



**Slovenská technická univerzita**  
**FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ**  
Ilkovičova 3, 842 16 Bratislava 4



# **Tvorba obal'ovačov na získavanie informácií z webu**

**Softvérový systém**

Tím číslo 5: *Lubomír Chamraz , Ivan Kišac , Ján Krausko , Michal Kur'ák , Marián Šimko , Michal Šimún*  
Vedúci tímu: *Mgr. György Frivolt*  
Študijný odbor / program: *Softvérové inžinierstvo / Softvérové inžinierstvo*  
Ročník, typ štúdia: *1, inžinierske štúdium*  
Dátum odovzdania: *máj 2007*



# Obsah

1 Úvod.....	1 - 1
1.1 Slovník pojmov.....	1 - 1
1.2 Použité skratky.....	1 - 2
1.3 Použitá notácia.....	1 - 3
2 Existujúce nástroje na tvorbu obalovačov.....	2 - 1
2.1 Kapow RoboSuite.....	2 - 1
2.1.1 Architektúra nástroja.....	2 - 1
2.1.2 ModelMaker.....	2 - 2
2.1.3 RoboMaker.....	2 - 2
2.1.4 Lokalizácia častí dokumentu (tag finders).....	2 - 3
2.1.5 Ošetrovanie chýb.....	2 - 4
2.1.6 Zhodnotenie nástroja.....	2 - 4
2.2 Lixto Visual Wrapper.....	2 - 5
2.2.1 Architektúra nástroja.....	2 - 5
2.2.2 Základné prvky – vzory, filtre, podmienky .....	2 - 6
2.2.3 Elog.....	2 - 9
2.2.4 XML Tool.....	2 - 9
2.3 Aplikačný rámec Wrapper Suite.....	2 - 10
2.3.1 Architektúra.....	2 - 10
2.3.2 Program obalovača.....	2 - 11
2.3.3 Výkonná časť obalovača.....	2 - 13
2.3.4 Načítanie obalovača.....	2 - 13
2.3.5 Pomocné triedy.....	2 - 14
2.3.6 Architektúra zápisu do cieľových prostredí.....	2 - 14
3 Opis riešenia.....	3 - 1
3.1 Ciele a vlastnosti produktu.....	3 - 1
3.2 Prehľad produktu.....	3 - 2
4 Špecifikácia obalovača.....	4 - 1
4.1 Prípady použitia.....	4 - 1
4.2 Popis prípadov použitia.....	4 - 2
5 Použité technológie.....	5 - 1
5.1 Jazyk XUL.....	5 - 1
5.1.1 Použitie jazyka XUL.....	5 - 1
5.1.2 Štruktúra jazyka XUL.....	5 - 1
5.2 Komponentový objektový model XPCOM.....	5 - 2
5.2.1 Rozhrania.....	5 - 3
5.2.2 XPConnect.....	5 - 3
5.2.3 JavaXPCOM.....	5 - 4
6 Hrubý návrh systému.....	6 - 1
6.1 Architektúra systému.....	6 - 1
6.2 Prezentačný modul aplikácie.....	6 - 1
6.2.1 Proces tvorby obalovača.....	6 - 2
6.2.2 Proces úpravy obalovača.....	6 - 2
6.2.3 Spustenie obalovača.....	6 - 3
6.3 Prezentačný modul prehliadača.....	6 - 3
6.4 Komunikačný modul.....	6 - 3
6.5 Aplikačný modul – Interpreter.....	6 - 4
6.5.1 Kontext.....	6 - 4
6.5.2 Akcia.....	6 - 5
6.5.3 Vzor.....	6 - 5

6.5.4 Cookies.....	6 - 5
6.5.5 Autentifikačné údaje.....	6 - 5
6.5.6 Ošetrenie výnimiek.....	6 - 5
6.6 Aplikačný modul – Tvorba vzorov.....	6 - 6
6.6.1 Vzor.....	6 - 6
6.6.2 Typy filtrov.....	6 - 7
6.6.3 Učenie.....	6 - 8
6.7 Aplikačný modul – Tvorba obalovača.....	6 - 10
6.7.1 Akcie obalovača.....	6 - 10
6.7.2 Ostatné charakteristiky obalovača.....	6 - 12
6.8 Výstupné objekty.....	6 - 13
6.9 Implementačné prostriedky a vývojové nástroje.....	6 - 13
7 Prototyp riešenia.....	7 - 1
7.1 Ciele.....	7 - 1
7.2 Zmeny v jadre.....	7 - 2
7.2.1 Návrh modulu „Tvorba vzorov“.....	7 - 2
7.2.2 Integrácia modulu „Tvorba vzorov“ do jadra systému.....	7 - 4
7.3 Stratégie učenia.....	7 - 4
7.3.1 Opakujúca stratégia učenia XPath príkladov (RepeatingXPathExampleLearningStrategy).....	7 - 4
7.3.2 Jednoduchá stratégia učenia XPath príkladov (SimpleXPathExampleLearningStrategy).....	7 - 5
7.4 Používateľské rozhranie.....	7 - 6
8 Použité pramene.....	8 - 1
Príloha A: Používateľská príručka prototypu.....	A - 1
A.1 WrapperDesigner.....	A - 1
A.2 Demonštrácia prototypu.....	A - 5

# 1 Úvod

Predložený dokument opisuje riešenie zadania „Tvorba obalovačov na získavanie informácií z webu“. Ide o dokumentáciu softvérového systému, v jednotlivých kapitolách sa postupne venujeme všetkým etapám jeho vývoja.

Kapitola 2 sa zaoberá analýzou existujúcich riešení na pôde predprogramovaných obalovačov. Približujeme v súčasnosti najpoužívanejšie riešenia, ako aj aplikačný rámec, ktorý bol vytvorený v tímovom projekte počas minulého roka. Na základe výsledkov analýzy definujeme v kapitole 3 ciele projektu a požiadavky, ktoré sú softvérový systém kladené. Kapitola 4 sa venuje špecifikácií obalovača z hľadiska prípadov jeho použitia. V kapitole 5 je uvedený stručný opis nových technológií, ktoré je pri návrhu požadovanej funkcionality obalovača potrebné poznať a ktoré budú súčasťou nášho riešenia. Kapitola 6 je zameraná na hrubý návrh systému a na základe uvedenej architektúry rozoberá jednotlivé moduly navrhovaného riešenia.

## 1.1 Slovník pojmov

Pojmy týkajúce sa procesnej funkcionality obalovača:

Obalovač (angl. wrapper)	Nástroj na dolovanie dát z neštruktúrovaného vstupu
Extrakcia	Proces dolovania semištruktúrovaných dát do štruktúrovaného výstupu
Navigácia	Pohyb po dokumentoch s účelom dostať sa k cieľovému zdroju
Kontext	Dátová štruktúra, ktorá obsahuje aktuálny stav obalovania patriaci k akcii. Obsahuje vstupné premenné, vzor, aktuálny subdokument, lokálny výstupný objekt
Vstupný parameter	Podmienka obalovania, zvyčajne zadaná používateľom
Akcia	Operácia nad kontextom. Extrakčná, navigačná
Vzor	Výber subdokumentu. Obsahuje filtre ako kritérium na tento výber. Aplikovaním vzoru na subdokument vznikajú nové subdokumenty
Filter	Kritérium na výber subdokumentu; v prípade XPath výrazu – DOM, v prípade regulárneho výrazu – String
Subdokument	Časť dokumentu vybraná filtrom
Výstupný objekt	Objekt obsahujúci výsledky procesu obalovania

Pojmy týkajúce sa učenia obalovača:

Učenie	Proces generalizácie a konkretizácie vzoru na základe interakcie s používateľom
Príklad	Časť dokumentu, ktorý špecifikoval používateľ. Pozitívny alebo negatívny
Generalizácia	Zovšeobecnenie príkladov také, že vytvorený vzor obsahuje všetky pozitívne a žiaden negatívny príklad

Ostatné pojmy:

cookie	Krátka textová informácie o internetovom spojení týkajúcom sa relácie so vzdialeným serverom. Je uložená na strane klienta a obsahuje informácie o používateľovi týkajúce sa napr. autentifikácie, preferovaných vlastností, atď.
JavaBeans	Znovupoužiteľné softvérové komponenty napísané v programovacím jazyku Java.

## 1.2 Použité skratky

CSS	Cascade Sheet Styling	Kaskádové štýly
DOM	Document Object Model	Objektový model dokumentu
HTML	HyperText Markup Language	Hypertextový značkovací jazyk
HTTP	HyperText Transfer Protocol	Hypertextový transportný protokol
IDL	Interface Description Language	Jazyk opisu rozhraní
URL	Uniform Resource Locator	Jednotný identifikátor zdroja
XML	eXtensible Markup Language	Rozšíriteľný značkovací jazyk
XPCOM	Cross Platform Component Object Model	Multiplatformový komponentový objektový model
XPCOM	Cross Platform Connect	Multiplatformové prepojenie
XPIDL	Cross Platform Interface Description Language	Multiplatformový jazyk opisu rozhraní
XUL	XML User Interface Language	Jazyk na opis používateľského rozhrania založený na XML

## 1.3 Použitá notácia

<code>ForEachTag</code>	neproporcionálnym písmom budeme označovať úryvky z programového kódu, názvy tried a pod.
<b>Lixto Wrapper Designer</b>	tučné písmo zvýrazňuje a zdôrazňuje obzvlášť dôležité kľúčové slová
<i>Actions</i>	<ul style="list-style-type: none"><li>• kurzívou budú označené názvy XML elementov, resp. ich atribútov</li><li>• taktiež budú zvýraznené menej dôležité kľúčové slová</li></ul>





## 2 Existujúce nástroje na tvorbu obalovačov

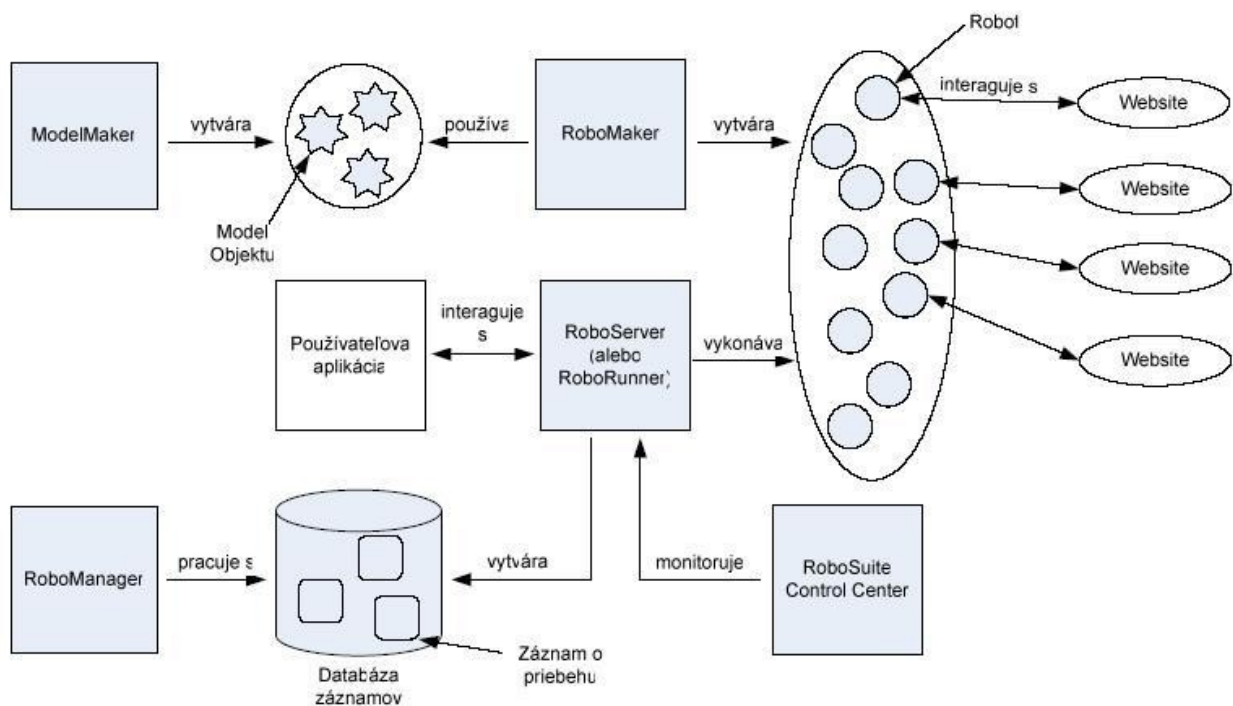
V tejto kapitole sa postupne pozrieme na najpoužívanejšie nástroje na tvorbu obalovačov (časti 2.1 a 2.2) a pokúsime sa analyzovať aplikačný rámec Wrapper Suite, ktorý je výsledkom tímového projektu minulého roka (časť 2.3).

### 2.1 Kapow RoboSuite

RoboSuite je nástroj, ktorý napomáha transformovať informácie a funkcionality webových systémov do presne definovaného formátu [1]. Funkcionality interakcie, navigácie, extrakcie a integrácie dát obsiahnutých v informačnom priestore zabezpečuje robot (obalovač).

#### 2.1.1 Architektúra nástroja

RoboSuite pozostáva z viacerých aplikačných častí. Obr. 2-1 zobrazuje vzťahy medzi definovanými časťami nástroja.



Obr. 2-1: Architektúra nástroja RoboSuite [1]

Prvou etapou používania nástroja RoboSuite je modelovanie charakteristík robota pomocou aplikačných častí ModelMaker a RoboMaker. Program vytvoreného robota možno vykonávať viacerými spôsobmi:

- dávková úloha pomocou aplikačnej časti RoboRunner, ktorý umožňuje spustenie robota z príkazového riadku
- spustenie robota zo vzdialenej klientskej aplikácie, ktoré zabezpečuje RoboServer; vzdialený monitoring časti RoboServers a ich robotov umožňuje aplikačná časť RoboSuite Control

## Center

RoboManager predstavuje nástroj na riadenia a udržiavanie viacerých robotov, umožňuje tiež prezeranie stavových a chybových hlásení vygenerovaných robotmi.

