundárního (b) zdroje

3.2.2 Seznam

Uspořádání dat do seznamu je po tabulce druhé nejčastěji používané formátování dat. Řádky jsou uvozeny odrážkou, která není textová a je součástí formátování dokumentu. Seznam s textovými odrážkami není sémanticky odlišitelný od souvislého textu a proto jej neuvažujeme. Číslované seznamy se pro seznamy parametrů nepoužívají, protože parametry nemají definováno pořadí.

Řádky v seznamu jsou nejčastěji ve tvaru “parametr oddělovač hodnota“, ale v některých případech je parametr s oddělovačem vynechán. Seznam pak obsahuje pouze seznam hodnot. Intuitivně sice dokážeme rozpoznat, které hodnoty co znamenají, pro automatické rozpoznávaní je však absence názvu parametru významná komplikace. Pokud odrážky obsahují větu s popisem parametru, charakter dat se více blíží souvislému textu než seznamu parametrů a tomu by mělo být přizpůsobeno i jeho zpracování.

3.2.3 Text s parametry oddělenými oddělovačem

Tento typ formátování parametrů lze chápat jako speciální případ seznamu. Místo odrážky na každém řádku je použit dostatečně specifický oddělovač jednotlivých parametrů. Tento druh zdrojových dat však není pro extrakci vhodný, protože nemáme zaručenu unikátnost oddělovače v rámci textu.

Při pohledu na používané oddělovače je zřejmé, že při jejich výběru není na zdrojový text brán zřetel. Mezi nejčastější patří hvězdička, pomlčka, středník a čárka. Tyto znaky jsou tak časté, že pro oddělení jednotlivých parametrů nemohou být spolehlivě použity a na takováto data můžeme tedy pohlížet spíše jako na souvislý text.

3.2.4 Souvislý text

Takovýto zdroj je pro získávání dat nejméně vhodný. Textový popis produktu má malou hustotu informací, které lze pro získávání parametrů využít. Pokud máme k dispozici text výrobce určený pro marketingové účely, je pravděpodobné, že z něj nebudeme moci získat parametry žádné. V takovémto textu jsou totiž technické podrobnosti často nežádoucí.

4 Analýza zadání a návrh systému

Na základě zadání a informací z předchozích kapitol byl sestaven seznam požadavků kladených na navrhovaný systém pro extrakci dat:

- Systém by měl jít jednoduše integrovat do již běžících eshopů.
- Intuitivní a nenáročný proces definování pravidel pro extrakci parametru v dané třídě dokumentů.
- Systém by měl co nejvíce redukovat potřebu definovat obdobná pravidla vícekrát.
- Systém automaticky rozpozná do jaké třídy dokumentů webová stránka patří a aplikuje příslušná extrakční pravidla.
- Pokud proces extrakce selže, měl by systém umožnit pro parametr definovat nové pravidlo, případně jej upřesnit.

Z prvního požadavku vyplývá, že by měl být systém schopen pracovat jak samostatně, tak ve spojení s již existujícím systémem elektronického obchodu. Protože jsem doposud nasazené obchody vyvíjel v prostředí PHP/MySQL, rozhodl jsem se systém implementovat na stejné platformě. Pro dynamické prvky na straně klienta bude použit Javascriptu, případně dle potřeby technologie AJAX (Asynchronous JavaScript and XML [10]).

Z analýzy zdrojů dat vyplynulo, že data vstupující do procesu extrakce budou mít nejčastěji tvar tabulky a seznamu, případně textového seznamu s oddělovači. Pro tabulková data resp. seznamy dosahují wrappery vysoké spolehlivosti. Pro rozpoznání parametru ze seznamu s textovými odrážkami můžeme použít regulární výrazy.

Nalezení a rozpoznání hodnoty parametru na stránce je založeno na vyhledávání charakteristické stromové struktury HTML elementů, která obsahuje uživatelem označený řetězec, nejčastěji název parametru. Takový způsob je použitelný pro tabulková data, nehodí se však pro data, u nichž je více parametrů v rámci jednoho HTML elementu, např. pro seznam s textovými odrážkami. Zde bude nutné, pro nalezení cesty od názvu parametru k jeho hodnotě, aplikovat vhodně regulární výrazy, případně nalézt jinou vhodnou metodu.

4.1 Změny v návrhu

Proti návrhu systému v semestrálním projektu bylo provedeno několik úprav. V tabulce Rule byly odstraněny sloupce *outset* a *location*. Tyto údaje je vhodnější umístit přímo jako součást extrakčního

[10] AJAX - http://cs.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML

pravidla, protože jsou při jeho aplikaci vždy použity. Navíc by ukládání údajů do více sloupců nebylo efektivní. Hodnoty jsou proto uloženy spolu s informacemi o názvu a hodnotě parametru pod kvalifikovaným jménem *selText*.

Druhou změnou v návrhu databáze je zavedení cizího klíče *id_category* v tabulce *Parameter*. Důvodem je praktická realizace vyhledávání v této tabulce, kde probíhá selekce záznamů podle hodnot parametrů a kategorie. Absence sloupce *id_category* by vedla ke spojování tabulek *Parameter* a *Product*, což by při počtu dotazů a velikosti tabulek bylo příliš pomalé.

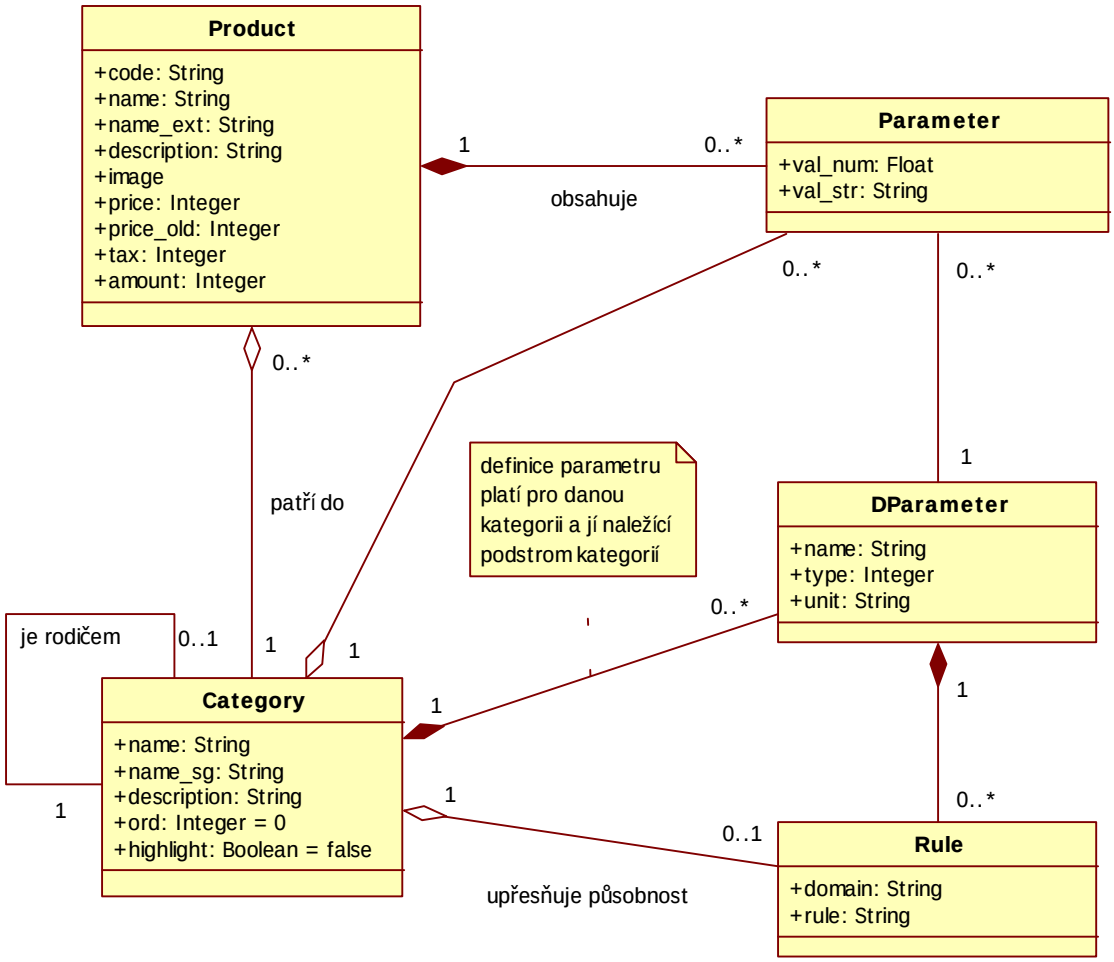
Poslední změnou oproti návrhu systému v semestrálním projektu bylo vynechání DOM API v PHP. Při praktickém testování třídy *locBrowser* bylo zjištěno, že se mnoha stránkám rozpadnul design. Toto chování nastalo vždy při transformaci zdrojového kódu z normy HTML do XHTML, kterou je pro práci s DOM nutné provést. Optimalizace CSS stylů pro normu HTML (např. kvůli rozdílné interpretaci box-modelu apod.) způsobila v XHTML deformaci vzhledu stránek a znemožnila je v této podobě komfortně prohlížet.

Úprava zdrojového kódu stránek proto probíhá přes regulární výrazy a to pouze pro ty části, které jsou pro změnu kontextu stránky potřeba (zejména odkazy). Možnost zpracování přes DOM API však byla v programu zachována a to s ohledem na budoucí rozvoj systému.

4.2 Návrh systému

Na základě neformálních požadavků byla vytvořena sada diagramů. Logický datový model je znázorněn v ER diagramu, dynamické chování systému popisuje diagram použití.

4.2.1 ER diagram



Obrázek č. 8 - ER diagram navrhovaného systému

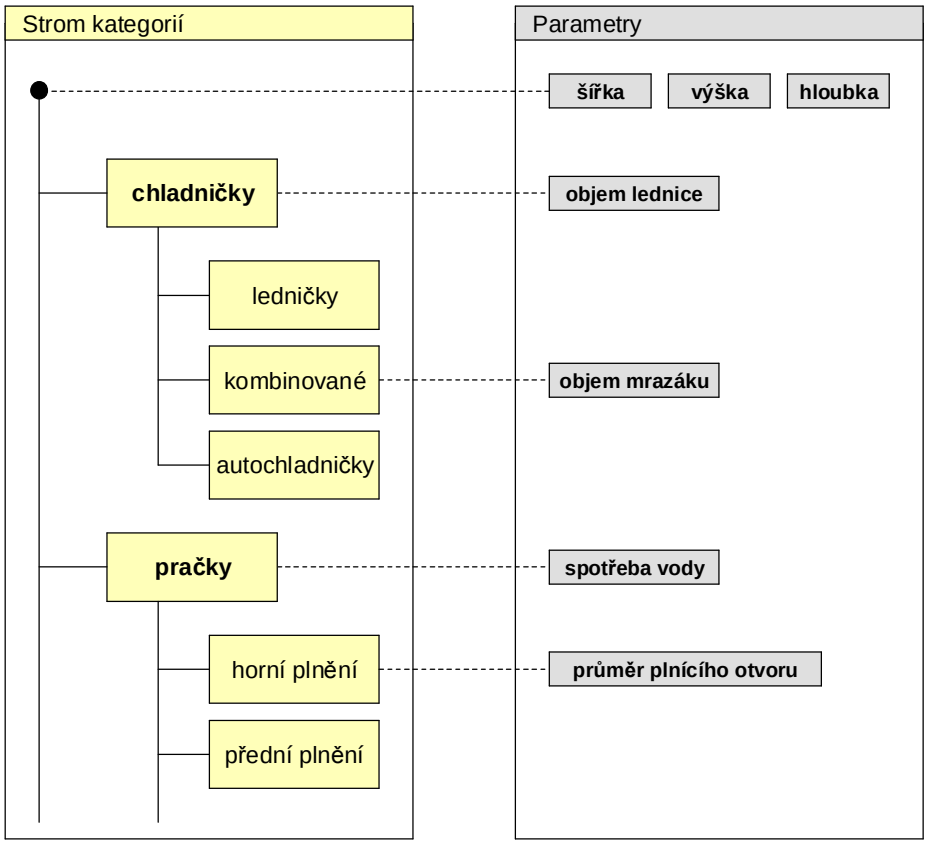
ER diagram modeluje strukturu databáze. Obsahuje pouze ty tabulky, které bezprostředně souvisí se systémem pro extrakci dat. Další tabulky, určené pro funkce elektronického obchodu, nejsou relevantní a na diagramu tedy nejsou zobrazeny.

Z diagramu lze vyčíst, že systém pracuje s katalogem produktů, které jsou řazeny do kategorií ve stromové struktuře. Produkty v katalogu obsahují v rámci všech kategorií společné vlastnosti jako jsou kód, název, rozšířený název, popis, cena, DPH, fotografie a počet kusů na skladě.

Ke každé kategorii lze přiřadit libovolný počet parametrů. Definice parametru zahrnuje jeho jméno, typ hodnoty a volitelně jednotku zobrazovanou za hodnotou. Působnost každého parametru je v aktuální kategorii a jí náležícím podstromu kategorií. Toto rozšíření je pro použitelnost systému vhodné, protože odpadne mnohonásobná definice stále stejných parametrů. Také pokud bude do systému vložena nová kategorie, nebude jí potřeba parametry přiřazovat, protože je zdědí z nadřazených kategorií (viz Obrázek č. 9).

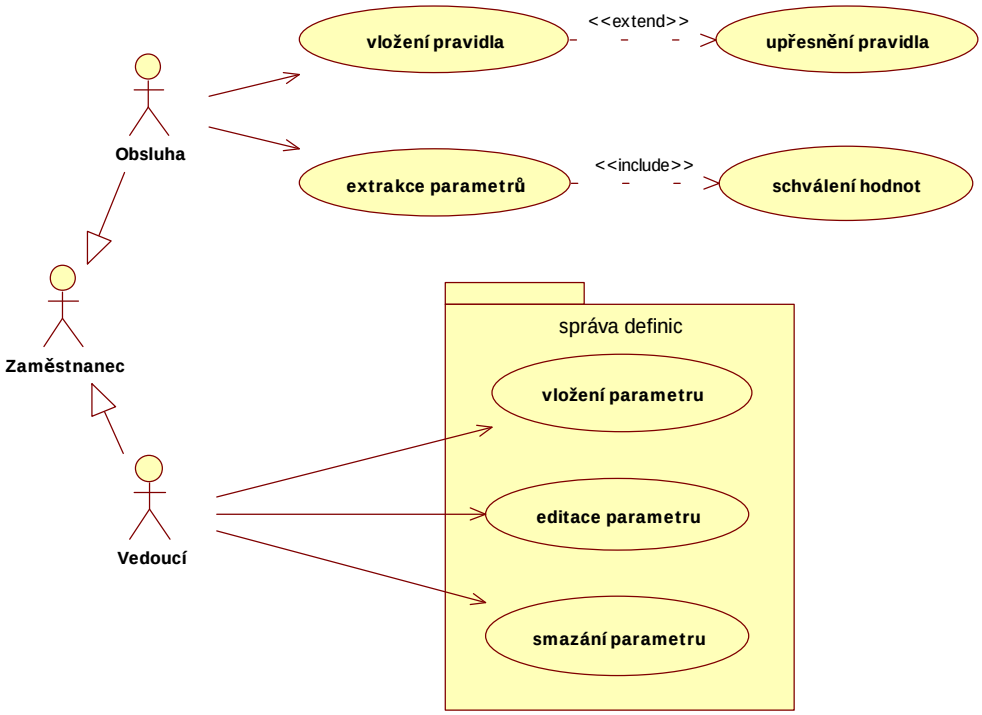
Pro každý parametr v systému existuje libovolné množství extrakčních pravidel, z nichž každé je určeno pro jinou třídu dokumentů. Protože je však působnost pravidla dána přes více kategorií a nemáme zaručeno, že na straně zdroje dat mají stránky z různých kategorií stejné formátování (tj. že patří do stejné třídy dokumentů), je třeba mít možnost upřesnit působnost pravidla pro konkrétní kategorii.

Hodnoty extrahovaných parametrů jsou uloženy v tabulce *Parameter* a mohou mít řetězcový, nebo číselný charakter. Nad touto tabulkou bude v běžícím elektronickém obchodě probíhat velké množství dotazů, proto je třeba vhodně zvolit indexy.



Obrázek č. 9 - Působnost parametrů je rozšířena o podstrom příslušné kategorie

4.2.2 Diagram případů použití



Obrázek č. 10 - Diagram případů použití

Protože bude navrhovaný informační systém součástí systému elektronického obchodu, budeme v něm rozlišovat 2 aktéry: vedoucího a obsluhu. Toto rozdělení je nutné proto, aby bylo možné přidělit oprávnění pro správu definic parametrů pouze vedoucímu obchodu. Vkládání, editace a mazání definic parametrů je citlivou činností, kterou by měl provádět pouze prověřený pracovník. Obsluha má právo provádět extrakci parametrů, kontrolovat a potvrzovat nalezené hodnoty. V případě doposud nedefinovaných pravidel pro aktuální třídu dokumentů může obsluha vkládat pravidla nová, případně upřesnit jejich působnost na určitou kategorii obchodu.