V nasledujúcej časti opisu nástroja Kapow RoboSuite sa budeme venovať aplikačným častiam na tvorbu obalovača (ModelMaker a RoboMaker), keďže tie predstavujú najdôležitejšie časti nástroja vzhľadom na ciele nášho projektu.

### 2.1.2 ModelMaker

ModelMaker predstavuje aplikáciu pre tvorbu objektov, s ktorými pracuje robot [2]. Objekty modelujú reálne časti informačného priestoru a možno ich rozdeliť na nasledovné časti:

- vstupné objekty – objekty, ktoré slúži ako vstup robota pre vykonávanie jeho práce. Medzi vstupné objekty možno zaradiť napríklad dáta, ktoré bude robot vkladať do formulárov
- výstupné objekty – objekty extrahované robotom, stavové alebo chybové objekty
- databázové výstupné objekty – objekty, ktoré sú extrahované a následne uložené do špecifikovanej databázy

Objekty sa skladajú z atribútov modelujúcich určitú charakteristiku objektov a predstavujú miesto v objekte, kde sú dáta uložené (napr. názov pracovnej ponuky). Vytvorené objekty sú združené do doménového modelu, ktorý predstavuje informačný priestor robota.

### 2.1.3 RoboMaker

RoboMaker predstavuje aplikačnú časť nástroja slúžiacu na tvorbu robota. Môže byť dvoch typov:

- robot na zbieranie dát (data collection robot)
- robot na získavanie funkcionality web systému alebo jeho časti (clipping robot)

Ďalej sa budeme zaoberať iba prvým typom robota, keďže cieľom nášho projektu je tvorba nástroja na extrakciu dát.

Robot je realizovaný ako stavový automat, ktorý sa počas vykonávania svojej činnosti nachádza v určitom stave [3], ktorý je definovaný nasledovnými komponentmi:

- *Windows* – predstavuje práve otvorenú web stránku, s ktorou robot práve pracuje
- *Objects* – aktuálne hodnoty objektov
- *Cookies* – HTTP cookies získané počas komunikácie s web serverom
- *Autentications* – autentifikačné údaje získané počas komunikácie s web serverom

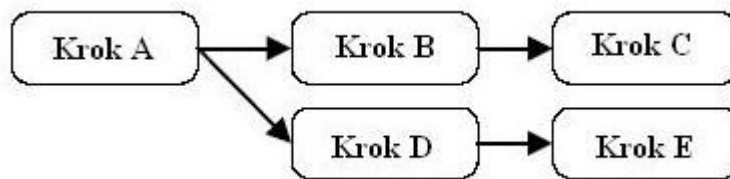
Stavový automat robota pozostáva z jednotlivých krokov, ktoré určujú elementárnu činnosť robota. Krok prijíma ako vstup stav robota a produkuje v závislosti od typu kroku nula alebo viac stavov robota. Pri tvorbe kroku je možno využiť jednu z definovaných typov akcií:

- navigačná akcia – získanie definovanej web stránky, nasledovanie odkazu
- extrakčné akcie – výber lokalizovaných dát z dokumentu, ich konverzia do definovaného

formátu z textového formátu a uloženie do atribútov objektu. Získané dáta môžu byť pred uložením do objektov spracované pomocou regulárnych výrazov

- iteračné akcie – akcie, ktoré sa vykonávajú pre všetky lokalizované časti dokumentu (napr. `ForEachTag`)
- vyplňovanie formulárov
  - Krok pozostáva z nasledujúcich častí:
    - názov kroku
    - tag finders – lokalizácia časti dokumentu (jeden alebo viac elementov)
    - step action – akcie, ktoré vykonávajú zmenu stavu robota v závislosti od typu akcie. Napríklad extrakčná akcia mení objekty robota alebo navigačné akcie menia *Windows*, *Cookies* a *Authentications*

Posledným krokom v tvorbe robota je vytvorenie spojení medzi špecifikovanými krokmi, ktoré definujú poradie vykonávania jednotlivých akcií. Výstupný stav kroku robota predstavuje vstupný stav robota nasledujúceho kroku vo vytvorenej postupnosti krokov. Špeciálnou akciou je iteračná akcia, ktorá produkuje viacero výstupných stavov (pre každý krok iterácie jeden stav). Krok môže byť spojený s viacerými nasledujúcim krokmi, takýto typ spojenia predstavuje vetvenie. Príklad postupnosti krokov je zobrazený na Obr. 2-2.



Obr. 2-2: Príklad postupnosti krokov robota [3]

Vykonanie akcií zobrazených na obrázku závisí na móde vetvenia. V prostredí tvorby RoboMaker je možné zdefinovať dva módy vetvenia:

- vykonanie všetkých vetiev – paralelne sa vykonávajú všetky vetvy
- vykonaj vetvy pokiaľ sa nenájde úspešná vetva – postupne sa vykonávajú všetky vetvy akcií v definovanom poradí, pokiaľ nejaká vetva neskončí úspešnou akciou, ktorá nemá žiadny chybový stav

Vyššie uvedený príklad vykoná najprv akcie v kroku A a výstupný stav tohto kroku bude predstavovať vstupný stav kroku B a tiež kroku D. Za predpokladu, že je nastavený krok vetvenia vykonania všetkých vetiev a zároveň sa nevyskytnú žiadne chybové stavy, bude po ukončení akcie v kroku B vykonaný krok C a po ukončení akcie v kroku D aj krok E.

#### 2.1.4 Lokalizácia častí dokumentu (tag finders)

Tag finders slúžia na určenie časti dokumentu, kde bude aplikovaná daná akcia [3]. Pri určení časti dokumentu je možné využiť stromovú štruktúru HTML dokumentu a/alebo prácu

s regulárnymi výrazmi. Tag finders pozostávajú z nasledovných častí:

- *Find where* – špecifikuje, kde sa má daná časť dokumentu vyhľadať (je možné určiť rozsah celej stránky)
- *Tag Path* – špecifikovanie cesty v stromovej štruktúre dokumentu. Tag Path predstavuje upravené XPath s obmedzenou vyjadrovacou silou a upravenou syntaxou
- *Attribute name* – možnosť špecifikovania, že element musí mať daný atribút
- *Attribute value* – možnosť špecifikovania, že určený atribút nadobúda danú hodnotu. Hodnotu atribútu možno určiť pomocou regulárnych výrazov
- *Tag pattern* – možnosť určenia vzoru elementu pomocou regulárnych výrazov (napr. `.*<b>\w+<b>.*`)
- *Tag depth* – špecifikuje hĺbku elementu v strome dokumentu
- *Tag number* – určuje počet elementov v prípade, že viacero elementov spĺňa definované podmienky

### 2.1.5 Ošetrovanie chýb

Kroky robota môžu pri svojej činnosti generovať chybové stavy, napr. v prípade, že sa nemôžu lokalizovať požadované dáta v dokumente, s ktorými má akcia pracovať. Každému kroku robota možno priradiť vlastné ošetrovanie chybových stavov, ktoré spĺňa jednu z nasledovných typov ošetrovania chýb [3]:

- *reportuj tu* – vykonávanie robota sa zastaví a vykoná sa hlásenie o chybovom ukončení činnosti robota
- *pošli chybový stav predchádzajúcemu* – chybový stav sa bude ošetrovať v kroku, ktorý predchádzal vykonaniu aktuálneho kroku
- *ignoruj a choď na ďalší krok* – chybový stav nie je ošetrený a vykonávanie robota pokračuje v ďalšom kroku
- *ignoruj a ukonči vykonávanie vetvy* – chybový stav nie je ošetrený, ale preskočia sa všetky kroky vo vetve, kde nastala chyba

### 2.1.6 Zhodnotenie nástroja

Medzi najväčšie výhody nástroja patrí jeho vizuálna podpora tvorby robota, ktorá umožňuje vytvárať postupnosti krokov a taktiež konfigurovať jednotlivé kroky iba pomocou vizuálnych možností nástroja. Prostredie tiež obsahuje vlastný integrovaný prehliadač, kde sa zobrazuje aktuálny dokument a používateľ môže vyznačiť údaje, ktoré má nástroj extrahovať. Prostredie tiež obsahuje ladiaci nástroj, kde je možné vyskúšať funkcionality robota a prezerať aktuálny stav robota v každom kroku. Nástroj má tiež výbornú podporu vzdialeného prístupu z iných aplikácií, umožňuje integráciu dát viacerých robotov, podporuje tiež prácu s cookies, formulármi alebo autentifikáciou.

Medzi nevýhody nástroja patrí slabá podpora výstupných objektov, kde je možné nadefinovať iba výstup v tvare XML súboru alebo zápis do určitých databáz, ktoré nástroj

podporuje. Kapow RoboSuite tiež nepodporuje učenie sa robota, kde by bolo možné na základe príkladov naučiť robota, ktoré dáta má extrahovať, čo by zvýšilo flexibilitu robota.

## 2.2 Lixto Visual Wrapper

Lixto je vizuálny interaktívny nástroj na vytváranie obalovačov webových stránok pod vedením človeka. Umožňuje automatickú extrakciu informácií z neštruktúrovaných webových stránok s použitím obalovačov a preklad získaného obsahu do štruktúrovaného XML dokumentu (produkujú sa tzv. XML Companions).

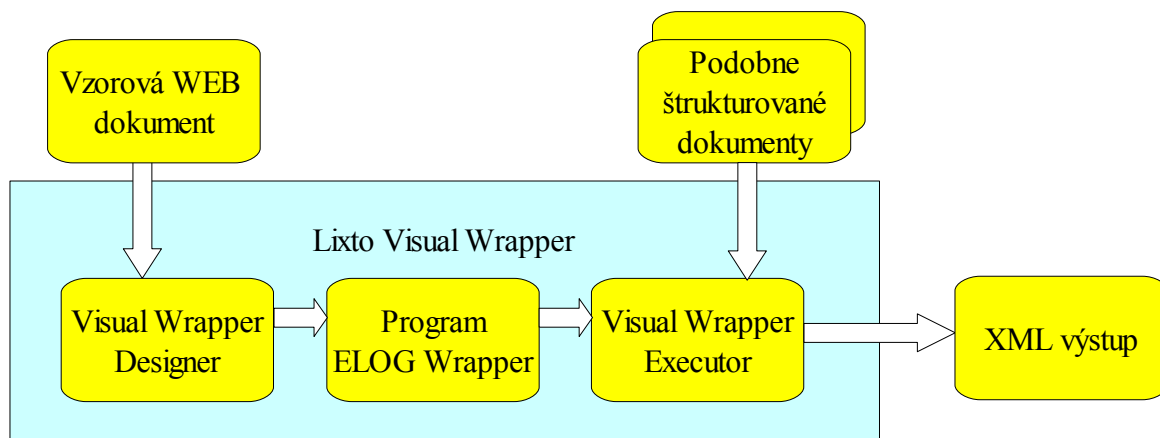
Systém pracuje na základe vizuálnej interakcie používateľa so spracovávaným dokumentom v prostredí Lixto Visual Wrapper. Používateľovi sa priebežne zobrazujú výsledky, ktoré by za súčasného stavu obalovača získal. Tak môže podľa potreby upravovať zadané požiadavky. Tvorba obalovača sa realizuje pomocou ovládacích prvkov, takže používateľ nemusí ovládať žiadny programovací jazyk. Pre úspešnú a efektívnu prácu so systémom by mal ovládať aspoň základy štruktúry HTML, prípadne prácu s reťazcami (regulárne výrazy a pod.).

V systéme Lixto Visual Wrapper nie je proces tvorby obalovača limitovaný len na jednu webovú stránku alebo na webové stránky so štruktúrou rovnakého typu. Počas definície obalovača sa môže operátor presunúť na ďalšie vzorové stránky a pokračovať v definícii obalovača tam.

### 2.2.1 Architektúra nástroja

Lixto Visual Wrapper sa skladá z dvoch hlavných častí: *Lixto Wrapper Designer* a *Lixto Wrapper Executor*.

- **Lixto Wrapper Designer** je nástroj na vytváranie a uchovávanie programu pre obalovač. Špecifikuje, ako budú programom extrahované údaje preložené do XML
- **Lixto Wrapper Executor** aplikuje program obalovača a XML prekladovú schému na spracovávané dokumenty



Obr. 2-3: Architektúra systému Lixto Visual Wrapper [4]

Na vzorovom webovom dokumente sa pomocou Visual Wrapper Designera definuje, ktoré

údaje sa majú extrahovať a stanoví sa aj ich namapovanie do výstupného XML dokumentu. Tieto definície VW Designer zapíše v podobe programu, ktorý sa používa pri samotnej extrakcii. Program je napísaný v jazyku ELOG.

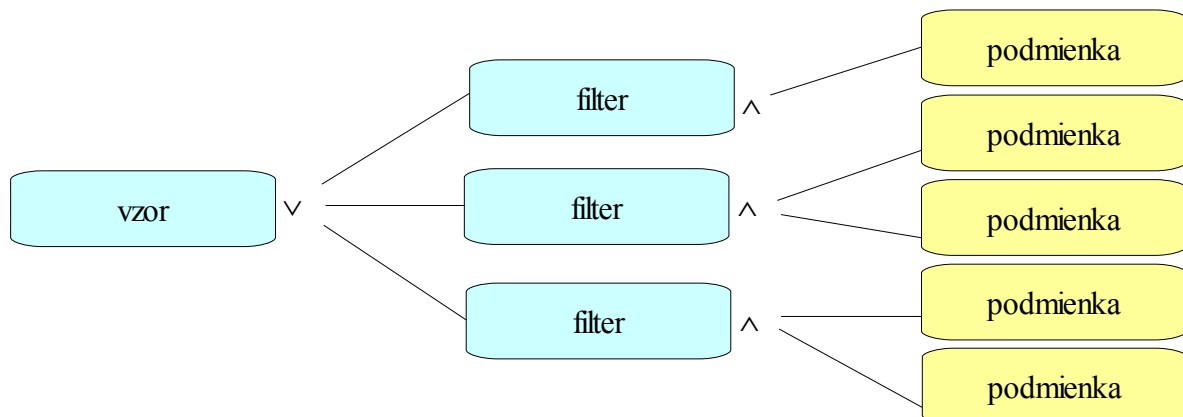
Visual Wrapper Executor potom na základe tohoto programu môže vykonávať extrakciu údajov z ďalších dokumentov. Vďaka dostatočnému množstvu metód a podmienok výberu údajov môže operátor vytvoriť robustný program, ktorý možno použiť na viaceré dokumenty vykazujúce znaky určitej štruktúrálnej podoby so vzorovým dokumentom. Architektúru systému a proces tvorby obalovača (programu pre extrakciu údajov z dokumentov) zobrazuje Obr. 2-3. Skutočná extrahovaná informácia závisí od vstupného dokumentu. Jeden program môže byť použitý na celú triedu podobne štruktúrovaných dokumentov.

## 2.2.2 Základné prvky – vzory, filtre, podmienky

Na tvorbu programov pre výber údajov z dokumentov sa používajú 3 základne prvky:

- vzory
- filtre
- podmienky

V architektúre Lixto *vzory* popisujú, ako extrahovať časti údajov z webových stránok. Vzor sa skladá z *filtrův*. Filtre definujú časť stránky, z ktorej sa majú získať údaje. Interpretujú sa disjunktívne, t.j. stačí, aby sa dal aplikovať jeden z filtrov. Ak filtre vyberú aj neželané údaje, na obmedzenie tohoto výberu sa používajú *podmienky*. Podmienky sa interpretujú konjunktívne, t.j. musia byť splnené všetky, aby došlo k výberu údajov, ktoré by bez aplikácie podmienok vybral filter pre daný vzor. Tieto vzťahy schématicky zobrazuje Obr. 2-4.



Obr. 2-4: Vzťahy medzi vzormi, filterami a podmienkami [4]

Proces tvorby Lixto obalovača pozostáva z vytvárania vzorov, následného pridelenia filtrov k týmto vzorom, testovania vybraných inštancií a prípadného nastavenia podmienok na obmedzenie výberu filtrov. Výsledný obalovač sa interne zaznamená v jazyku *Elog*. Na podporu úprav výstupu poskytuje systém používateľovi nástroj XML Tool.

## Vzory

Vzory sú základnými konštrukciami obalovača a definujú jeho hierarchickú štruktúru. Každý vzor je postupne mapovaný na XML element a každý potomok vzoru v stromovej hierarchii vzorov je mapovaný do XML elementov vnhniezených do rodičovského.

Opisujú, ako extrahovať časti údajov z webových stránok. Každý vzor pozostáva z filtrov, ktoré rozhodujú, čo sa má extrahovať.

Vzory sa vytvárajú interaktívne: Operátor vytvorí nový vzor a označí želaný príklad na webovom dokumente. Systém zovšeobecní výber a vráti operátorovi všetky inštancie vzoru vybrané po zovšeobecnení. Odpoveď systému je čisto vizuálna – každá inštancia vzoru je zvýraznená. Takýto systém odozvy systému umožňuje predchádzať princípu: Vytvorenie a spustenie obalovača a následná oprava chýb. Operátor môže potom zúžiť výber inšancií pomocou obmedzení (zavedenie podmienok) a následne môže pridávať ďalšie inštancie, ak aktuálny výber nezahŕňa všetky, ktoré si želal.

### Kategórie vzorov

V závislosti na type požadovanej informácie si treba zvoliť jednu z troch kategórií vzorov:

- *strom* (tree) – Vzor strom slúži na extrakciu časti dokumentu v závislosti na HTML elementy alebo listy elementov.
- *reťazec* (string) – Tento vzor slúži na extrakciu textových reťazcov z viditeľných aj neviditeľných častí dokumentu. Môže to byť napríklad e-mailová adresa, poštové smerové číslo alebo nejaký atribút, napríklad meno obrázku.
- *dokument* (document) – Vzor dokument slúži na extrakciu celých webových stránok a používa sa na navigáciu na odkazmi pripojené stránky. Základom každého obalovača je koreňový vzor (`rootPattern`). Je to dokumentový vzor, na ktorý sa môžu pripájať aj ďalšie dokumentové vzory na extrakciu informácií z ďalších prepojených stránok.

Každý vzor má svoj rodičovský vzor (*parent pattern*). Vzory a filtre obsahujú informácie o ich rodičoch.

## Filtre

Logická organizácia extrakčných vzorov je nasledovná: Každý extrakčný vzor má meno a obsahuje niekoľko filtrov. Každý filter poskytuje alternatívnu definíciu údajov, ktoré sa majú extrahovať a asociovať s daným vzorom. Vzory musia mať aspoň 1 filter, aby mohli extrahovať údaje. No, môžu mať viacero ďalších vzorov ako svoje deti.

Filtre vymedzujú, aké informácie sa majú extrahovať z kontextu príslušného vzoru, t.j. ktoré časti dokumentu majú byť extrahované. Upravujú výber v stromovej štruktúre HTML dokumentu, alebo prístupujú k dokumentu ako k reťazcu a podľa toho sú aj definované podmienky výberu. Filtre môžu byť obmedzené podmienkami.

Ak je špecifikovaný viac ako jeden filter, musí byť splnený aspoň jeden, aby bola inštancia vybraná. Typ filtra je definovaný typom rodičovského vzoru, ku ktorému filter patrí:

- Filter stromu (*tree filter*)
- Filter reťazca (*string filter*)
- Filter dokumentu (*document filter*) – slúži na pospájanie informácií z viacerých HTML stránok do jedného XML dokumentu. Používajú sa v dvoch prípadoch:
  1. Detailné informácie sa nachádzajú na stránkach, na ktoré sú na spracovávanej stránke odkazy. Tieto stránky s detailnými informáciami majú obyčajne inú štruktúru než pôvodná stránka a musia byť analyzované inými vzormi.
  2. Dlhé zoznamy bývajú často rozdelené na viacero stránok, na ktoré sú prepojené pomocou odkazov (*Next link*). Tieto stránky majú obyčajne rovnakú štruktúru ako hlavná stránka a na ich spracovanie sa môže použiť rekurzia – koreňovému vzoru sa pridá *document filter* sledujúci *Next link*.
  - ◆ Použitím filtra dokumentu môžu byť vzory zahŕňajúce informácie z ďalších stránok vytvorené veľmi ľahko. Nie je potrebné robiť žiadnu opakovanú definíciu vzoru na ďalších stránkach a nie je na to potrebné ani procedurálne programovanie. Vzory môžu byť znovu použité pomocou **rekurzíe**.

## Podmienky

Ak test filtra vyberie príliš veľa inštancií, dá sa tento výber obmedziť použitím podmienok. Slúžia na spresnenie výberu údajov. Lixto používa nasledovné typy podmienok:

- kontextové podmienky – špecifikujú, aký element sa musí alebo nesmie nachádzať pred alebo za týmto vzorom
  - stromová kontextová podmienka – vychádza z HTML štruktúry dokumentu
  - reťazcová kontextová podmienka – s kontextom pracuje ako s reťazcom
- interné podmienky – špecifikujú, aký element sa musí alebo nesmie nachádzať v tomto vzore (stromová, reťazcová)
- podmienka referencie na vzor – používa sa na referenciu na už existujúci súrodenecký vzor v strome vzorov
- podmienka existencie n-tého dieťaťa – používa sa na výber dieťaťa z detí HTML elementu alebo regiónu. Špecifikuje, že jedno z detí musí existovať
- podmienky rozsahu – špecifikovanie rozsahu je jednou z možností pre obmedzenie filtra. Definuje sa, ktoré inštalácie v rozsahu filtra majú byť extrahované.

Všetky podmienky poskytujú možnosť použiť regulárne výrazy (používa sa typ výrazov ako v jazyku Perl verzia 5<sup>1</sup>).

Navyše môže operátor vyžadovať zhodu inštalácie s výrazom z preddefinovanej ontologickej triedy (uloženej v nejakej databáze) a porovnanie inštancií vzoru (napríklad: inštalácia musí byť číslo menšie ako 7).

Je na operátorovi, ktoré podmienky si zvolí na výber želaných údajov. Obyčajne sám vie, ktoré podmienky sú z hľadiska správneho výberu danej inštalácie najlepšie. Systém asistuje

<sup>1</sup> Perl, výnimočne dynamický programovací jazyk, <http://www.perl.org/>



operátorovi pri výbere týchto podmienok.

### 2.2.3 Elog

Na zadefinovanie výberu sa používa extrakčný jazyk Elog. Tento jazyk bol vyvinutý špeciálne na extrakciu údajov z webu. Je to pružný, intuitívny a ľahko rozširiteľný jazyk podobný jazyku Datalog<sup>2</sup>. Má jasne definovanú sémantiku. Extrahuje informácie na základe okolitých značiek, samotného obsahu údajov, HTML atribútov, poradí výskytu, sémantických a syntaktických konceptov.

V extrakčnom programe Elog sú vzory hierarchicky organizované. Pri vytváraní ďalších vzorov extrakcie sa postupuje výhradne v kontexte aktuálneho vzoru. Táto stromová metóda vytvárania vzorov napomáha vzniku robustných programov obalovačov, ktoré potom môžu správne pracovať aj nad dokumentami podobnými vzorovému. Hoci sú vzory hierarchicky organizované, nemusia tvoriť stromovú štruktúru a umožňujú tak operátorovi nahradiť implicitnú štruktúru prítomnú vo webovej stránke.

Dôležité je, že návrhár obalovača (operátor) sa nepotrebuje jazyk Elog učiť, dokonca ho nemusí ani poznať. Všetky vlastnosti sú dostupné cez vizuálne rozhranie.

### Sémantický koncept

Pri tvorbe obalovačov môže návrhár použiť aj sémantické koncepty. Tieto koncepty sú založené na ontológiách. Na ich načítanie a prácu s nimi slúži editor – Semantic Concept Editor. Samotné sémantické koncepty sú uložené v SQL databáze.

### Syntaktický koncept

Lixto Visual Wrapper umožňuje použitie syntaktických konceptov na výber určitej syntaxe z webových stránok. Napríklad dátumy, čas alebo čísla sú zapísané v špeciálnom formáte v závislosti od regiónu. Tento formát môže byť uložený ako syntaktický koncept a použitý vo filtri na výber príslušných inštancií z webovej stránky.

### 2.2.4 XML Tool

Dôležitou časťou systému Lixto Visual Wrapper je nástroj XML Tool. Tento nástroj poskytuje viaceré možnosti práce s úpravou obsahu výstupu do výsledného XML dokumentu. Medzi hlavné funkcie, ktoré nástroj poskytuje, patrí:

- **Transformácia extrahovaných vzorov do výsledného XML dokumentu**
  - Používateľ (operátor) môže určiť, ktoré z extrahovaných vzorov budú zaznamenané vo výslednom XML dokumente. Ďalej môže určiť poradie, v akom sa budú jednotlivé elementy v dokumente vyskytovať a tiež aj namapovať vybrané vzory s nastaveným menom na XML tagy s iným menom.
- **Prezeranie štruktúry obsahu výstupného dokumentu**

---

<sup>2</sup> Datalog, dopytovací jazyk používaný v deduktívnych databázach, v podstate podmnožina jazyka Prolog

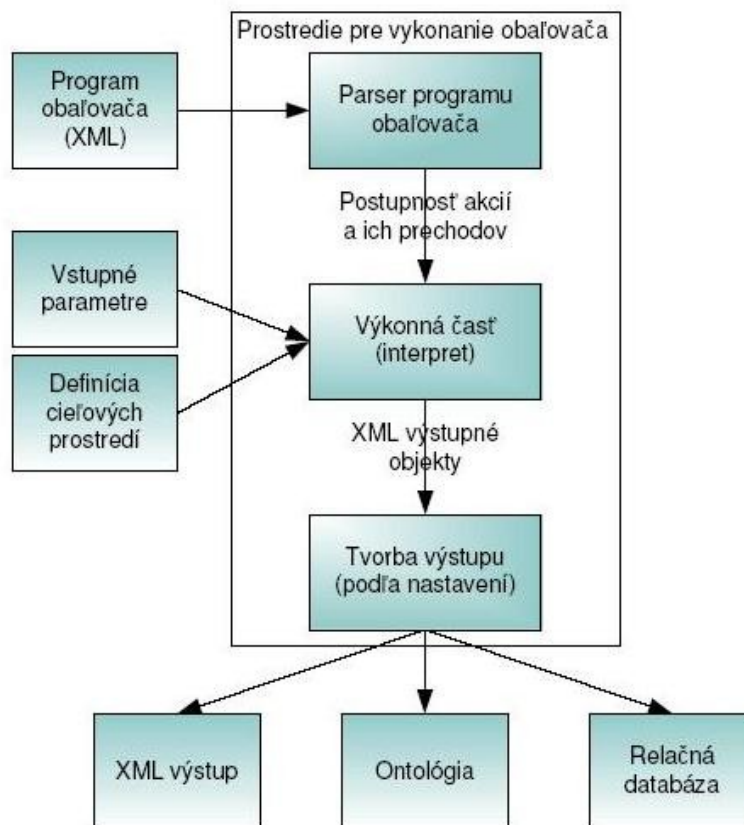
- Nástroj umožňuje prezeranie XML štruktúry výsledného dokumentu pre aktuálny obalovač.
- **Verifikácia**
  - XML Tool poskytuje množstvo ďalších funkcií na stanovenie štruktúry aj obsahu cieľového dokumentu. Patria sem napríklad aj mnohé kontrolné mechanizmy, napr. stanovenie minimálneho alebo maximálneho počtu výskytov daného elementu, hlásenie v prípade zmeny ich počtu a pod.

## 2.3 Aplikačný rámec Wrapper Suite

Aplikačný rámec Wrapper Suite predstavuje prostredie na tvorbu predprogramovaných obalovačov. Bol vytvorený ako tímový projekt v akademickom roku 2005/2006. Náš nástroj si kladie za cieľ jeho rozšírenie.

### 2.3.1 Architektúra

Na Obr. 2-5 je znázornená základná architektúra systému.