4.2.3 Návrh tříd

Aby byla maximalizována možnost znovupoužití vytvořených komponent, je nutno jejich vazby soustředit do rozhraní. Je také žádoucí, aby každá třída poskytovala množinu metod, které mají logickou souvislost. Na základě analýzy zadání bylo navrženo těchto 5 tříd:

4.2.3.1 locBrowser

Třída *locBrowser* je jako jediná určena pro serverovou část systému. Jejím hlavním úkolem je načítání (X)HTML dokumentů a jejich transformace do takové podoby, aby je bylo možné správně interpretovat z místního serveru.

4.2.3.2 extractorEngine

Třída *extractorEngine* je hlavním třídou klientské části programu. Zajišťuje vytvoření ostatních objektů, nastavení jejich parametrů, vygenerování GUI programu do určeného kontejneru a obsluhu událostí iniciovaných uživatelem. Třída také obsahuje logiku pro synchronizaci množiny extrakčních pravidel (ukládání na server, načítání do aplikace).

4.2.3.3 ruleProcessorEngine

Třída *ruleProcessor* zapouzdřuje veškeré funkce potřebné pro vytváření a modifikaci extrakčních pravidel. Obsahuje metody pro použití pravidel na aktuální dokument, práci s označeným textem, výpočet potřebných metrik a také metody pro identifikaci a převod jednotek.

4.2.3.4 domEngine

Třída *domEngine* se snaží odstínit nekompatibilitu jednotlivých prohlížečů při práci s DOM stránky. Zavádí kratší aliasy pro často používané funkce jako jsou např. *getElementById (gI)*, *getElementsByTagName (tN)*, nebo *createTextNode (cT)*, protože by jejich používání zdrojový kód zbytečně protahovalo. Třída také programátorovi usnadňuje detekovat uživatelem používaný prohlížeč.

4.2.3.5 debugEngine

Účelem třídy *debugEngine* je poskytnout vývojáři možnost vypisovat ladící informace i za běhu skriptu. V konstruktoru třída vytvoří na aktuální stránce kontejner, do kterého pak lze vypisovat ladící informace, případně jakékoliv další zprávy. Pokud nastane chyba, lze takto zajistit přehled o aktuálním obsahu proměnných, místě i kontextu chyby apod.

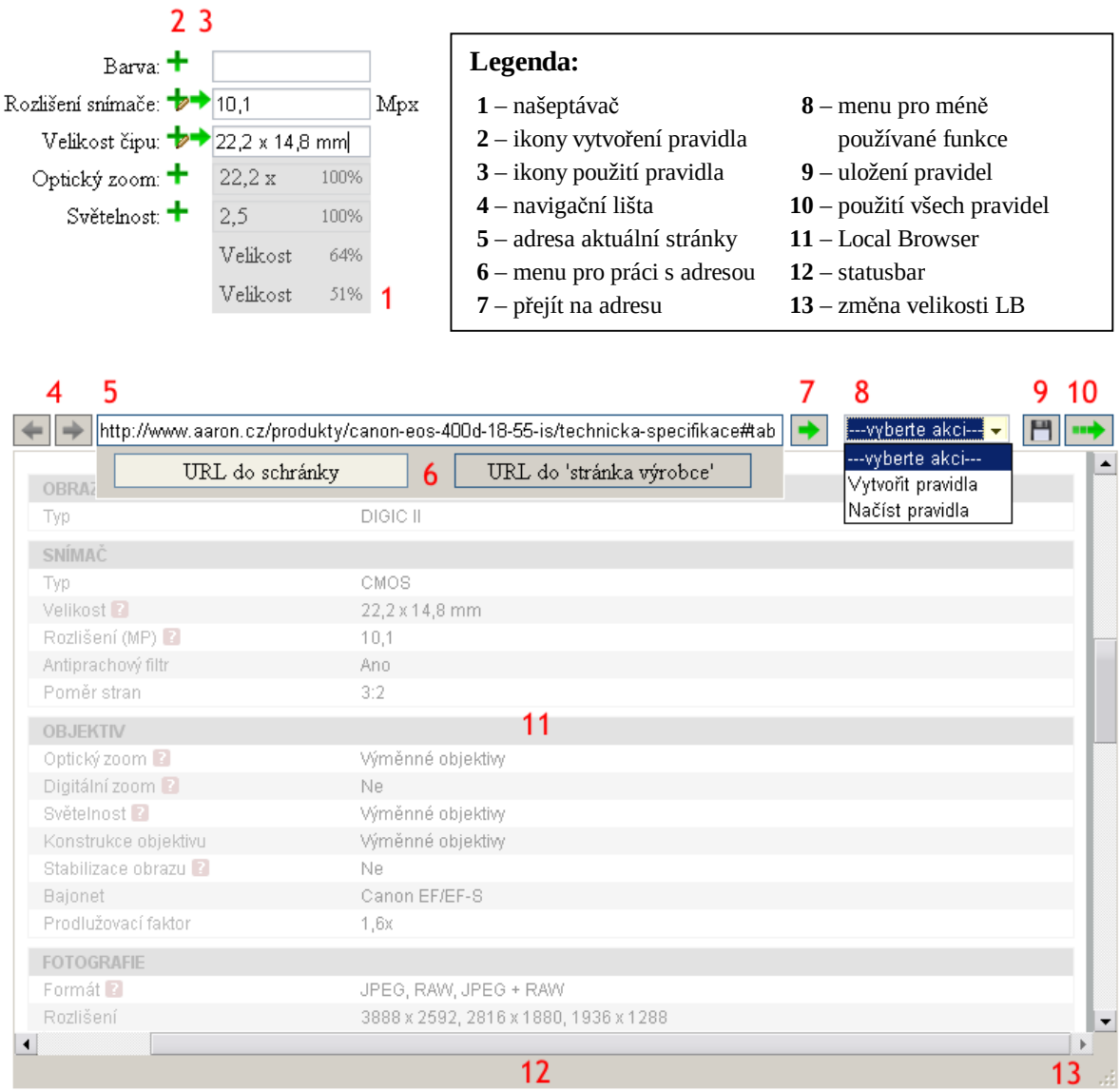
Přestože je nástroj pro debugging v JavaScriptu nutností pro vývoj jakéhokoli netriviálního programu, existuje uspokojivé řešení pouze pro prohlížeč Mozilla Firefox[11]. Konkurenční Microsoft Script Debugger určený pro Internet Explorer poskytuje pouze základní funkce, prohlížeč Opera má podporu pro ladění skriptů dokonce ještě horší.

4.2.4 Návrh GUI

Při návrhu grafického rozhraní bylo nutno vzít v úvahu, že jeho uživatelé nebudou vždy počítačově zdatní, proto by mělo být ovládání co nejvíce intuitivní. Při návrhu jsem vycházel z předpokladu, že se uživatel bude dobře orientovat v prostředí, které již zná. Rozhraní webového prohlížeče toto splňuje, proto z něj vychází navigační lišta i rozmístění tlačítek. Pro rychlou a efektivní práci je také důležité mít často používané funkce ihned dostupné (tlačítka), méně používané na 1-2 kliknutí (rozbalovací nabídka akcí).

[11] Doplněk Firebug - http://www.getfirebug.com

V některých případech není při extrakci nalezena pouze jedna hodnota, ale rovnou několik. Může se jednat jak o nerelevantní data, tak o deriváty nalezených hodnot (např. v jiné jednotce). Uživatel by měl mít možnost tyto alternativní hodnoty (a jejich relevanci) zobrazit a vybírat z nich. Toto je řešeno tzv. našeptávačem, který se zobrazí při kliknutí na pole s hodnotou parametru.



Obrázek č. 11 - Návrh uživatelského rozhraní aplikace

5 Implementace

Jak již z návrhu vyplývá, systém je rozdělen na klientskou a serverovou část. Klientská část je tvořena JavaScriptovou aplikací, která je vykonávána v prohlížeči uživatele. Umožňuje vytvářet, upravovat i aplikovat pravidla na aktuálně načtený dokument.

Protože je podpora skriptů v současných prohlížečích na různé úrovni, byl vývoj omezen pouze pro některé z nich. Vývoj např. pro Safari, Konqueror či Camini by byl zdlouhavý (nutnost ladění ve virtuálním systému), zároveň by tuto podporu využila pouze minimální skupina uživatelů. Odladění klientské aplikace tedy bylo provedeno pro trojici nejpoužívanějších prohlížečů: Internet Explorer 7, Firefox 2 a Opera 9.

Část systému umístěná na serveru se skládá z informačního systému e-shopu, do kterého je extrakce parametrů včleněná, služby pro načítání a ukládání extrakčních pravidel a služby Local Browser, která je určena pro přesun kontextu webové stránky na aktuální server.