Obr. 2-5: Základná architektúra Wrapper Suite [5]

Definícia obalovača je obsiahnutá v programe obalovača, kde sú opísané jednotlivé časti obalovača. Úlohou parsera programu obalovača je transformovať program obalovača vo forme XML do vnútornej reprezentácie. Súčasťou vnútornej reprezentácie obalovača je vytvorenie akcií,

ich zreťazenie pomocou definovaných prechodov, registrovanie štartovacej akcie a tiež aj určenie HTTP klienta pre komunikáciu so serverom a HTML parsera pre získanie stromovej štruktúry (DOM) určeného dokumentu. Interpret zabezpečuje vykonávanie definovaných akcií obalovača v poradí ako je dané zoznamom prechodov (v programe obalovača). Vstupné premenné predstavujú informácie potrebné na vykonanie akcií obalovača (napr. prihlasovacie údaje) a sú uložené v kontextových premenných. Definícia cieľových prostredí určuje špecifikáciu výstupných objektov obalovača. Modul tvorby výstupu vykonáva zápis extrahovaných dát do výstupných objektov.

### 2.3.2 Program obalovača

Program obalovača sa skladá z:

- identifikácie akcií – definuje typ akcie (navigačná, extrakčná, iteračná, pomocná) a jej atribúty
- definovania prechodov medzi akciami
- premenných – vstupné premenné obalovača

Príklad programu obalovača zapísaného vo forme XML sa nachádza na Obr. 2-6. Akcie sú definované v elemente *Actions*, kde pre každú akciu je definované jedinečné meno (*name*) a typ (*type*) akcie. Každá akcia je špecifikovaná definovaním atribútov akcie v elemente *Attribute*, každý atribút obsahuje jedinečné meno (*name*), definovaný typ (*type*) a hodnotu atribútu. Prechody medzi akciami sú definované v elemente *Transitions*, kde pre každý prechod (*Transition*) je určená zdrojová (*src*) a cieľová (*dst*) akcia. Premenné sú definované v elemente *Variables*. Pre každú premennú (*Property*) je určený jedinečný názov (*name*), definovaný type (*type*) a hodnota (*value*) premennej.

```

<Wrapper>
  <Actions>
    <Action name="" id="" type="">
      <Attribute name="" type="">value</Attribute>
      . . .
      <Attribute name="" type="">value</Attribute>
    </Action>
    . . .
  </Actions>
  <Transitions>
    <Transition src="" dst=""/>
    . . .
    <Transition src="" dst=""/>
  </Transitions>
  <Variables>
    <Property name="" type="">value</Property>
    . . .
    <Property name="" type="">value</Property>
  </Variables>
</Wrapper>

```

Obr. 2-6: Príklad programu obalovača [5]

Akcie obalovača pracujú nad globálnym kontextom obalovača (menia ho, využívajú jeho

údaje). Globálny kontext pozostáva z:

- dokumentov – mapa stromovej štruktúry dokumentov (DOM), nad ktorými daná akcia pracuje
- výstupných objektov – štruktúra objektov, ktoré sú získané procesom extrakcie dát
- premenných – parametrizácia akcií počas behu programu. Premenné môžu v sebe obsahovať aj iné premenné
- cookies – zoznam cookies získaných pri navigácií
- autentifikačných dát

Akcie, ktoré pracujú s dátami v dokumente obsahujú lokalizáciu týchto dát v stromovej štruktúre. Na lokalizáciu elementov v stromovej štruktúre dokumentu je určený `TagLocator`, ktorý sa skladá z 3 častí:

- `inDocument` – určuje dokument v kontexte, v ktorom sa bude hľadať,
- XPath výraz – stanovenie cesty v stromovej štruktúre dokumentu,
- regulárny výraz – časť dokumentu získanú aplikovaním XPath výrazu možno filtrovať pomocou regulárneho výrazu.

## Navigačné akcie

Akcie, ktoré zabezpečujú prechod na určenú stránku a jej uloženie do globálneho kontextu. Medzi navigačné akcie patrí:

- `LoadPage` – načítanie stránky (určenej pomocou url adresy) do dokumentu
- `FollowLink` – nájdenie odkazu na stránke a prechod na dokument, kde odkaz smeruje. Element odkazu je v dokumente určený pomocou lokátora (`TagLocator`)
- `SubmitForm` – odoslanie formulára určenou metódou (GET, POST), do formulára sú odoslané hodnoty určených vstupných parametrov

## Extrakčné akcie

Akcie, ktorých cieľom je získanie požadovaných dát zo stránky:

- `SelectSubDocument` – výber a uloženie do kontextu nového poddokumentu,
- `ExtractData` – extrahovanie a konverzia dát z elementu (určeného lokátorom) do objektu alebo premennej v kontexte,
- `WriteObject` – výstupný objekt z kontextu identifikovaný svojim názvom sa pošle modulu tvorba výstupu.

## Iteračné akcie

Akcie, ktoré zabezpečujú opakované vykonávanie určitých operácií:

- `ForEachTag` – pre každý nájdený podstrom (určený lokátorom) sa opakovane vykonajú nasledujúce akcie

- `DoWhileLinkExists` – opakovanie nasledujúcich akcií, pokiaľ existuje určený odkaz

## Pomocné akcie

Pomocné akcie sú určené na prácu so zásobníkom kontextov a na vykonávanie všetkých vetiev akcií obalovača.

### 2.3.3 Výkonná časť obalovača

Obalovač je určený jednotlivými špecifikovanými akciami a ich zret'azením vykonávania. Vnútoraná reprezentácia obalovača sa vytvorí špecifikovaním akcií obalovača, odkazom na prvú akciu, vytvorením kontextu obalovača, zoznamu zapisovačov do cieľových prostredí, HTML parsera dokumentu a tiež HTTP klienta pre komunikáciu so serverom. Cieľom výkonnej časti obalovača je spustenie obalovača vytvoreného vo forme vnútornej reprezentácie. Spustenie programu obalovača je zabezpečené spustením štartovacej akcie, ktorá po ukončení svojej činnosti spustí akciu, ktorá nasleduje za danou akciou.

## Spracovanie chýb

V akciách obalovača možno ošetrovať chybové stavy (napr. nenašli sa dáta v dokumente) nasledovným spôsobom:

- `StopThrowErrorHandler` – v prípade vzniku chyby sa ukončí vykonávanie obalovača
- `ReturnBackErrorHandler` – v prípade vzniku chyby sa nepokračuje vykonávaním nasledovníkov, ale chybový stav je ošetrovaný v predchodcovi
- `IgnoreContinueErrorHandler` – chybový stav sa ignoruje a pokračuje sa ďalej
- `ExecuteCommandErrorHandler` – v prípade vzniku chyby sa vykoná definovaný príkaz (notifikácia mailom o vzniknutej chybe)

### 2.3.4 Načítanie obalovača

Trieda `WrapperLoader` slúži na načítanie programu obalovača z XML reprezentácie do vnútornej formy skladajúcej sa z navzájom prepojených akcií a kontextu. Na základe typu jednotlivých akcií uvedeného v atribúte *type* elementu *Action* sa volá metóda príslušnej triedy na naplnenie jej parametrov. Počas vytvárania a naplnenia akcií dochádza zároveň k registrácii akcií k programu obalovača, k inštancii triedy `WrapperProgram`. Ako prvá sa teda vytvorí trieda `LoadPage` a následne sa volajú jej metódy `setUri` a `setAsDocument` na nastavenie jej základných parametrov.

Vytváranie nových akcií a volanie ich metód je realizované pomocou reflexie, čo zabezpečuje flexibilitu v prípade, že budeme potrebovať implementovať nové akcie alebo meniť už existujúce. Triedu `WrapperLoader` v týchto prípadoch nie je vôbec potrebné meniť. Potrebná je zmena metódy na naplnenie údajov a je tiež nutné tieto zmeny zohľadniť aj v XML schéme, ktorou je validovaná vstupná XML reprezentácia. Na jednoduchší prístup k funkciám Java

Reflections API slúži trieda `Reflections`, ktorá poskytuje metódy na vytváranie nových objektov a na volanie funkcií zadaním ich názvu a parametrov.

Vstupný XML súbor obsahuje časť nazvanú *Transitions*, v ktorej sú definované prechody medzi akciami. Na ich základe dochádza k zreťazeniu akcii vo vektore `nextActions`, ktorý obsahuje potomkov každej akcie.

### 2.3.5 Pomocné triedy

Na serializáciu DOM dokumentov a elementov do XML súboru alebo textového reťazca sú implementované pomocné triedy, využívajúce `XMLSerializer`. Ide o triedu z knižnice projektu Apache XML Project Xerces<sup>3</sup>.

- `DocumentFileWriter` - trieda implementujúca zápis DOM dokumentu do XML súboru
- `DocumentStringWriter` - trieda implementujúca zápis DOM dokumentu alebo jeho elementu do textového reťazca obsahujúceho XML kód
- `OutputObjectStringSerializer` - trieda implementujúca zápis výstupného objektu, ktorého obsah je reprezentovaný DOM elementom, do textového reťazca obsahujúceho XML kód

### 2.3.6 Architektúra zápisu do cieľových prostredí

Z hľadiska implementácie je fáza tvorby výstupu spojená s fázou interpretácie obalovača. Každý program obalovača (trieda `WrapperProgram`) obsahuje okrem iného aj zoznam zapisovačov do cieľových prostredí. Je teda možné vykonávať zápis do viacerých cieľových prostredí súčasne a zároveň to dáva priestor na implementáciu a začlenenie vlastných typov zapisovačov.

Naplnené výstupné objekty sa priebežne zapisujú do vyrovnávacej pamäte a na konci behu programu sa pre každý zapisovač asociovaný s obalovačom vykoná fyzický zápis získaných údajov (`flush`). Aplikácia pracuje s jednotným modelom výstupného objektu a každý zapisovač musí vedieť tento výstupný objekt transformovať do jemu akceptovateľnej podoby prostredníctvom mapovacích tried (`OutputObjectToJobOfferMapper`, `OutputObjectToRDFMapper`).

Obalovač ponúka rozhranie `OutputWriter` reprezentujúce všeobecný zapisovač získaných údajov do bližšie nešpecifikovaného cieľového prostredia. Z hľadiska údajov sa pracuje na úrovni naplnených výstupných objektov, ktoré sú predávané jeho metódam. Zapisovače do konkrétneho výstupného prostredia dedia jeho hlavné metódy rozhrania:

- `write` – vykoná priebežný zápis naplneného výstupného objektu do interného zoznamu objektov určitého typu v pamäti
- `flush` – vykoná fyzický zápis údajov do daného cieľového prostredia
- `cleanUp` – vykoná vyčistenie vnútornej vyrovnávacej pamäti po realizácii fyzického zápisu

---

<sup>3</sup> Apache XML Project Xerces, <http://xerces.apache.org/>.

údajov

## Zápis na konzolu

O zápis na konzolu sa stará trieda `ConsoleOutputWriter` rozširujúca rozhranie `OutputWriter`. Trieda má len jeden atribút `name` na jednoznačnú identifikáciu zapisovača.

- `write` – serializovaný výstupný objekt na textový reťazec zapisuje na konzolový výstup
- `flush` – výpis konzolovej vyrovnávacej pamäte na konzolu
- `cleanUp` – nerealizuje žiadnu funkcionálnosť

## Zápis do XML

Na zápis do XML súboru je určená trieda `XMLOutputWriter` obsahujúca nasledovné atribúty:

- `name` - identifikátor objektu zapisovača
- `objectList` – interný zoznam výstupných objektov
- `fileName` – názov výstupného XML súboru
- `rootElementName` – názov koreňového uzla v DOM dokumente

a základné metódy:

- `write` – výstupné objekty pridáva do zoznamu (`objectList`)
- `flush` – vytvorí DOM dokument s koreňovým uzlom `rootElementName`, pre všetky výstupné objekty v zozname sa do dokumentu importuje uzol s príslušnou hierarchickou štruktúrou a nastáva serializácia súboru
- `cleanUp` – maže položky zoznamu `objectList`

## Zápis do ontológie

Zápis do ontológie je možné realizovať na úrovni naplnených RDF/XML reťazcov, čo je najnižšia úroveň reprezentácie údajov, závislá od štruktúry ontológie, ale nezávislá od domény, alebo na úrovni naplnených komponentov JavaBeans typu *JobOffer*<sup>4</sup> závislých na doméne, ale nezávislých na štruktúre ontológie.

## Zapisovač na úrovni RDF/XML

Zápis do ontológie realizuje trieda `RdfTemplateOntologyOutputWriter` prostredníctvom Sesame API rozhrania ontologického úložiska Sesame<sup>5</sup>. Obsahuje nasledovné atribúty:

- `rdfXmlCodes` – zoznam RDF/XML kódov objektov

---

4 *JobOffer* JavaBeans boli dodané tímom číslo 8, ktorý ich vytvoril na základe ontológie projektu NAZOU, <http://www2.dcs.elf.stuba.sk/TeamProject/2005/team08/>.

5 Sesame, <http://www.openrdf.org/index.jsp>

- `outObjToRdfMapping` – „hash“ mapa získaná mapovacou triedou `OutputObjectRDFMapper`, ktorá umožňuje k výstupnému objektu nájsť odpovedajúcu parametrizovanú RDF šablónu
- `baseUri` – základ URI pre ontologické úložisko
- `sesameRepository` – rozhranie umožňujúce prácu s ontologickým úložiskom
- `sesameService` – administrácia úložiska

Okrem zdedených základných metód implementuje trieda aj `connect` a `disconnect` na pripojenie a odpojenie sa od ontologického úložiska.

### Zapisovač na úrovni JavaBeans typu JobOffer

Trieda `JobOfferOntologyOutputWriter` plní funkciu ukladania výstupných objektov do ontologického úložiska a to špecificky pomocou rozhrania `SesameRepositoryAccess` dodaného tímom 8. Toto rozhranie umožňuje na základe špeciálneho XML súboru transformovať určitý komponent JavaBeans (napr. typu *JobOffer*) na zodpovedajúci Sesame graf RDF trojíc (subjekt, predikát, objekt), ktorý sa následne vkladá do ontológie. Samotná trieda má tri významné atribúty:

- `sesameRepositoryAccess` – rozhranie pre zápis do ontológie
- `graphBeanTransformerFile` – meno transformačného súboru
- `jobOffers` – zoznam vytvorených komponentov JavaBeans

Výstupný objekt sa transformuje na JavaBeans typu `JobOffer` mapovacou triedou `OutputObjectToJobOfferMapper`. Tak isto okrem zdedených základných metód implementuje trieda aj funkcie `connect` a `disconnect` na pripojenie a odpojenie sa od ontologického úložiska.



## 3 Opis riešenia

Predmetom tejto kapitoly je opis nami navrhovaného riešenia. V časti 3.1 si vytýčíme ciele produktu a vlastnosti, ktorými bude produkt disponovať a časť 3.2 prinesie zoznam funkcionálnych požiadaviek, ktoré sú na systém kladené.

### 3.1 Ciele a vlastnosti produktu

Úlohou vytváraného softvérového systému je v prvom rade tvorba obalovačov v prostredí webu. Obaľovanie dokumentov je pomerne komplikovaná činnosť, z čoho vyplýva aj komplikovaná tvorba týchto nástrojov. Vytvorenie jednoduchého obalovača si vyžaduje pomerne rozsiahle znalosti štruktúry obalovaných dokumentov, nástrojov na dopytovanie jednotlivých častí týchto dokumentov (napr. XPath v prípade XML dokumentov) a v neposlednom rade aj samotnej činnosti obalovača (interpretácia programu obalovača, stavové automaty). Na druhej strane je zrejmé, že obalované dokumenty sa často menia. Malou zmenou dokumentu (napr. pridanie záhlavia) sa obalovač stáva nepoužiteľný a vracia nesprávne výsledky. Najjednoduchším výstupom obalovača je XML súbor, ktorý obsahuje zozbierané výsledky. Keďže však obaľovanie môže a bežne aj produkuje veľké množstvá výsledkov je potrebné, aby bolo možné výstupy generovať do iného úložiska, napríklad do databázy alebo ontológie. Na základe týchto poznatkov môžeme hlavné ciele produktu zhrnúť nasledujúcimi bodmi:

- Zameranie sa na používateľa. Interaktívna tvorba obalovačov.
- Systém pomáha používateľovi pri tvorbe obalovačov.
- Tvorba obalovačov bez nutnej znalosti obalovaných dokumentov.
- Tvorba obalovačov bez rozsiahlych znalostí interpretácie jazyka obalovača. Zvládnutie jazyka obalovača bez akéhokoľvek štúdia.
- Jednoduchá zmena obalovača, ak boli obalované dokumenty zmenené iba preusporiadaním obalovaných elementov, prípadne pridaním nových neobaľovaných elementov.
- Podpora rôznych výstupov obalovačov.
- Rovnomerné zaťaženie zdrojov. Množstvo výsledkov, ktoré obalovač produkuje musí byť zapisované do úložiska postupne, aby nedošlo k zahlteniu pamäti počítača, na ktorom bol obalovač spustený.
- Doteraz boli spomínané iba funkcionálne ciele systému. Okrem týchto cieľov musí mať vytváraný systém aj nasledujúce nefunkcionálne vlastnosti:
  - Podpora rôznych platforiem. Systém musí byť čo najviac všeobecný.
  - Zachovanie bezpečnosti pri použití citlivých údajov. Niektoré dokumenty vyžadujú autentifikačné údaje.

Cieľom projektu je tiež dosiahnuť architektúru rozšíriteľnú o ďalšie typy reprezentácie dokumentov flexibilitu výmeny vzorov za iné typy pomocou tzv. adaptácie.

## 3.2 Prehľad produktu

Táto kapitola ilustruje očakávané funkcionálne vlastnosti produktu uvedené v predchádzajúcej časti na zozname požiadaviek (Tab. 3-1). Každá požiadavka je definovaná unikátnym identifikátorom, kontextom a oblasťou systému, na ktorú sa vzťahuje, textom požiadavky a vysvetlením požiadavky. Takýto prehľad produktu má oproti častejšie používaným scenárom použitia jednu podstatnú výhodu. Vyznačuje sa vyššou granularitou nastavenou tak, aby sa každá požiadavka vzťahovala iba na jednu oblasť v rámci kontextu použitia. Požiadavky nám naznačujú hrubý pohľad na architektúru systému prostredníctvom kontextu a na prípady použitia prostredníctvom oblastí použitia.

Tab. 3-1: Požiadavky na produkt

ID	Kontext	Oblasť	Text/Vysvetlenie
P1	Jadro	Akcie	V rámci navigácie sa musí vedieť obalovač pohybovať po stránkach, pracovať s formulármi a cookies, pracovať nad protokolom HTTP a HTTPS.
			-
P2	Jadro	Akcie	Extrakčné akcie získavajú dáta z akejkoľvek časti dokumentu.
			Môže to byť celý element, iba jeho časť, obrázok alebo odkaz na iný typ súboru.
P3	Jadro	Akcie	K extrakčnej akcii je možné definovať výstupný objekt a mapovanie na extrahované dáta.
			-
P4	Jadro	Akcie	Navigačnej formulárovej akcii je možné definovať mapovanie formulárových elementov na vstupné dáta.
			-
P5	Jadro	Vzory	Akcie pracujú nad elementami dokumentu, ktoré vzniknú aplikovaním vzoru na dokument.
			Vzor je súbor filtrov nad dokumentom.
P6	Jadro	Vzory	Nový vzor je možné vytvoriť v dvoch reprezentáciách: XPath a regulárne výrazy (v budúcnosti možno aj viac).
			Nie je možné zmeniť reprezentáciu už vytvoreného vzoru.
P7	Tvorba obalovača	-	Obalovač pracuje nad určenou doménou.
			Doména je dôležitá z hľadiska integrácie výsledkov.
P8	Tvorba obalovača	Učenie	Systém pomáha pri tvorbe vzorov. Používateľ zadáva pozitívne a negatívne príklady.
			Systém sa pokúša zovšeobecňovať (generalizovať) zadávané príklady. Prebieha proces učenia.
P9	Tvorba obalovača	Učenie	Príklady v procese učenia je možné zadávať interaktívne.
			Interaktívne zadávanie: napr. grafický výber pozitívnych alebo negatívnych elementov v HTML stránke.
P10	Tvorba obalovača	Úprava obalovača	Vytvorené obalovače je možné upravovať.
			Systém poskytuje možnosť upravenia stromu akcií (pridávanie, rušenie, presúvanie akcií), možnosť zmeny subdokumentu,

			parametrov akcie.
P11	Tvorba obalovača	Úprava obalovača	<p>Pri úprave akcie musí systém upozorniť používateľa na akcie, ktoré mohli byť zmenou ovplyvnené a označí obalovač ako konfliktný.</p> <p>Systém poskytuje používateľovi možnosť označiť obalovač ako nekonfliktný. Používateľ si prešiel, opravil konflikty a potvrdí, že je všetko v poriadku.</p>
P12	Tvorba obalovača	Úprava obalovača	<p>Systém na tvorbu obalovačov dovoľuje manuálnu zmenu naučeného vzoru (filtrov).</p> <p>Systém poskytuje možnosť ako naučený vzor upraviť manuálne (napr. prepísanie XPath výrazu).</p>
P13	Tvorba obalovača	Úprava obalovača/ Učenie	<p>Vzor je možné doučiť.</p> <p>Ak mal vzor manuálne zmenené filtre, sú nahradené poslednými naučenými, až potom sa začne proces doučenia.</p>
P14	Tvorba obalovača	Úprava obalovača/ Učenie	<p>Naučené príklady je možné editovať. Ak používateľ odstráni naučený príklad vzoru, filtre sa dostanú do stavu v akom boli pred naučením tohto príkladu.</p> <p>Systém si pamätá históriu učenia. Ku každej zadanej množine príkladov si systém pamätá filtre, ktoré vznikli generalizáciou týchto príkladov.</p>
P15	Činnosť obalovača	Spúšťanie obalovača	<p>Ak je obalovač konfliktný (bola vykonaná riziková akcia P11) systém na to pred spustením obalovača upozorní používateľa.</p> <p>-</p>

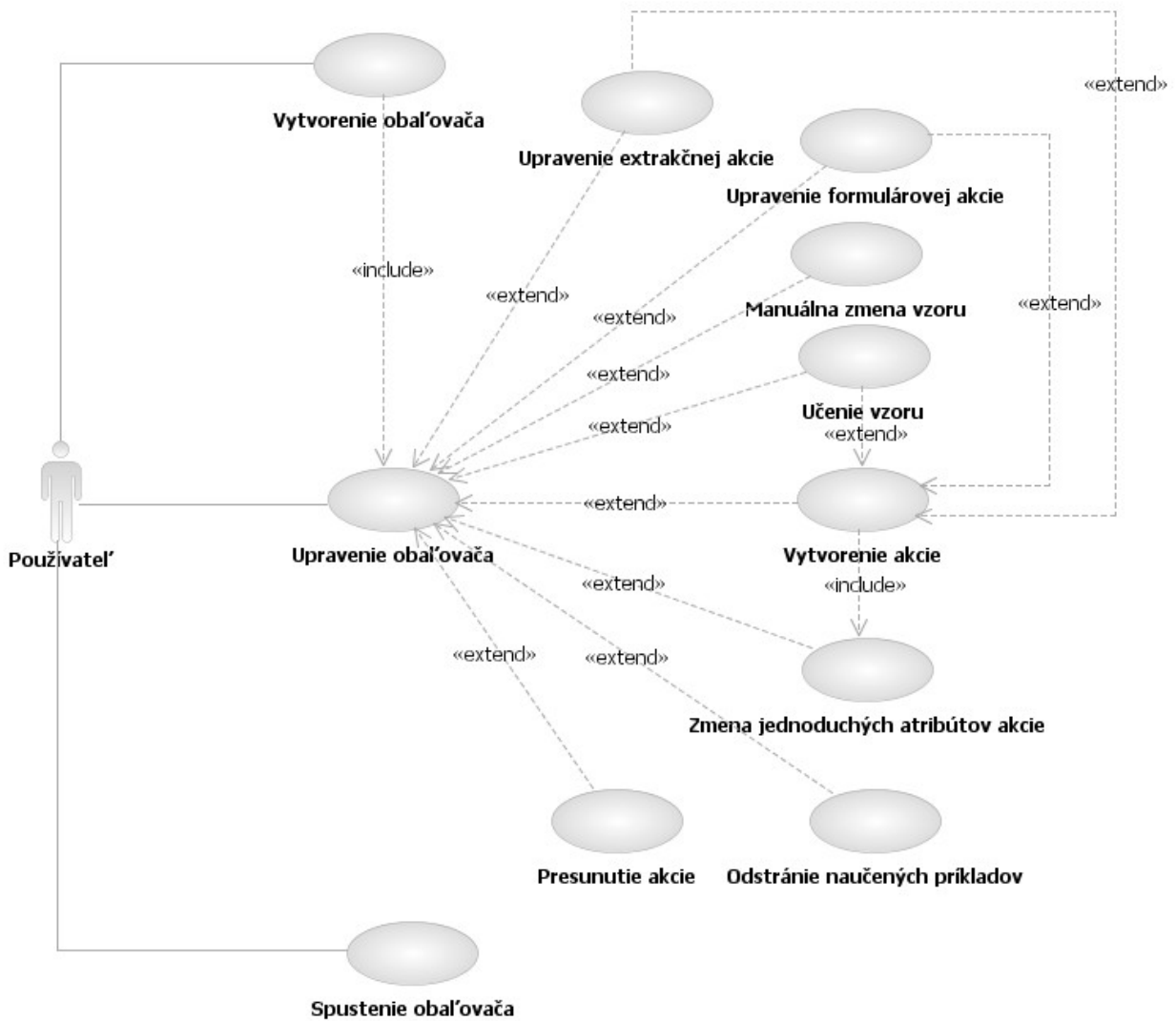


## 4 Špecifikácia obal'ovača

V tejto kapitole opíšeme špecifikáciu systému pomocou špecifikácie prípadov použitia, ktoré predstavujú konkrétnu realizáciu funkcionálnych požiadaviek na systém uvedených v časti 3.2. Vytváraný systém na tvorbu obal'ovačov je rozšírením existujúceho systému, preto sú vytvorené len prípady použitia, ktoré pokrývajú požiadavky nerealizované existujúcim systémom.

### 4.1 Prípady použitia

Pri špecifikácii prípadov použitia sme podobne ako pri definovaní požiadaviek vychádzali z už existujúceho systému. V diagrame prípadov použitia (Obr. 4-1) sa nachádzajú iba nové prípady použitia systému alebo prípady použitia, ktoré sú v novom systéme prepracované na základe nových, prípadne pozmenených požiadaviek.



Obr. 4-1: Diagram prípadov použitia

## 4.2 Popis prípadov použitia

Každý prípad použitia je opísaný tabuľkou. Jednotlivé prípady klasifikujeme prioritou podľa Tab. 4-1.

Tab. 4-1: Priority prípadov použitia

Priorita	Význam
1	Vysoká
2	Stredná
3	Nízka

Každý z prípadov použitia je označený jednoznačným unikátnym identifikátorom v tvare

UCXX, kde XX je číslo prípadu použitia.

<b>Identifikátor:</b>	UC01	<b>Názov:</b>	Vytvorenie obalovača	<b>Priorita:</b>	1
<b>Účel:</b>	Vytvorenie nového obalovača.				
<b>Vstupné podmienky:</b>	-				
<b>Výstupné podmienky:</b>	Je vytvorený obalovač so stromom akcií.				
<b>Pokryté požiadavky:</b>	P7				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Používateľ zvolí meno, doménu obalovača, názov výstupného objektu a koreňový dokument (v prípade html - koreňová stránka).			
	2.	Systém vytvorí nový obalovač a koreňovú akciu (načítanie koreňového dokumentu). Aktuálny subdokument je koreňový dokument a vzor koreňovej akcie je prázdny (bez filtrov).			
	4.	Používateľ upravuje obalovač (include Úprava obalovača).			