5.1 Serverová část

5.1.1 E-shop

IS elektronického obchodu je rozsáhlý systém, který poskytuje základní funkce pro správu i další doplňkové funkce, které se při reálném nasazení osvědčily (skladový systém, generování faktur). Tyto funkce však nejsou z hlediska tématu diplomové práce podstatné. Systém parametrů navržený v kapitole 4.2 naopak s vyvíjenou aplikací úzce souvisí a bude proto rozebrán níže.
Definování i vkládání parametrů a jejich hodnot k výrobkům probíhá v administračním rozhraní, vyhledávání pak ve veřejné části stránek. Zajímavým problémem se ukázalo prohledávání výrobků dle parametrů. V úvahu přichází několik variant:

1. Vyhledávání v rámci všech parametrů na samostatné stránce. Skrývání nerelevantních parametrů probíhá při výběru prohledávané kategorie. Informace o parametrech jsou uloženy přímo na stránce v JavaScript objektu.
2. Výše uvedené řešení, avšak s načítáním údajů o parametrech přes AJAX.
3. Formulář pro filtrování výrobků na každé stránce kategorie. Načítají se pouze parametry aktivní v rámci aktuální kategorie.

Řešení č.1 a 2 nejsou z hlediska přístupnosti vyhovující, protože jsou závislé na JavaScriptu. Pokud návštěvník JavaScript nepodporuje (kvůli bezpečnostním opatřením, nastavení prohlížeče,

nebo např. přistupuje z mobilního klienta), zobrazí se všechny parametry (i nerelevantní), nebo naopak žádné (v případě AJAX varianty).

Řešení č.3 kritéria přístupnosti splňuje, při praktické realizaci však zabírá formulář na stránce příliš mnoho místa. Situaci vyřeší jednoduchý JavaScript, který formulář při načtení stránky skryje a místo něj vloží tlačítko pro opětovné zobrazení. Přístupnost ani použitelnost řešení se tímto nesnižuje, protože formulář zůstane bez zapnutého skriptování stále viditelný.

5.1.2 Služba pro načítání a ukládání extrakčních pravidel

Jak bylo uvedeno v kapitole 4.2, působnost pravidla je daná pro určitou doménu (tj. třídou dokumentů). Protože je ale v databázi více kategorií výrobků, u kterých se formát pravidla může mírně lišit, je třeba pro některé kategorie pravidlo upřesnit. Pravidla se nejdříve ukládají ke kořenu stromu kategorií (jejich působnost je tedy maximální), v případě potřeby jsou upřesněna a uložena do více zanořené kategorie.

Při načítání je použit opačný postup, aby byly vráceny pouze nejrelevantnější pravidla. Maximální relevanci mají pravidla uložená přímo v kategorii, pro kterou jsou určena. Pokud takové pravidlo není nalezeno, hledá se ve směru ke kořenu stromu kategorií.

Výše uvedenou logiku je vhodné umístit mimo klientskou aplikaci a tak ji od výběru pravidel odstínit. V opačném případě by bylo jejich načítání a ukládání na klientovi příliš komplikované.

5.1.3 Local Browser

Jedná se o samostatný skript, který načítá (X)HTML dokumenty a transformuje je do podoby interpretovatelné z místního serveru. Adresy dokumentů k načtení získává z parametru předaného metodou GET. Téměř veškerou funkčnost skriptu zajišťuje třída *locBrowser*.

Principem transformace je nalezení odkazů s relativními cestami a jejích nahrazení absolutními. Dále je třeba ze zdrojového kódu odstranit JavaScript a to obsah tagů skript, sekcí CDATA i další kód navázaný na události přímo v (X)HTML kódu. Odstranění skriptů probíhá z důvodu ohrožení funkčnosti aplikace v případě, že by generoval chybu, výjimku apod.

Problematickým místem se ukázala konfigurace serveru, na kterém byla zapnuta PHP direktiva *safe_mode*. Toto nastavení je sice běžné u většiny webhostingů, znemožňuje však následovat přesměrování při načítání dokumentů přes knihovnu *curl* a znemožňuje tak načítat významnou část webů. V případě nutnosti přesměrování se uživateli zobrazí buďto prázdná stránka, případně HTTP hlavička B*ad Request*. O tomto problému se ví již delší dobu a v současné době se pracuje na podpoře přesměrování v rámci *safe_mode*, která se objeví v některé z nových verzí PHP.

Protože nelze na testovacím serveru direktivu *safe_mode* vypnout, je při procházení stránek přes Local Browser citelně snížen komfort. Do budoucna však lze očekávat eliminaci tohoto problému přechodem na novou verzi PHP.

5.2 Klientská část

Tuto část systému tvoří JavaScript aplikace, běžící v klientově prohlížeči. Při její implementaci bylo řešeno množství úloh a problémů, které souvisí s principem extrakce, její rychlostí a efektivností, ale i s kompatibilitou programu v jednotlivých prohlížečích. Následující podkapitola se proto zaměří pouze na ty části systému, které jsou vzhledem k tématu diplomové práce zajímavé. Popisovat podrobně veškeré funkce není nutné, protože je lze nalézt v projektové dokumentaci.

5.2.1 Algoritmus pro extrakci dat

Pro extrakci dat byla zvolena metoda hledající určitou cestu ve stromu dokumentu. Algoritmus pro hledání počátečního elementu používá princip wrapperu, další fáze algoritmu je variantou algoritmu pro porovnávání podstromů v rámci DOM. Následuje stručný popis algoritmu, který je poté detailně vysvětlen:

Vstupy:

- DOM prohledávané stránky
- extrakční pravidlo
- definice parametru

Výstupy:

- pole dvojic nalezených hodnot a jejich odhadnutá relevance

Algoritmus:

- **nalezení elementů s názvem parametru**
  - elementy se stejným jménem tagu
  - zúžení výběru na elementy obsahující hledaný text
- **nalezení elementů s hodnotou parametru**
  - průchod přes DOM cestou jednoznačně danou pravidlem
  - seřazení cílových elementů dle relevance
- **zpracování textových informací**
  - nalezení textu vyhovujícího podmínkám pravidla
  - identifikace a převod jednotek u numerických hodnot

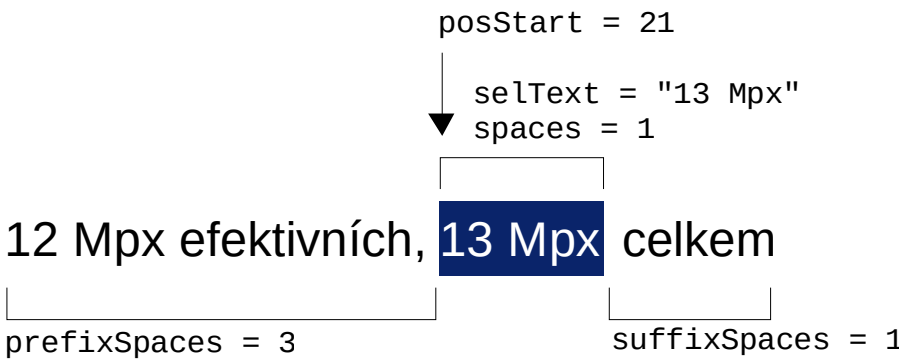
V prvním kroku se snažíme nalézt elementy obsahující název parametru. Nejprve jsou elementy v dokumentu filtrovány podle jména tagu, poté je výběr zúžen na elementy obsahující hledaný text.

Cílem druhého kroku je nalezení elementu s hodnotou parametru. Pro každý počáteční element je otestováno, zda existuje posloupnost elementů v DOM shodná s cestou v extrakčním pravidle. Pro každý průchod je počítána relevance cílového elementu k uložené cestě. Pokud je cesta zcela shodná, relevance zůstává na původní hodnotě 100%, v opačném případě se hodnota snižuje. Důvodem pro zavedení relevance je snížení závislosti na přesné struktuře HTML dokumentu. Extrakční pravidlo u nepatrně odlišné struktury cílový element také nalezne, i když s nižší relevancí. Tato situace nastane např. pokud se začnou názvy parametrů označovat tučným písmem oproti stavu při vytváření pravidel.

Nevýhodou přístupu, který toleruje odchylky při průchodu stromem, je nalezení většího počtu nerelevantních elementů. Pokud však v dokumentu správná cesta existuje, bude mít stále relevanci 100% a ostatní elementy nás tedy nemusí zajímat. Při hledání cílových elementů jsem se proto rozhodl umožnit průchod stromem, i přes to, že cesta není zcela totožná.

Ve třetím kroku se analyzuje text v nalezených cílových elementech. Program otestuje, zda se v něm nachází část odpovídající podmínkám pravidla a také definici parametru (u číselných parametrů hledá číslo, jednotku apod.). Tento krok je možné realizovat v mnoha variantách, s přihlédnutím na postup zadávání extrakčního pravidla (označení myší) byl zvolen způsob nevyžadující žádné dodatečné informace:

- Pokud je cílový element označen kliknutím, je jako cílový text brán celý vnitřní text.
- Pokud je v cílovém elementu vybrána část textu, zjistíme jeho pozici od začátku textu, od konce textu a jeho délku. Tyto informace jsou dále zpracovány.



Obrázek č. 12 - Ukázka zpracování označené oblasti v rámci elementu

Pro účely extrakčních pravidel, které mají popisovat pozici parametru v textu univerzálně, nelze brát pro metriku vzdálenosti počet znaků. Vhodnějším údajem je počet skupin alfanumerických, nebo nealfanumerických znaků (tj. počet slov, nebo počet skupin mezer). V implementaci systému je použita varianta s počtem nealfanumerických skupin znaků, kterou zajišťuje metoda

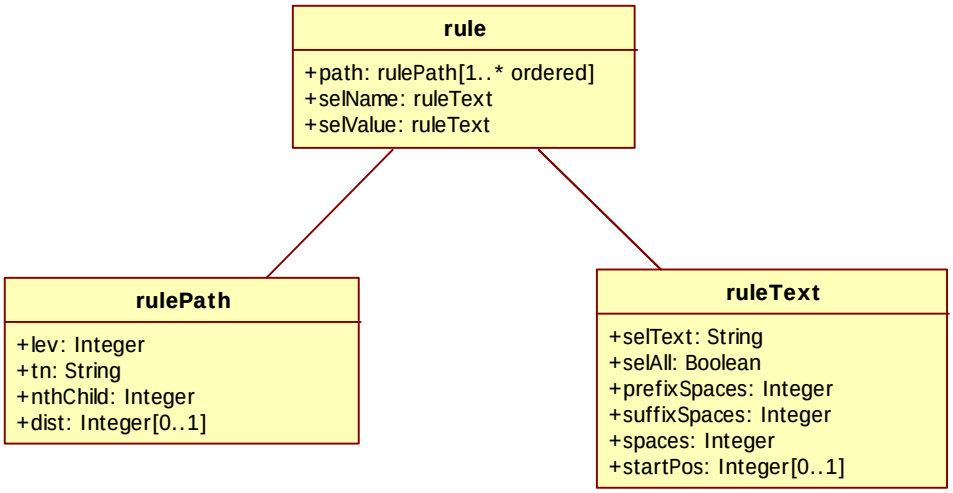
_nonAlphaNumericGroupCount. Po dokončení systému lze uvažovat o hledání metriky s lepšími vlastnostmi.

5.2.2 Datová struktura extrakčních pravidel

Návrh vhodné datové struktury pro extrakční pravidla měl při návrhu systému zcela zásadní roli. Při vytváření konkrétní reprezentace pravidla byly identifikovány požadavky, které musí extrakční pravidlo splňovat.

- Schopnost jednoznačně zachytit libovolnou cestu mezi 2 elementy dokumentu.
- Snadná interpretace a manipulace s touto cestou.
- Uložení informací o pozici hodnoty v rámci textu elementu.

Další vhodnou vlastností je datová nenáročnost pravidla a také jeho snadná pochopitelnost, která se uplatní při odlaďování systému. Po zvážení těchto požadavků byla navržena struktura, kterou popisuje následující diagram:



Obrázek č. 13 - Diagram datové struktury extrakčního pravidla

Pravidlo je složeno z části *path*, která popisuje cestu v dokumentu (realizováno kolekcí datových typů *rulePath*), a z částí *selName* a *selValue* (obě typu *ruleText*), které zahrnují informace o názvu a hodnotě parametru. Typ *rulePathNode* obsahuje informaci o umístění jednoho uzlu cesty (element v dokumentu), případně doplňkové informace o jeho pozici vzhledem k ostatním uzlům. Podrobnější popis jednotlivých atributů lze nalézt v příloze č.4.

5.2.3 Ukládání a načítání extrakčních pravidel

Jak bylo uvedeno v kapitole 2.2, platnost každého extrakčního pravidla je svázaná s určitou třídou dokumentů. Ta je dána množinou dokumentů, které jsou generované stejným nebo podobným způsobem. V našem případě jsou třídy identifikovány doménou 1. a 2. řádu, tedy řetězcem ve tvaru *domain.tld*.

Při procházení webu (ve vnořeném rámu Local Browser) uživatel postupně prochází přes různé domény a třída dokumentů se neustále mění. Aby byla extrakční pravidla pro aktuální doménu platná, je třeba provádět jejich synchronizaci (uložení nových pravidel pro starou doménu, načtení pro aktuální).

Pro zasílání i přijímání synchronizačních dat je použita technologie AJAX, avšak místo standardního XML se využívá vhodnějšího datového formátu JSON. Výhodou JSON je hlavně jednodušší manipulace s daty a úspornost tohoto formátu. Při experimentech velká redundance XML navýšila průměrnou velikost pravidla na 3Kb, oproti 1 Kb u JSON. Takový nárůst již mohl negativné ovlivnit rychlost synchronizace pravidel se serverem.

Pokud aplikace při události onload rámu Local Browser detekuje přechod na novou doménu, ověří, zda je třeba provést uložení změněných pravidel. Pro tyto účely obsahuje třída *extractorEngine* frontu *rulesToUpdate*, do které se ukládají nově vzniklá nebo upravená pravidla. Pokud uživatel chce uložení provést, je zaslán požadavek na server. Zároveň jsou odeslaná pravidla zaznamenána do pole pravidel odeslaných k uložení. Poté, co server vrátí odpověď, aplikace ověří, zda byla pravidla korektně uložena a poté je z fronty *rulesToUpdate* odstraní. V případě chyby je uživatel upozorněn a pravidla ve frontě zůstávají.

Každý požadavek na server musí obsahovat parametry *url* a *id_category* předané metodou GET, aby bylo možno jednoznačně identifikovat třídu dokumentů. Pravidla jsou předávána metodou POST (ukládání), nebo v obsahu odpovědi (načítání) a jsou zakódovaná v Base64.

5.3 Řešené implementační problémy

5.3.1 Cross-site scripting (XSS)

Technika Cross-site scripting je založena na injektování kódu do webové aplikace tam, kde to autor nepředpokládal. Vložení vlastního JS kódu může mít za následek poškození funkčnosti stránek, vylákání citlivých informací z uživatele nebo jeho sledování. Pro minimalizaci možnosti útočníků provádět XSS útoky je nutné, aby webové prohlížeče neumožňovaly přenos informace mezi dokumenty z různých domén.

Při vývoji aplikace se ukázalo problematické použití více rámců iframe, pokud jejich obsahem byl dokument načtený z jiné domény, než hlavní rámec. Prohlížeč v tomto případě musí zajistit, aby se interpretovaly zcela izolovaně. Skript nesmí mít přístup k DOM takového dokumentu, nemůže z něj zachytávat události ani spouštět v něm existující kód. Toto omezení bylo nutno řešit službou Local Browser, popsanou v kapitole 5.1.3.

5.3.2 Zpracování událostí

Při vytváření webové aplikace patří přiřazování a obsluha událostí mezi základní prostředky, kterými lze zajistit interakci uživatele s programem. Standardní model událostí, který definovala organizace W3C ve specifikaci DOM level 2, bohužel není kompatibilní s modelem událostí prohlížeče Internet Explorer. Proto bylo nutno vytvořit společné rozhraní pro přiřazení a odebrání události. Alternativním přístupem by byla doimplementace standardního rozhraní do Internet Exploreru [viz 13]. Tato možnost je však složitější a pro naše potřeby (práce s dokumentem v iframe) není potřeba.

Společné rozhraní pro přiřazování a odebírání události bylo implementováno metodami *aE (attachEvent)* a *rE (removeEvent*) třídy *domEngine*.

5.3.3 Práce s označenou oblastí

Při vytváření extrakčního pravidla by měl mít uživatel možnost předat programu informaci o pozici, délce a dalších vlastnostech cílového textu. Jako praktické se ukázalo řešení, nechat uživatele myší označit cílový text a potřebné informace o oblasti poté zjistit z objektu *window.getSelection* (resp. *document.selection* pro Internet Explorer). Zjišťované atributy jsou pozice začátku označeného textu, jeho délka a pozice konce označeného textu od konce.

Nestandardní implementace objektu *selection* v prohlížeči Internet Explorer však neumožňuje jednoduše zjistit počáteční a koncovou pozici oblasti v rámci textu. Naštěstí lze toto omezení obejít vytvořením virtuální označené oblasti, jejíž počátek nastavíme na první znak textu v elementu a konec na poslední znak v označené oblasti. Pokud odečteme délky těchto výběrů, zjistíme počáteční pozici původní označené oblasti.

Výše uvedená logika pro zpracování pozice oblasti v kontextu textu elementu je zajištěna metodou *processSelection* třídy *ruleProcessorEngine*.