<b>Identifikátor:</b>	UC02	<b>Názov:</b>	Upravenie obalovača	<b>Priorita:</b>	1
<b>Účel:</b>	Úprava existujúceho stromu akcií vybraného obalovača.				
<b>Vstupné podmienky:</b>	-				
<b>Výstupné podmienky:</b>	Obalovač zvolený používateľom je upravený.				
<b>Pokryté požiadavky:</b>	P11, ostatné požiadavky úpravy pokrývajú rozširujúce UC				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Používateľ nemá zvolený žiaden aktuálny obalovač. Systém ponúkne používateľovi zoznam obalovačov.			
	2.	Používateľ si zvolí obalovač, ktorý chce upravovať.			
	3.	Systém ponúkne používateľovi strom akcií obalovača.			
	4.	Používateľ si opakovane vyberá činnosť (extension point výber činnosti).			
	5.	Obalovač nemá konflikty. Koniec UC.			
<b>Alternatívy:</b>	1a.	Používateľ má zvolený aktuálny obalovač. Chod' na krok 3 v základnom toku.			
	1b.	Používateľ zadá obalovač zo súboru.			
	5a.	Obalovač má konflikty, ale používateľ jeho správnosť nepotvrdí. Koniec UC.			
	5b.	Obalovač má konflikty. Používateľ potvrdí jeho správnosť.			

<b>Identifikátor:</b>	UC03	<b>Názov:</b>	Spustenie obalovača	<b>Priorita:</b>	3
<b>Účel:</b>	Spustenie obalovača. Účel tohto UC sa týka iba správania sa systému pri spúšťaní obalovača s konfliktami. Ostatná funkcionálna (krokovanie, spúšťanie z príkazového riadku) je už implementovaná v existujúcom systéme.				
<b>Vstupné podmienky:</b>	-				
<b>Výstupné podmienky:</b>	Obalovač vykonáva svoju činnosť.				
<b>Pokryté požiadavky:</b>	P15				

	Krok	Činnosť
<b>Základný tok:</b>	1.	Používateľ nemá zvolený žiaden aktuálny obalovač. Systém ponúkne používateľovi zoznam obalovačov.
	2.	Používateľ si zvolí obalovač, ktorý chce upravovať.
	3.	Systém zistil, že obalovač nemá žiadne konflikty, uvedie obalovač do činnosti.
<b>Alternatívy:</b>	1a.	Používateľ má zvolený aktuálny obalovač. Chod' na krok 3 v základnom toku.
	1b.	Používateľ zadá obalovač zo súboru.
	3a.	Systém je v interaktívnom móde. Obalovač má nejaké konflikty. Systém to oznámi používateľovi a spustí obalovač.
	3b.	Systém nie je spustený z príkazového riadku. Obalovač má nejaké konflikty. Systém to zaznamená do log súboru a spustí obalovač.

<b>Identifikátor:</b>	UC04	<b>Názov:</b>	Vytvorenie novej akcie	<b>Priorita:</b>	1
<b>Účel:</b>					
<b>Vstupné podmienky:</b>	Používateľ má vybranú akciu, ktorej chce vytvoriť novú dcérsku akciu.				
<b>Výstupné podmienky:</b>	Bola vytvorená akcia, ktorá má vytvorený nový lokálny kontext s novým naučeným vzorom.				
<b>Pokryté požiadavky:</b>	P1, P2, P5, P6				
		<b>Krok</b>	<b>Činnosť</b>		
<b>Základný tok:</b>	1.		Používateľ vybraná akcia je rozšíriteľná. Systém povolí používateľovi novú akciu.		
	2.		Používateľ si vyberie typ vzoru (napr. XPath alebo regulárne výrazy), ktorý bude používaný v rámci akcie. Nová akcia zdedí kontext, ktorého aktuálny subdokument vznikol aplikovaním vzoru rodičovskej akcie a transformáciou novým vzorom. Vzor akcie je prázdny.		
	3.		Systém ponúkne používateľovi úpravu nového vzoru (include Naučenie vzoru).		
	4.		Používateľ si vybral vytvorenie akcie, ktorá nie je extrakčná ani formulárová.		
	5.		Používateľ nastaví jednoduché atribúty akcie (include Zmena jednoduchých atribútov akcie)		
<b>Alternatívy:</b>	1a.		Akcia je jednoduchá (nerozšíriteľná). Systém znemožní používateľovi zvolenie akcie. Späť na krok 1 v základnom kroku.		
	3a.		Nový vzor zostane prázdny. Akcia pracuje so zdedeným subdokumentom.		
	4a.		Používateľ si zvolil extrakčnú akciu.		
	4a.1.		Systém vytvorí prázdnu extrakčnú akciu a mapovanie na výstupný objekt (extend Upravenie extrakčnej akcie).		
	4b.		Používateľ si zvolil formulárovú akciu.		
	4b.1		Systém vytvorí prázdnu formulárovú akciu a mapovanie na formulárové dáta (extend Upravenie formulárovej akcie)		

<b>Identifikátor:</b>	UC05	<b>Názov:</b>	Učenie vzoru	<b>Priorita:</b>	1
-----------------------	------	---------------	--------------	------------------	---



<b>Účel:</b>	Naučenie vzoru pomocou pozitívnych a negatívnych príkladov.	
<b>Vstupné podmienky:</b>	Je vybraná akcia, ktorej lokálny kontext obsahuje vzor.	
<b>Výstupné podmienky:</b>	Vzor má priradené/upravené filtre. Vzor si pamätá históriu učenia.	
<b>Pokryté požiadavky:</b>	P8, P9, P13	
	<b>Krok</b>	<b>Činnosť</b>
<b>Základný tok:</b>	1.	Používateľ manuálne nezmenil filtre vzoru. Systém neupraví existujúce filtre.
	2.	Systém grafický vyznačí aktuálny vzor v dokumente.
	3.	Používateľ označí príklady, ktoré ho zaujímajú (pozitívne) a príklady, ktoré ho nezaujímajú (negatívne). Používateľ nevybral príklady, ktoré už v tréningovej množine sú.
	4.	Systém generalizuje vzor na základe príkladov. Aktuálne filtre si systém asociuje s množinou pridaných príkladov. Tieto príklady si vloží do histórie učenia. Späť na krok 2.
<b>Alternatívy:</b>	1a.	Používateľ manuálne zmenil filtre vzoru. Systém nahradí manuálne zmenené filtre filterami z poslednej histórie učenia.
	3a.	Medzi vyznačenými príkladmi sa nachádza príklad, ktorý sa už nachádza v tréningovej množine príkladov. Systém na to upozorní používateľa a zvolený príklad nepridá do tréningovej množiny.
	3b.	Používateľ je spokojný so vzorom. Ukončí učenie. Ak bol vzor zmenený podstrom akcií označí systém ako konfliktný. Koniec UC.

<b>Identifikátor:</b>	UC06	<b>Názov:</b>	Manuálna zmena vzoru	<b>Priorita:</b>	2
<b>Účel:</b>	Manuálna zmena naučených filtrov				
<b>Vstupné podmienky:</b>	Používateľ má zvolenú akciu a jej vzor, ktorý má priradené filtre				
<b>Výstupné podmienky:</b>	Filtre vzoru boli upravené. Vzor si pamätá všetky naučené filtre.				
<b>Pokryté požiadavky:</b>	P12				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Systém grafický vyznačí aktuálny vzor v dokumente.			
	2.	Používateľ ručne zmení naučené filtre. Späť na krok 1.			
<b>Alternatívy:</b>	2a.	Používateľ je spokojný so vzorom. Ak bol vzor zmenený podstrom akcií označí systém ako konfliktný. Koniec UC.			

<b>Identifikátor:</b>	UC07	<b>Názov:</b>	Úprava formulárovej akcie	<b>Priorita:</b>	2
<b>Účel:</b>	Vytvoriť mapovanie subdokumentu na formulárové dáta.				
<b>Vstupné podmienky:</b>	Používateľom zvolená akcia je formulárová.				
<b>Výstupné podmienky:</b>	Formulárová akcia má priradené mapovanie na formulárové dáta.				
<b>Pokryté požiadavky:</b>	P4				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Používateľ vytvára kontextové parametre a vkladá ich hodnoty.			
	2.	Systém postupne ponúka používateľovi vstupné formulárové elementy subdokumentu.			
	3.	Pre každý element vyberie používateľ zodpovedajúcu kontextovú			

		premennú, ktorej hodnota sa použije pri vyplnení formulára.
<b>Alternatívy:</b>	1a.	V kontexte sa nachádzajú všetky potrebné premenné. Chod' na krok 5 v základnom toku.

<b>Identifikátor:</b>	UC08	<b>Názov:</b>	Úprava extrakčnej akcie	<b>Priorita:</b>	1
<b>Účel:</b>	Namapovať subdokument (element zoznamu subdokumentov, ktorý vznikne aplikovaním vzoru na rodičovský subdokument) na výstupný objekt.				
<b>Vstupné podmienky:</b>	Používateľom zvolená akcia je extrakčná.				
<b>Výstupné podmienky:</b>	Extrakčná akcia má priradené mapovanie na výstupný objekt.				
<b>Pokryté požiadavky:</b>	P3				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Používateľ zadáva premenné do kontextu, ktorých hodnoty chce naplniť používateľskými dátami.			
	2.	Používateľ zadá cestu vo výstupnom objekte, do ktorej sa zapíše extrahovaný subdokument.			
<b>Alternatívy:</b>	2a.	Používateľ zadá premenné kontextu, do ktorých sa zapíše extrahovaný subdokument.			
	2b.	Používateľ zadá aj premenné kontextu aj cestu vo výstupnom objekte, do ktorých sa zapíše extrahovaný subdokument.			

<b>Identifikátor:</b>	UC09	<b>Názov:</b>	Presunutie akcie	<b>Priorita:</b>	3
<b>Účel:</b>	Zmena rodičovskej akcie				
<b>Vstupné podmienky:</b>	Používateľ má zvolenú akciu.				
<b>Výstupné podmienky:</b>	Presúvaná akcia je dcérskou akciou inej akcie. Presunutý je aj podstrom akcie				
<b>Pokryté požiadavky:</b>	P10, P11				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Používateľ si zvolí presúvanú a cieľovú akciu (systémom copy&paste alebo drag&drop)			
	2.	Cieľová akcia môže byť kompozitná. Systém presunie akciu aj s jej podstromom.			
	3.	Systém označí presunutý podstrom ako konfliktný.			
<b>Alternatívy:</b>	2a.	Cieľová akcia nemôže mať dcérske akcie. Koniec UC.			

<b>Identifikátor:</b>	UC10	<b>Názov:</b>	Zmena jednoduchých atribútov akcie	<b>Priorita:</b>	2
<b>Účel:</b>	Zmena atribútov akcie, ktoré majú jednoduchú hodnotu – žiadne štruktúrované dáta.				
<b>Vstupné podmienky:</b>	Používateľ má zvolenú akciu.				
<b>Výstupné podmienky:</b>	Akcia má zmenené atribúty.				
<b>Pokryté požiadavky:</b>	P10				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Používateľ nastaví nové hodnoty atribútov, pridá nové atribúty alebo odstráni existujúce atribúty			

	2.	Systém zistil, že zmena atribútov môže porušiť podstrom akcií. Označí tento podstrom ako konfliktný.
<b>Alternatívy:</b>	2a.	Zmena atribútov nemôže porušiť podstrom akcií. Systém neoznačí podstrom ako konfliktný

<b>Identifikátor:</b>	UC11	<b>Názov:</b>	Odstránenie naučených príkladov	<b>Priorita:</b>	3
<b>Účel:</b>	Odstránenie naučenia vybraných príkladov				
<b>Vstupné podmienky:</b>	Používateľ má vybranú akciu a vzor, ktorému bude odstraňovať príklady.				
<b>Výstupné podmienky:</b>	Vzor je v stave(má zhodné filtre) v akom bol pred naučením najstaršieho z odstránených príkladov.				
<b>Pokryté požiadavky:</b>	P14				
	<b>Krok</b>	<b>Činnosť</b>			
<b>Základný tok:</b>	1.	Systém ponúkne používateľovi zoznam všetkých naučených príkladov, ktorými vznikol vzor.			
	2.	Používateľ si vyberie príklady na zmazanie.			
	3.	Systém zobrazí používateľovi príklady na zmazanie a subdokument, ktorý vyfiltruje vzor po zmazaní príkladov.			
	4.	Používateľ potvrdí zmazanie.			
	5.	Systém vymaže vybrané vzory a filtre nastaví do stavu v akom boli pred naučením najstaršieho z odstránených príkladov (informácia z histórie učenia)			
<b>Alternatívy:</b>	4a.	Používateľ nepotvrdí zmazanie a chce zmeniť výber príkladov. Späť na krok 2.			



## 5 Použité technológie

Jedným z cieľov tohto projektu je priblíženie tvorby obalovačov rádovým používateľom. Keďže systém kladie dôraz na interaktivitu s používateľom, je nutné obalovač zakomponovať do takého konceptu, ktorý používateľovi ponúkne jednoduché rozhranie bez nutnosti mohutných zmien (inštalácie) v systéme. Ako najvhodnejšie riešenie sa javí použitie prehliadača Mozilla Firefox a možnosti jeho rozšírenia (ide o voľne dostupný prehliadač s otvoreným programovým kódom).

S návrhom a implementáciou rozšírení úzko súvisia technológie XUL a XPCOM, ktoré sú opísané v nasledujúcich častiach.

### 5.1 Jazyk XUL

XUL<sup>6</sup> je značkovací jazyk, ktorý slúži na vytváranie používateľských rozhraní. Je dôležitou súčasťou projektu Mozilla, ktorý ho využíva v širokom spektre. Výhodou tohto jazyka je jeho prenosnosť. To znamená, že ho môžeme aplikovať vo väčšine existujúcich operačných systémoch, od systému Windows až po systémy Linux. Táto technológia umožňuje tvorbu sofistikovaných aplikácií bez potreby použitia špeciálneho nástroja. Využíva pritom už existujúce ovládacie prvky (kontajnery, menu, panel nástrojov, ...).