5.3.4 CSS styly v JavaScriptu

Práce s CSS styly je v rámci JavaScriptu poměrně problematická a v některých případech lze řešení nalézt velmi obtížně. Při vývoji aplikace bylo nutné ošetřit následující problémy:

- Napojení externích stylů pro dynamicky vytvářený obsah. Pokud je do stránky napojen soubor s CSS styly (pomoci metody *appendChild* u elementu head), načtené definice stylů se aplikují

v některých případech korektně, jindy pouze na již vytvořený obsah. Bohužel se nepodařilo identifikovat příčinu tohoto jevu, proto jsou styly načítány přilinkováním stylů příslušným HTML tagem v hlavičce stránky.

- Názvy vlastností CSS stylů mohou obsahovat nestandardní znaky (např. pomlčka), což zamezuje použití standardní JS syntaxe pro přístup k hodnotám. Způsobů jak hodnoty získat existuje několik, každá z nich však funguje pouze pro konkrétní prohlížeč. Ve třídě *domEngine* zajišťuje načítání hodnot CSS stylů metoda *gS (getStyles)*. Více o této problematice lze nalézt na [12].

- Vnitřní reprezentace hodnot stylů v prohlížečích není jednotná. Některé z nich převádí hodnoty do vlastní formy, jiné ve stylech nechávají uživatelem nastavený obsah. Toto je obzvlášť problém u barevných hodnot, které lze v CSS vyjádřit množstvím zápisů (procentuální, desítkový, šestnáctkový, jednotlivé pojmenované barvy). Funkce, která zjišťuje hodnotu nastaveného stylu tedy musí korektně zpracovat jakýkoli z těchto zápisů.

5.3.5 AJAX a kódování extrakčních pravidel

Při zpracování AJAX požadavku prohlížeč automaticky předpokládá data v kódování UTF-8. V našem případě jsou data obecně v různém kódování, protože stránky načítáme z různých domén a často i v různých jazycích. Znaky s diakritickými znaménky obsažené v extrakčních pravidlech jsou pak nahrazeny otazníky nebo zcela jinými znaky.

Lze uvažovat o 2 řešeních a to o detekci kódování z hlavičky a převodu textu stránky do UTF-8, nebo o transformaci extrakčního pravidla do kódování, které diakritická znaménka neobsahuje. První možnost vyžaduje kvalitní nástroj pro převod kódování (např. PHP knihovna *iconv*), ale nezaručuje 100% spolehlivost. Zdrojem chyb může být špatná detekce kódování, špatný formát hlavičky, nebo může informace o kódování zcela chybět.

Variantě pro převod textu do formy bez diakritiky vyhovuje kódování Base64, pro který dokonce existuje JS knihovna podporující víceznakové kódování UTF-8[12]. Přenos diakritiky je takto ošetřen, nárůst velikosti pravidla o 25% lze při jeho velikosti akceptovat (nárůst odpovídá několika set bajtům) jak pro přenos, tak pro uložení v databázi serveru.

[12] JS Base64 encoding - http://www.webtoolkit.info/javascript-base64.html

6 Experimenty

Jak bylo uvedeno v kapitole 2.1, pro hodnocení extrakce informace se obvykle používají parametry *precision* (přesnost) a *recall* (úplnost), kde lze také nalézt vzorce pro jejich výpočet. Pro posouzení vlastností implementovaného algoritmu proto byly použity právě tyto parametry.

Extrakce parametrů probíhala na několika serverech z testovací množiny dat. Vybrány byly 2 testovací kategorie (pevné disky, digitální fotoaparáty), u kterých bylo vytvořeno celkem 10 parametrů. Definice parametrů zahrnovaly číselné hodnoty, číselné hodnoty s jednotkami i řetězce.

Následně byly procházeny vybrané servery a pro každý z nich byla vytvořena sada extrakčních pravidel. Schopnost extrahovat hodnoty parametrů se testovala na dalších výrobcích z aktuálního serveru. Zaznamenáván byl počet parametrů k extrakci, počet dostupných parametrů na stránce a počet nalezených a nenalezených parametrů. Sledován byl také počet případů, kdy sice systém hodnotu nalezl, ale přiřadil jí nižší relevanci, takže bylo nutno vybírat z našeptávače. Při vyhodnocení však byly tyto parametry brány jako nenalezené.

Podrobné výsledky experimentů lze nalézt v příloze č.5, jejich shrnutí pak uvádí následující tabulka:

Název serveru	Typ dat	Přesnost (%)	Úplnost (%)
Alfacomp.cz	Tabulka	97	100
Cybex.cz	Tabulka	86	100
Alza.cz	Souvislý text	0	0
Digiboss.cz	Tabulka	92	92
Aaron.cz	Tabulka	90	100
Megapixel.cz	Tabulka	85	85
Czechcomputer.cz	Nečíslovaný seznam	14	13
Průměr		**79**	**74**

Tabulka č. 1 - Přesnost a úplnost algoritmu dosažená při experimentech

Ze získaných výsledků je zřejmé, že algoritmus má dobré vlastnosti pro tabulková data, kde dosahuje přesnosti kolem 90%, u parametru úplnost není výjimkou hodnota 100%. Pro seznamy je bohužel efektivita algoritmu velice nízká (10-15%), i když byly předpokládány hodnoty podobné jako u tabulek. Po bližším pohledu na podrobné výsledky však lze odhalit, že elementy s hodnotou parametru byly nalezeny korektně, jejich textová hodnota však byla nevhodně extrahována. Jednoduchou optimalizací algoritmu proto bude možné dosáhnout dobrých výsledků i pro data v seznamech.

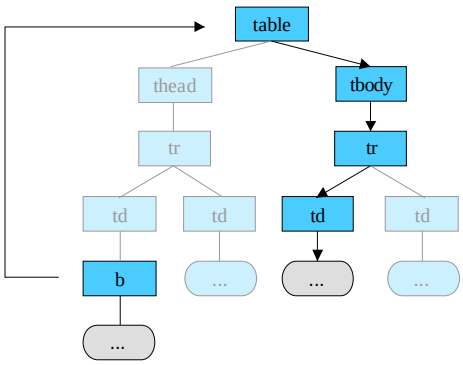
U dat uložených ve formě souvislého textu algoritmus zcela selhal, což bylo předpokládáno. Pro extrakci z tohoto typu dat je nutné zvolit metodu, která nezávisí na struktuře uložených dat v rámci dokumentu.

6.1 Další rozvoj systému

Po provedení experimentů bylo zjištěno, že systém pro extrakci dat je v současné podobě pro praxi použitelný, ale práci s ním je možné dalšími optimalizacemi dále urychlit.

Rozšíření, které by na zlepšení efektivity systému mělo největší vliv, by byla jednoduchá schopnost učení. V některých případech nastala situace, kdy bylo nalezeno více elementů s relevancí 100%, zejména pokud je název parametru příliš obecný. Uživatel tedy musí z našeptávače vybrat správnou hodnotu a v tomto případě by systém automaticky zaznamenal, o kterou hodnotu jde a na tomto základě by extrakční pravidlo doplnil. Při dalším použití by se doplnila již správná hodnota.

Úprava algoritmu pro hledání cesty mezi dvěma uzly v DOM by umožnila lépe reagovat na změny ve struktuře dokumentu a redukovala by velikost pravidel. Takovou úpravou může být např. zavedení univerzálního uzlu (ekvivalent * v regulárních výrazech), který by umožňoval přeskočit libovolný počet elementů. Tento princip lze samozřejmě použít pouze pro směr cesty od názvu elementu ke kořenovému uzlu. Algoritmus by univerzální uzel přeskočil a hledal následující element na cestě, libovolně vzdálený od aktuální pozice.




```
path: [{tn: "b", nthChild:1,  jump: true },
       {lev:0, tn:"table", nthChild:1 },
       {lev:1, tn:"tbody", nthChild:1 },
       {lev:2, tn:"tr", nthChild:1 },
       {lev:3, tn:"td", nthChild:1 }],
```

Obrázek č. 14 - Ukázka rozšířeného algoritmu pro průchod DOM stromem

Další možností zlepšení funkčnosti systému je úprava zpracování vybraných částí textu. To zahrnuje nalezení metriky, která lépe popisuje pozici vybraného textu než současná, vylepšené zpracování jednotek. Systém lze také doplnit o převody číselných hodnot v rámci soustav o různých řádech, což umožní lépe převádět např. údaje o kapacitě disků.

V případě použití systému pro extrakci dat v jiném než českém jazyce je vhodné zavést možnost externích jazykových souborů.

7 Závěr

Cílem této práce bylo navrhnout a implementovat systém určený pro elektronické obchody, který by jejich pracovníkům poskytl automatizovanou podporu při vytváření a údržbě databáze zboží. Tento systém je schopen za pomoci předem definovaných extrakčních pravidel získávat informace o parametrech výrobků. Tímto je proces aktualizace databáze zefektivněn a je ušetřena zdlouhavá manuální práce zaměstnanců obchodu.