#### 5.1.1 Použitie jazyka XUL

Použitie jazyka XUL v systéme Mozilla sa predovšetkým zameriava na implementáciu používateľského rozhrania vytváraných prídavných modulov alebo aplikácií, ktoré uľahčujú prácu, prípadne rozširujú funkcionality tohto systému. Nesmieme však zabúdať aj na to, že pomocou tohto jazyka a kaskádnych štýlov CSS vieme vytvárať dizajn webových stránok, ktoré je možné zobrazovať pomocou webového prehliadača Mozilla Firefox.

Jazyk XUL podporuje aj dynamické vytváranie daného kontextu. Vďaka možnosti prepojenia so skriptovacím jazykom JavaScript sme schopní vytvoriť dynamické používateľské rozhranie. Pomocou neho dokážeme modifikovať celý vzhľad tohto rozhrania na základe interakcie používateľa.

#### 5.1.2 Štruktúra jazyka XUL

XUL je stromovo-orientovaný štruktúrovaný jazyk, založený na metajazyku XML. Celý kontext, vytvorený v tomto jazyku, je uložený v textovom súbore (súbor s príponou .xul), čo uľahčuje prácu pri definícii používateľských rozhraní (stačí jednoduchý textový editor).

Základná štruktúra XUL súboru sa skladá z XML hlavičky a značiek, ktoré obsahujú definované atribúty a udalosti. Tieto značky môžeme rozdeliť do troch skupín:

- Kmeňové elementy
- Elementy rozloženia

---

<sup>6</sup> XUL, špecifikácia dostupná na <http://www.mozilla.org/projects/xul/>

- Vizualiéne prvky

## Kmeňové elementy

Kmeňové elementy tvoria základ každého používateľského rozhrania. Bez nich by ho nebolo možné vytvoriť. Kmeňové elementy sa používajú na definovanie typu zobrazenia vytváraného rozhrania (aplikačné okno, dialógové okno) a obsahujú zoznamy elementov (elementy rozloženia, kontajnery, vizualiéne prvky) na nižších vrstvách v stromovej štruktúre dokumentu XUL. Medzi najpoužívanéjšie kmeňové elementy patria:

- *window* - používa sa pri tvorbe hlavného okna alebo webových aplikácií  
- obsahuje atribúty, ktoré definujú pozíciu, názov okna a iné
- *dialog* - vytvorí dialógové okno, závislé od danej platformy
- *page* - slúži na vnorenie daného dokumentu do iného

## Elementy uloženia

Tieto elementy slúžia na umiestenie skupiny kontajnerov a vizualiénych prvkov do riadku, stĺpca alebo do tabuľky. V XUL dokumente sa môžu vyskytovať v tele kmeňových elementov. Najpoužívanéjšimi elementami uloženia sú:

- *vbox, hbox* - prvky budú uložené v stĺpci, resp. v riadku
- *stack* - prvky budú uložené nad sebou, všetky sa zobrazia
- *deck* - podobne ako *stack*, ale vždy sa zobrazí iba jeden prvok
- *tabbox* - manažér záložiek, obsahuje elementy *tabs* a *tabpanel*s
- *grid* - umiestni prvky do tabuľky

## Vizualiéne prvky

Poslednou skupinou elementov sú vizualiéne prvky. Sú to klasické ovládacie prvky, s ktorými sa stretávame pri tvorbe používateľského rozhrania v rôznych vývojových prostrediach. Nie je tomu inak ani pri jazyku XUL. Medzi najpoužívanéjšie môžeme zahrnúť:

- *label* - element na zobrazenie textu
- *textbox* - ovládací prvok na vkladanie používateľom zadaného textu
- *listbox* - prvok na zobrazenie zoznamu

## 5.2 Komponentový objektový model XPCOM

Multiplatformový objektový model XPCOM<sup>7</sup> je rámec dovoľujúci rozdeliť jednoliate a masívne softvérové projekty na malé modularizované časti. Tieto časti – komponenty – sú opäť zostavené dohromady v priebehu behu aplikácie.

---

<sup>7</sup> XPCOM špecifikácia dostupná na <http://www.mozilla.org/projects/xpcom/>

Účelom modelu XPCOM je možnosť nezávislého vývoja jednotlivých softvérových súčastí bez nutnosti ich vzájomného spojenia. XPCOM oddeľuje implementáciu komponentu od jeho rozhrania, čím je dosiahnutá znovupoužiteľnosť v rôznych aplikáciách a jednoduchá možnosť zmeny funkcionality existujúcich aplikácií.

XPCOM je úzko spojený s projektom Mozilla, keď bol vytvorený za účelom poskytnutia prístupu k funkcionalite zobrazovacieho jadra Gecko a možnosti jeho nenáročného rozširovania. Napríklad samotný prehliadač Mozilla Firefox je vo svojej podstate tiež iba súborom komponentov. Každé tlačidlo, každé menu, jednoducho každý element je samostatný komponent, čo robí prehliadač ľahko rozširiteľným pri zachovaní pôvodnej štruktúry. Výsledkom toho je aj množstvo tzv. rozšírení (angl. addons), ktoré prispievajú k jeho narastajúcej popularite.

Úloha komponentového objektového modelu XPCOM je zrejma – spolu s jazykom XUL (ale nie len ním) poskytuje možnosti rozšírenia funkcionality prehliadača (a iných aplikácií projektu Mozilla) bez nutnosti zmeny jeho programového kódu.

### 5.2.1 Rozhrania

Pojem rozhranie je v prípade komponentového modelu kľúčový. Prostredníctvom rozhraní zabezpečíme zapuzdrenie použitých tried, čím jednoznačne definujeme metódy, ktoré môžu byť nad komponentmi použité. Každá trieda v modeli XPCOM je odvodená od elementárneho rozhrania `nsISupports` (Obr. 5-1).

```
[scriptable, uuid(00000000-0000-0000-c000-000000000046)]
interface nsISupports {
    void QueryInterface(in nsIIDRef uuid,
                       [iid_is(uuid), retval] out nsQIResult result);
    [noscript, notxpcom] nsrefcnt AddRef();
    [noscript, notxpcom] nsrefcnt Release();
};
```

Obr. 5-1: Rozhranie `nsISupports` definované jazykom XPIDL<sup>8</sup>

Rozhranie `nsISupports` pomocou uvedených metód adresuje dve fundamentálne aspekty komponentového programovania – dĺžku života komponentu a tzv. „dopytovanie rozhrania“, t.j. schopnosť identifikácie komponentu s podporovaným rozhraním. Obe metódy tvoria základ procesu volania komponentu napr. z kódu rozšírenia prehliadača.

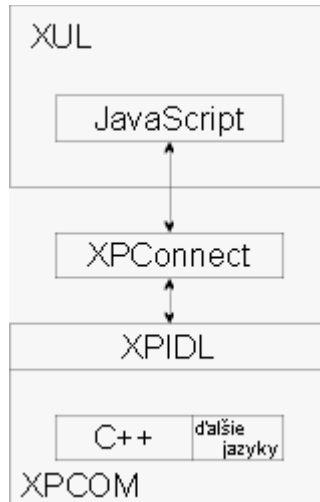
### 5.2.2 XPConnect

XPConnect je vrstva modelu XPCOM, ktorá slúži na prepojenie XPCOM komponentov a skriptovacieho jazyka, akým je napr. JavaScript (JavaScript je súčasťou balíkov rozšírení prehliadača Mozilla Firefox). Schématický náčrt architektúry je zobrazený na Obr. 5-2.

Prístup ku komponentom pomocou XPConnect by sa dal zhrnúť do troch krokov [6]:

<sup>8</sup> Jazyk XPIDL je mutáciou jazyka IDL, slúži na opí rozhraní softvérových komponentov; referencie dostupné na: <http://developer.mozilla.org/en/docs/XPIDL>

1. získanie objektu komponentu
2. prístup k rozhraniu
3. volanie požadovanej metódy



Obr. 5-2: Architektúra XPCOM [7]

Po vykonaní prvých dvoch krokov môžeme ten tretí opakovať koľkokrát to bude nutné. Uvedenú postupnosť demonštruje Obr. 5-2, ktorý uvádza volanie Mozilla-zabudovaného komponentu `nsILocalFile`.

```

var aFile =
  Components.classes["@mozilla.org/file/local;1"].createInstance();
if (aFile instanceof Components.interfaces.nsILocalFile) {
  aFile.initWithPath("/mozilla/testfile.txt");
  aFile.remove(false);
}
  
```

Obr. 5-3: Stručný príklad prístupu ku komponentu pomocou XPCConnect

V uvedenom príklade je ukážka vymazania súboru pomocou JavaScript kódu. Na začiatku dôjde k inicializácii premennej `aFile`, do ktorej sa uloží inštancia komponentu súboru. Na ďalšom riadku sa realizuje prístup k rozhraniu `nsILocalFile`. Zvyšná časť kódu sú metódy, ktoré zabezpečia samotnú funkčnosť nad súborom uvedeným ako argument.

### 5.2.3 JavaXPCOM

Zatiaľ nebolo spomenuté, že XPCOM implicitne podporuje jazyk C++. Pre tvorbu nepôvodných Mozilla komponentov v inom programovacom jazyku je potrebná nadstavba v podobe externého modulu, v našom prípade JavaXPCOM. JavaXPCOM slúži ako most medzi



modelom XPCOM a s ním súvisiacou funkcionalitou na jednej strane a programovým kódom v jazyku Java na strane druhej.

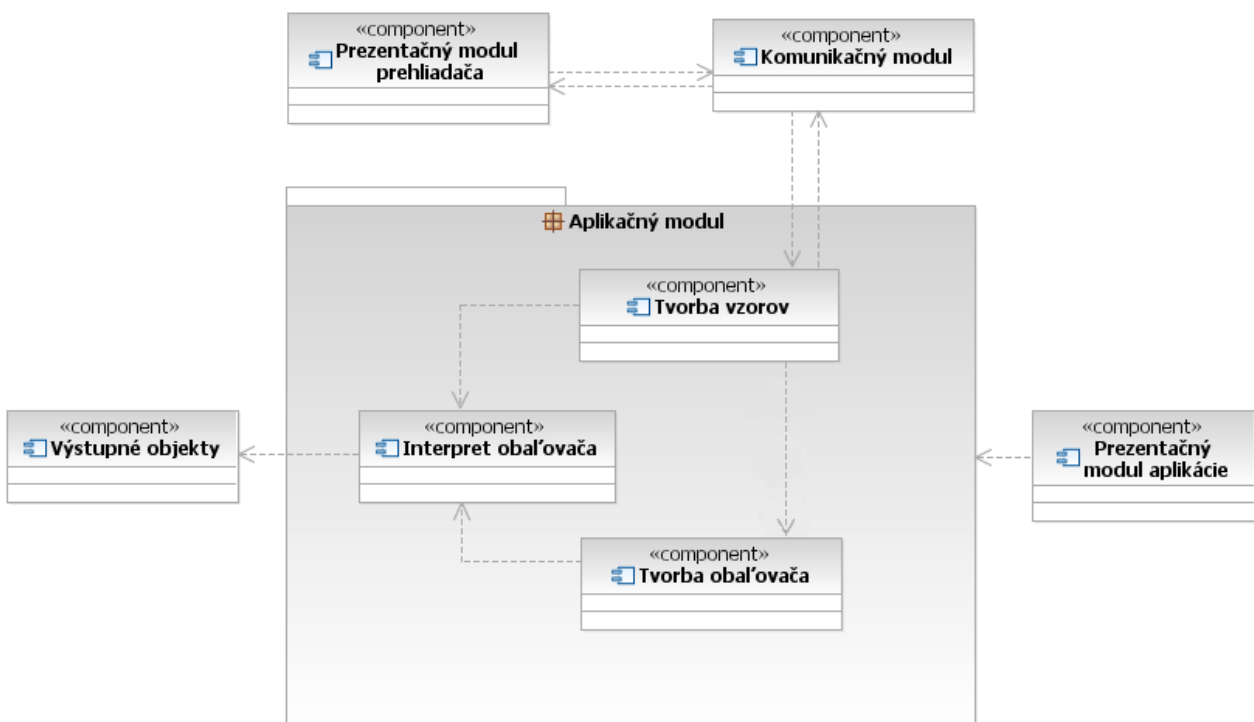


## 6 Hrubý návrh systému

Kapitola 6 prezentuje hrubý návrh systému. Na základe analýzy existujúcich riešení a využiteľnosti nových technológií navrhujeme architektúru systému, ktorá zodpovedá špecifikácii požiadaviek na systém (časť 6.1). Na základe architektúry postupne v ďalších siedmich častiach opisujeme hrubý náčrt jednotlivých komponentov – modulov systému. Kapitulu uzatvára časť 6.9, v ktorej uvádzame použité implementačné prostriedky a vývojové nástroje.

### 6.1 Architektúra systému

Základná architektúra systému sa nachádza na Obr. 6-1.



Obr. 6-1: Architektúra obalovača

Opis návrhu každého z modulov je predmetom ďalších častí tejto kapitoly.

### 6.2 Prezentačný modul aplikácie

Prezentačný modul aplikácie slúži ako rozhranie medzi používateľom a obalovačom. Umožňuje tak riadenie obalovača na základe interakcie používateľa. Hlavnou úlohou je distribúcia získaných údajov pre modul tvorby vzorov, ale aj prezentácia spracovaných údajov z obalovača. Používateľ bude môcť zasahovať do nasledujúcich činností:

- vytvorenie obalovača

- upravenie obalovača
- spustenie obalovača

### 6.2.1 Proces tvorby obalovača

Na začiatku tvorby obalovača je potrebné, aby používateľ vykonal jeho inicializáciu. V tomto prípade používateľ vyplní formulár, ktorý zahŕňa nasledujúce informácie:

- názov obalovača
- doména obalovača
- názov výstupného objektu
- odkaz na koreňový dokument

V prípade, že koreňovým dokumentom je webová stránka, je potrebné zadať URL adresu. Túto adresu je možné zadať ručne, alebo pomocou interakcie používateľa v prehliadači Mozilla Firefox (toto spravuje prezentačný modul prehliadača a komunikačný modul).