Za účelem návrhu takového systému byly prostudovány současné přístupy v oblasti extrakce informací. Značná pozornost byla věnována wrapperům a metodám pro jejich automatickou konstrukci. Jako perspektivní se v současnosti jeví také vizuální přístup k extrakci dat.

V další části práce byla za účelem analýzy sestavena testovací množina zdrojů dat. Ty byly rozděleny do tříd podle vlastností poskytovaných dat a jejich formy. Mezi nejčastější formy publikace dat o zboží patří tabulka, nebo souvislý textový popis. Vzhledem k možnostem automatické extrakce parametrů je nejvhodnější tabulka a seznam. Ten se však vyskytuje pouze zřídka.

Návrh systému, uvedený ve čtvrté kapitole, počítá s použitím wrapperu, jenž bude mít extrakční pravidla definovaná vizuální cestou. Jejich definice je pro obsluhu rychlá a intuitivní. Působnost pravidel byla navržena přes více kategorií, což redukuje množství práce uživatele při jejich zadávání. Pro komfortní ovládání aplikace bylo nutné nalézt nejčastější činnosti spojené s použitím systému a na jejich základě uživatelské rozhraní zoptimalizovat.

Samotný systém je implementován jako JavaScript aplikace, běžící v klientově prohlížeči. Serverová část zajišťuje výběr relevantních extrakčních pravidel a jejich načítání a ukládání. Algoritmus extrakce je založen na průchodu jednoznačnou cestou v DOM webové stránky. Tato metoda extrakce je vhodná pro většinu zdrojů poskytujících data využitelná pro elektronické obchody.

Nasazení systému do reálného provozu potvrdilo vysokou spolehlivost algoritmu pro dobře strukturovaná data, ale také některé nedostatky, zejména pak nedostatečnou optimalizaci algoritmu pro seznamy. Lepších výsledků je však možné dosáhnout nalezením vhodnější metriky pro vzdálenost v textu, případně drobnou úpravou algoritmu. Experimenty také potvrdily, že metoda hledání cesty není vhodná pro extrakci dat ze souvislého textu a pro tyto případy je vhodnější použití např. regulárních výrazů.

Literatura

[1] NAHM, Un Yong. Text Mining with Information Extraction. In Proceedings of the AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowl. Bases. 2002. Dostupný na WWW: <http://www.cs.utexas.edu/users/ml/papers/discotex-melm-03.pdf>. ISBN 0-496-01283-5.

[2] GRISHMAN, Ralph, SUNDHEIM, Beth. Message Understanding Conference - 6 : A Brief History. Proceedings 16th Int'l Conf. on Computational Linguistics (COLING 96). 1996, Dostupný na WWW: <http://acl.ldc.upenn.edu/C/C96/C96-1079.pdf>.

[3] KUSHMERICK, Nickolas, WELD, Daniel S., DOORENBOS, Robert B. Doorenbos. Wrapper Induction for Information Extraction. In Intl. Joint Conference on Artificial Intelligence (IJCAI). 1997. s. 729-737. Dostupný na WWW: <http://citeseer.ist.psu.edu/kushmerick97wrapper.html>.

[4] SODERLAND, Stephen. Learning to Extract Text-based Information from the World Wide Web. In Knowledge Discovery and Data Mining. 1997. s. 251-254. Dostupný na WWW: <http://citeseer.ist.psu.edu/soderland97learning.html>.

[5] CIRAVEGNA, Fabio. (LP)2 an Adaptive Algorithm for Information Extraction from Web-related Texts. In Proceedings of the IJCAI-2001. 2001. Dostupný na WWW: <http://eprints.aktors.org/120/01/Atem01.pdf>.

[6] FREITAG, Dayne. Information Extraction from HTML : Application of a General Machine Learning Approach. In Proceedings of the Fifteenth Conference on Artificial Intelligence. 1998. s. 517-523. Dostupný na WWW: <http://www.daviszhou.net/research/seminar/Freitig1998.pdf>.

[7] BURGET, Radek. Information Extraction from HTML Documents Based on Logical Document Structure. 2004. 85 s. Faculty of Information Technology VUT. Dizertační práce. Dostupný na WWW: <http://www.fit.vutbr.cz/research/view_pub.php.iso-8859-2?id=7607>.

[8] EMBLEY, David W., et al. A Conceptual-Modeling Approach to Extracting Data from the Web. In International Conference on Conceptual Modeling / the Entity Relationship Approach. 1998. s. 79-91. Dostupný na WWW: <http://pages.cs.wisc.edu/~smithr/pubs/er98.pdf>.

[9] GOLD, Mark. Language identification in the limit. Information and Control. 1967, vol. 10, no. 5, s. 447-474. Dostupný na WWW: <http://www.isrl.uiuc.edu/~amag/langev/paper/gold67limit.html>.

[10] FREITAG, Dayne. Using grammatical inference to improve precision in information extraction. In Workshop on Grammatical Inference, Automata Induction, and Language Acquisition (ICML-97). 1997. Dostupný na WWW: <citeseer.ist.psu.edu/freitag97using.html>.

[11] KOSALA, Raymond, et al. Information Extraction in Structured Documents Using Tree Automata Induction. In PKDD '02. 2002. s. 299-310. Dostupný na WWW: <http://citeseer.ist.psu.edu/574333.html>. ISBN 3-540-44037-2.

[12] KUSÝN, Michal. Pokročilé zpracování událostí napříč platformami podruhé [online]. 6. 3. 2003 [cit. 2008-05-05]. Dostupný z WWW: <http://interval.cz/clanky/pokrocile-zpracovani-udalosti-napric-platformami-podruhe/>. ISSN 1212-8651.

[13] SMITH, Garrett. Setting Styles with JavaScript : Getting Style [online]. [2007] [cit. 2008-05-05]. Dostupný z WWW: <http://dhtmlkitchen.com/learn/js/setstyle/index4.jsp>.

Seznam obrázků

Seznam vzorců

Seznam zkratek

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

CoNLL Conference on Computational Natural Language Learning

CSS Cascading Style Sheets

DARPA Defense Advanced Research Projects Agency

DOM Document Object Model

GUI Graphical user interface

HMM Hidden Markov Model

HTML Hyper Text Markup Language

IREX Information Retrieval and Extraction Exercise

JS JavaScript

JSON JavaScript Object Notation

LREC Language Resources and Evaluation Conference

MET Multilingual Entity Task Evaluation

MUC Message Understanding Conference

NraD Naval Research and Development Group

PDF Portable Document Format

PHP PHP Hypertext Preprocessor

RTF Rich Text Format

SVG Scalable Vector Graphics

W3C World Wide Web Consortium

WWW World Wide Web

XHTML Extensible HyperText Markup Language

XML Extensible Markup Language

XSS Cross-site scripting

Seznam příloh

Příloha č.1 – Zdroje dat

	DOMÉNA	P/S	Formát	Formátování dat	Typ hodnot	Vhodnost zdroje (1-5)
DVD přehravače	www.xoro.com	**P**	HTML	tabulka	textové hodnoty	4
	www.thomsonlink.com	**P**	PDF	tabulka	diskrétní hodnoty	5
	www.panasonic.cz	**P**	HTML	tabulka	diskrétní hodnoty	2
	www.jvc.cz	**P**	HTML	seznam	celé věty	5
	www.i-digi.cz	**S**	HTML	text s oddělovači	textové hodnoty	5
	www.datart.cz	**S**	HTML	tabulka	diskrétní hodnoty	2
	sigmatek.svet-nakupu.cz	**S**	HTML	tabulka, seznam	diskrétní hodnoty	1
	www.cddvd.cz	**S**	HTML	tabulka	diskrétní hodnoty	1
	www.sigmatek.cz	**P**	HTML	tabulka	textové hodnoty	2
	www.ferguson-digital.cz	**P**	HTML	tabulka	diskrétní hodnoty	2
HDD	www.seagate.com	**P**	HTML	tabulka	diskrétní hodnoty	1
	www.cybex.cz	**S**	HTML	tabulka	diskrétní hodnoty	1
	www.wdc.com	**P**	HTML	víceúrovňová tabulka	diskrétní hodnoty	3
	www.hitachigst.com	**P**	PDF	víceúrovňová tabulka	diskrétní hodnoty	5
	www.alfacomp.cz	**S**	HTML	tabulka	diskrétní hodnoty	1
	sdd.toshiba.com	**P**	HTML	tabulka	diskrétní hodnoty	1
Fotoaparáty	www.sony.cz	**P**	HTML	tabulka	diskrétní hodnoty	1
	www.europe-nikon.com	**P**	HTML	tabulka	textové hodnoty	4
	www.olympus.cz	**P**	HTML	tabulka	diskrétní hodnoty	2
	www.megapixel.cz	**S**	HTML	tabulka	diskrétní hodnoty	1
	www.digineff.cz	**S**	HTML	tabulka	diskrétní hodnoty	1
	www.steves-digicams.com	**S**	HTML	víceúrovňová tabulka	textové hodnoty	3
	www.pentaximaging.com	**P**	HTML	tabulka	celé věty	5
	www.canon.cz	**P**	HTML	tabulka	textové hodnoty	3
LCD dipleje	www.patro.cz	**S**	HTML	text s oddělovači	textové hodnoty	2
	www.benq.com	**P**	HTML	víceúrovňová tabulka	diskrétní hodnoty	2
	www.elektro-garden.cz	**S**	HTML	tabulka, textový seznam	diskrétní hodnoty	1
	www.consumer.philips.com	**P**	HTML	tabulka	diskrétní hodnoty	1
	www.samsung.com	**P**	HTML	víceúrovňová tabulka	diskrétní hodnoty	2
	cz.lge.com	**P**	HTML	tabulka	diskrétní hodnoty	1
	www.eternity.cz	**S**	HTML	tabulka	diskrétní hodnoty	1
	www.acer.cz	**P**	HTML	víceúrovňová tabulka	diskrétní hodnoty	3
	cz.asus.com	**P**	HTML	tabulka	textové hodnoty	3
	clanky.katalogmonitoru.cz	**S**	HTML	tabulka	diskrétní hodnoty	1

Legenda:
P = primární, S = sekundární
Vhodnost zdroje: 1 - nejlepší, 5 - nejhorší

Příloha č.2 – Seznam konfiguračních parametrů

Následující konfigurační parametry lze použít při inicializaci objektu *extractor*. Ukázka použití je uvedena v příloze č.3.

- **extractorWrapId** [default='extractorContainer']
  ID elementu na stránce, kde bude umístěno GUI systému.
- **parameterWrapClass** [default='paramContainer']
  Název třídy, kterou jsou označeny řádky s parametry. V HTML dokumentu je definice třídy vícenásobná. Druhá třída označuje číslo parametru (např. <tr class='paramContainer 3'>
- **parameterObj** [default='parameter']
  ID pole s hodnotou parametru. V HTML dokumentu jsou pak parametry ve formátu <input type=‘text‘ id=‘parameter[3]’ name=’parameter[3]’ />
- **urlId** [default='url']
  ID elementu, ve kterém je adresa výrobku. Při inicializaci se z tohoto pole načte adresa, na kterou je Local Browser přesměrován.
- **categoryId** [default='id_category']
  ID elementu <select>, ve kterém lze vybrat ID aktuální kategorie.
- **localBrowserURL** [default='/extractorLocalBrowser.php']
  URL adresa, na kterém běží Local Browser.
- **localBrowserParam** [default='url']
  Název GET parametru, ve kterém se předává URL adresa pro zpracování.
- **localBrowserDefWidth** [default=400]
  Standardní šířka Local Browseru v pixelech.
- **localBrowserDefHeight** [default=250]
  Standardní výška Local Browseru v pixelech.
- **rulesProviderURL** [default='/extractorRulesLoad.php']
  URL adresa, která poskytuje aktuální sady pravidel.
- **rulesSubscriberURL** [default='/extractorRulesSave.php']
  URL adresa, na které lze uložit sady pravidel.
- **imgPath** [default='images']
  Cesta do adresáře s obrázky (bez lomítka na konci).
- **debug** [default=false]
  Zapnutí/vypnutí vypisování ladících informací.

Příloha č.3 – Příklad integrace systému do webové stránky

Příklad včlenění objektu *extractor* do HTML stránky, jeho inicializace a konfigurace (kód je třeba umístit do hlavičky dokumentu):

```
<link rel="stylesheet" type="text/css" href="extractorStyles.css" media="all" />
<!--[if IE 7]>
<link rel="stylesheet" type="text/css" href="extractorStylesIE7.css" media="all" />
<![endif]-->
<script type="text/javascript" src="extractor.js"></script>
<script>
<!--

function extract_init() {
  extractor.init({
    extractorWrapId : 'extractor_container',
    parameterWrapClass : 'param_container',
    parameterObj : 'parameter',
    urlId : 'urlFirm',
    categoryId : 'idCategory',
    localBrowserURL: 'http://spromo.cz/extractorLocalBrowser.php'
    rulesProviderURL: 'http://spromo.cz/extractorRulesLoad.php',
    rulesSubscriberURL: 'http://spromo.cz/extractorRulesSave.php',
    debug : true
  });
}

window.onload = function(){ extract_init(); };

//-->
</script>
```

Na HTML stránce je poté třeba definovat místo, kde bude aplikace umístěna:

```
<div id='extractor_container'></div>
```

Příloha č.4 – Popis použitých datových struktur

Typ rule

- **rule.path** (kolekce typů rulePath) – cesta od elementu s názvem parametru k elementu s hodnotou parametru
- **rule.selName** (typ ruleText) – informace o vybraném textu názvu parametru
- **rule.selValue** (typ ruleText) – informace o vybraném textu hodnoty parametru

Typ rulePath

- **rulePath.lev** (integer) – vzdálenost od kořenového uzlu cesty
- **rulePath.tn** (string) – název elementu (tagName)
- **rulePath.nthChild** (integer) – pořadí elementu v rámci svých sourozenců. (berou se v úvahu pouze uzly typu element (nodeType=3), ignorovány by měly být textové uzly, komentáře, sekce CDATA a další.
- **rulePath.dist** (integer) [optional] – vzdálenost mezi uzly pod kořenovým uzlem cesty (kladné číslo značí počet sourozenců za elementem, záporné číslo před elementem).

Typ ruleText

- **ruleText.selText** (string) – uživatelem vybraný text
- **ruleText.selAll** (boolean) – je vybraný text veškerým textem v elementu?
- **ruleText.prefixSpaces** (integer) – počet skupin mezer před vybraným textem
- **ruleText.suffixSpaces** (integer) – počet skupin mezer za vybraným textem
- **ruleText.spaces** (integer) – počet skupin mezer vybraného textu
- **ruleText.posStart** (integer) [optional] – počet znaků před počátkem vybraného textu

Příloha č.5 – Tabulka výsledků experimentální extrakce dat

HDD	celkem parametrů	počet údajů na stránce	správně doplněno	pouze nalezeno	nenalezeno	nalezeno chybně	Úplnost	Přesnost
Alfacomp.cz								
(tabulka)	5	5	5	0	0	0		
	5	5	5	0	0	0		
	5	5	5	0	0	0		
	5	5	5	0	0	0		
	5	5	5	0	0	0		
	5	4	4	0	0	1		
	30	**29**	**29**	**0**	**0**	**1**	**100**	**96,6667 %**
Cybex.cz								
(tabulka)	5	5	5	0	0	0		
	5	4	4	0	0	1		
	5	5	5	0	0	0		
	5	4	4	0	0	1		
	5	5	5	0	0	0		
	5	4	4	0	0	1		
	5	3	3	0	0	2		
	35	**30**	**30**	**0**	**0**	**5**	**100**	**85,7143 %**
Alza.cz	5	5	0	0	0	5		
(souvislý text)	5	5	0	0	0	5		
	5	5	0	0	0	5	zastaveno, nechytá se	
	15	**15**	**0**	**0**	**0**	**15**	**0**	**0 %**
Digiboss.cz	5	5	5	0	0	0		
(tabulka)	5	5	4	0	0	1		
	5	5	5	0	0	0		
	5	5	4	0	0	1		
	5	5	5	0	0	0		
	25	**25**	**23**	**0**	**0**	**2**	**92**	**92 %**
Fotoparáty								
Aaron.cz	4	4	4	0	0	0		
(tabulka)	4	4	4	0	0	0		
	4	4	4	0	0	0		
	4	3	3	0	1	0		
	4	3	3	0	1	0		
	20	**18**	**18**	**0**	**2**	**0**	**100**	**90 %**
Megapixel.cz	4	4	2	0	0	2		
(tabulka)	4	4	3	0	0	1		
	4	4	4	0	0	0		
	4	4	4	0	0	0		
	4	4	4	0	0	0		
	20	**20**	**17**	**0**	**0**	**3**	**85**	**85 %**
Czechcomputer.cz	4	2	0	0	2	2		
(seznam)	4	4	0	4	0	0		
	4	4	0	4	0	0		
	4	4	2	2	0	0		
	16	**14**	**2**	**10**	**2**	**2**	**14,2857**	**12,5 %**
CELKEM	161	151	119	10	4	28	**78,8079**	**73,913 %**

Uplnost = správně / na stránce
Přesnost = správně / všech