Po zadaní inicializačných údajov môže používateľ daný obalovač rôzne konfigurovať. Vykonáva to cez používateľské rozhranie, kde si môže vytvárať a modifikovať celý strom akcií (formulár bude tento strom zobrazovať a používateľ ho môže modifikovať – tvorba, rušenie a presun akcií), ktoré sa majú vykonať. Každá akcia obsahuje určité parametre, ktoré môže používateľ podľa potreby meniť (zobrazenie príslušného formulára nastavení po vybratí danej akcie v strome akcií). V prípade extrakčných akcií je potrebné zadať výstupný objekt, do ktorého sa budú ukladať vydolované údaje (výberom objektu zo zoznamu). Ak pracujeme s navigačnými formulárovými akciami, musíme zabezpečiť namapovanie formulárových elementov na vstupné údaje (rozhranie, kde používateľ priradí ku každému typu vstupného údajja práve jeden element vo formulári). Niektoré akcie si vyžadujú zásah používateľa. Jednou z nich je aj akcia učenia vzorov, kde používateľ určí príklady toho, čo chce, prípadne nechce získať (pozitívnych, resp. negatívnych príklad). Toto určovanie sa vykonáva pomocou prezentačného modulu prehliadača, ktorý je popísaný v kapitole 6.3.

Medzi ďalšie funkcie prezentačného modulu aplikácie patrí aj odstránenie naučených príkladov a manuálna zmena vzorov. V prípade manuálnej zmeny vzorov si používateľ vyberie daný vzor (XPath výraz alebo regulárny výraz) zo zoznamu a tento sa mu zobrazí. Ak ho potrebuje zmeniť, môže to vykonať. V prípade odstránenia naučených príkladov si používateľ vyberie určitú akciu a vzor (zo stromu akcií), ktorému chce odstrániť príklady, a potvrdí zrušenie.

### 6.2.2 Proces úpravy obalovača

Úprava obalovača obsahuje rovnakú podskupinu činností, ktoré sa vyskytujú aj počas tvorby obalovača. Nie je preto dôležité opisovať tieto činnosti a návrh rozhrania ešte raz. Tieto informácie sú dostupné v predchádzajúcej časti. Jediným rozdielom je začiatok vykonávania týchto činností. Pri tvorbe obalovača musel používateľ uviesť inicializačné údaje o obalovači. V prípade úpravy už nemusí zadávať počiatočné informácie, ale vyberie si už vytvorený obalovač a ten

modifikuje. Systém mu ponúkne zoznam existujúcich obal'ovačov a používateľ postupuje rovnako ako v prípade jeho tvorby (tvorba, rušenie a presun akcií).

### 6.2.3 Spustenie obal'ovača

Ak chce používateľ dolovať údaje, musí spustiť obal'ovač. Tu mu systém ponúkne, ktorý obal'ovač chce použiť (zoznam všetkých definovaných obal'ovačov). Používateľ si jeden vyberie a môže spustiť režim behu alebo ladenia. Počas spracovania sa mu vo formuláre zobrazia výsledky dolovania, prípadne dodatočné informácie pri režime ladenia.

## 6.3 Prezentačný modul prehliadača

Prezentačný modul prehliadača sa využíva na uľahčenie práce pri tvorbe obal'ovača. Slúži ako nadstavba prezentačného modulu aplikácie a využívajú sa tu funkcie systému Mozilla Firefox. Prezentačný modul prehliadača pracuje na princípe rozšírenia tohto systému.

Jeho prvou úlohou pri tvorbe obal'ovača je získanie adres koreňových dokumentov (URL) a zaslanie tejto informácie prezentačnému modulu aplikácie. Túto adresu zadá používateľ do okna prehliadača, prezentačný modul prehliadača (rozšírenie systému Mozilla Firefox) ju vhodne zakóduje a vyšle do komunikačného modulu. Ďalšou úlohou prezentačného modulu je získavanie pozitívnych, prípadne negatívnych príkladov. Tie slúžia pri procese učenia vzorov. Používateľ si vyberie určitý element dokumentu a prezentačný modul túto operáciu zaznamená. Zistí, o ktorý element sa jedná a cestu k nemu pošle prezentačnému modulu aplikácie (cez komunikačný modul), ktorý ho spracuje. Táto cesta sa reprezentuje pomocou XPath alebo regulárnych výrazov. Poslednými úlohami tohto modulu je zobrazenie určitej časti dokumentu (subdokument), ktorá bližšie špecifikuje oblasť požadovaných (dolovaných) údajov, a grafické označenie aktuálneho vzoru v dokumente.

## 6.4 Komunikačný modul

Návrh komunikačného modulu vychádza z analýzy komponentového objektového modelu XPCOM v časti 5.2. Keďže je nutné zabezpečiť komunikáciu medzi prezentačným modulom prehliadača a samotnou aplikáciou obal'ovača (založenou na platforme Java), bude využitá aj nadstavba JavaXPCOM.

Hlavnou úlohou komponentového modelu XPCOM bude priniesť rozšíreniu do prehliadača Mozilla Firefox (t.j. prezentačnému modulu prehliadača) takú funkcionálnosť, na ktorú JavaScript a XUL nestačia. Pôjde o využitie funkcionality obal'ovača, ktorý slúži na dolovanie dát z webu.

Komunikačný modul kopíruje schému uvedenú na Obr. 5-2. Na jednej strane bude definované rozhranie jednotlivých Java tried aplikačného rámca WrapperSuite, na strane druhej Mozilla Firefox rozšírenie tvorené jazykom XUL a JavaScript-om. Komunikácia bude prebiehať v dvoch smeroch:

- prezentačný modul prehliadača -> aplikačný modul

- aplikačný modul -> prezentačný modul prehliadača

V prvom prípade prezentačný modul prehliadača Mozilla Firefox, ktorý zabezpečuje interaktivitu nad HTML stránkami, bude aplikačnej časti poskytovať informácie o tom, aké pozitívne, resp. aké negatívne príklady boli vybrané. Tie budú reprezentované XPath výrazmi, resp. regulárnymi výrazmi. Taktiež budú prenášané informácie o navigačných akciách, tj. o pohybe v priestore webu.

Druhý prípad bude prenášať informácie z aplikačného modulu smerom do rozšírenie. V tejto časti bude nutné zabezpečiť prispôsobovanie sa obsahu prehliadača požiadavkám, ktoré definuje používateľ prostredníctvom obalovača. V prevažnej miere pôjde o zmeny nad DOM modelom zobrazovaných stránok. Predpokladáme využitie W3C DOM Connectora [9]. W3C Connector je v podstate balík jazyka Java, ktorý sa používa na prístup k DOM stromu prehliadača Mozilla Firefox z platformy Java, pretože implementuje štandardné rozhrania `org.w3c.*`. Znamená to, že bude možné efektívne dopytovanie nad DOM stromom prostredníctvom robustných XPath nástrojov, akými sú napr. Xerces alebo Saxon.

## 6.5 Aplikačný modul – Interpreter

Spúšťanie a vykonávanie akcií používateľom definovaného obalovača zabezpečuje Interpreter. Vykonávanie začína spustením operácií štartovacej akcie, ktorá predstavuje načítanie požadovanej web stránky a inicializácia kontextu obalovača. Po ukončení aktuálnej akcie Interpreter automaticky spustí vykonanie všetkých ďalších používateľom naplánovaných akcií (NextAction). Interpreter pre nasledujúce akcie inicializuje lokálny kontext vzhľadom na predchádzajúcu akciu.

Z hľadiska architektúry systému (časť 6.1) Interpreter vykonáva akcie vytvorené v module Tvorba obalovača a v lokálnych kontextoch akcií používa vzory z modulu Tvorba vzorov. Výsledky svojej práce, extrahované dáta, odovzdáva Interpreter modulu Výstupné objekty (všetky uvedené moduly budú popísané neskôr).

Interpreter je spustený cez API Dialog a umožňuje okrem normálneho spôsobu spustenia dolovania aj proces dolovania v ladiacom režime, kedy je možné beh akcií krokovať.

### 6.5.1 Kontext

Všetky akcie súčasnej implementácie Wrapper Siute pracujú s globálnym kontextom. Naším cieľom je prejsť na systém tzv. lokálneho kontextu, teda každá ďalšia nasledovná akcia v strome akcií bude pracovať už len so svojim lokálnym kontextom, nezávislým od kontextu akcie z inej vetvy stromu akcií. To zvýši odolnosť vytvorených obalovačov proti prípadným čiastočným zmenám štruktúry na obalovanej stránke. Pod kontextom rozumieme dátovú štruktúru, ktorá obsahuje aktuálny stav obalovania. Obsahuje vstupné premenné, vzor, aktuálny subdokument, lokálny výstupný objekt, cookies, prípadne autentifikačné údaje. Aktuálny subdokument sa vytvorí prostredníctvom rozhrania na dokument, teda v závislosti od vstupného dokumentu to môže byť

časť DOM stromu prípadne reťazec String.

## 6.5.2 Akcia

Pod akciou rozumieme operácie nad lokálnym kontextom. Každá akcia vo svojej reprezentácii obsahuje atribút `NextActions`, ktorý určuje, ktoré akcie majú v programe po nej bezprostredne nasledovať (t.j. aké vetvy z nej vychádzajú). Každá ďalšia akcia pracuje nad svojim lokálnym kontextom.

Rozoznávame typy akcií:

- *Navigačné akcie* - zabezpečujú pohyb po web priestore, prechod na želanú stránku a jej samotné načítanie do interného formátu využívaného obalovačom, ale aj odoslanie formulára na server. Napríklad akcia `LoadPage`, ktorá načíta stránku zo zadanej URL.
- *Extrakčné akcie* - zabezpečujú výber a dolovanie požadovaných dát prostredníctvom vzorov. Detailnejší popis akcií a ich atribútov je v časti 6.7 - Tvorba obalovača.

## 6.5.3 Vzor

Ide o nový termín, z ktorým doterajšia implementácia obalovača nepracuje. Akcia pomocou filtrov identifikuje vzor. Vzor teda obsahuje filtre ako kritérium na výber subdokumentu. Aplikovaním vzoru na subdokument vznikajú nové subdokumenty. Vzory nám umožnia do tvorby obalovača zaviesť prvok učenia sa. Obalovač sa na základe pozitívnych a negatívnych príkladov, získaných interakciou s používateľom, naučí identifikované vzory. Do budúcnosti počítame s prítomnosťou rôznych typov vzorov, preto samotný vzor bude okrem filtrov obsahovať aj svoj typ a spôsob učenia. Ak teda pracujeme s HTML dokumentom, aktuálnym subdokumentom je DOM strom, vzor je typu HTML a filtre sú typu XPath. Ak by bol potrebný napr. vzor typu Text, filtre by boli regulárne výrazy a aktuálny subdokument by bol znakový reťazec (string).

## 6.5.4 Cookies

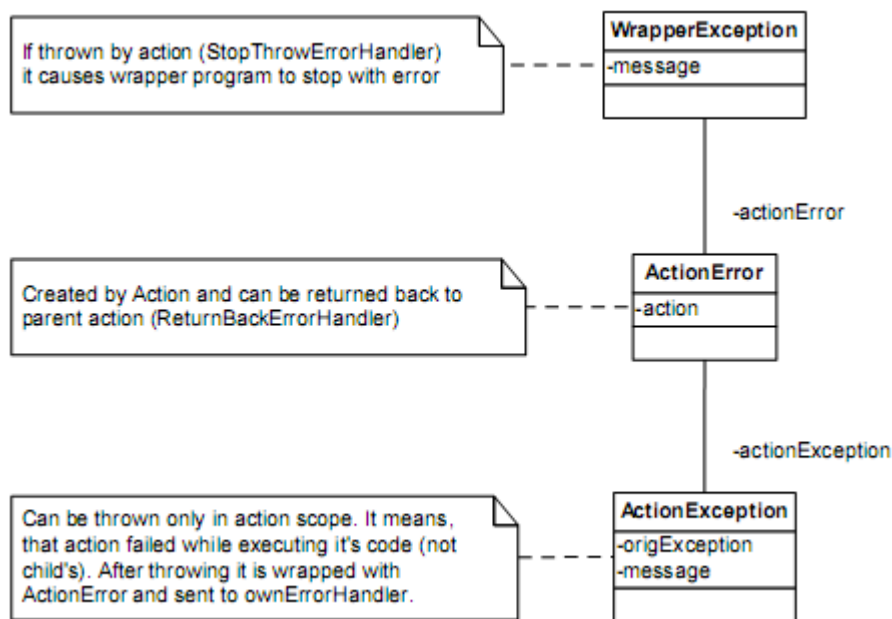
Pri odosielaní požiadavky na web server sa serveru posiela aj zoznam všetkých doteraz pri navigácii na tomto serveri získaných cookies.

## 6.5.5 Autentifikačné údaje

Sú web serverom vyžadované údaje potrebné na pre http autentifikáciu, ktoré sa musia posielat' po prihlásení na server v každej ďalšej požiadavke.

## 6.5.6 Ošetrovanie výnimiek

Model ošetrovania výnimiek vychádza už z existujúceho aplikačného rámca obalovača a nepredpokladáme potrebu jeho výraznej zmeny.



Obr. 6-2: Model výnimiek [5]

WrapperException je výnimka, ktorá nie je odchyťovaná akciami a dostane ju až interpret, ktorý spustil obalovač a má za dôsledok ukončenie vykonávania programu obalovača. Generuje sa akciami v prípade vzniku chyby a zároveň je použitý StopThrowErrorHandler. Táto výnimka potom obaluje ActionError, ktorý v akcii vznikol. V prípade, že vykonávanie kódu akcie skončí chybou (napr. na stránke nie sú požadované dáta) je vygenerovaná ActionException, obalená do ActionError a poslaná na vstup ownErrorHandler, ktorý ma na starosti spracovanie vlastných chýb. V prípade, že ide o ReturnBackErrorHandler, chyba sa vráti akcii, ktorá volala aktuálnu akciu.

## 6.6 Aplikačný modul – Tvorba vzorov

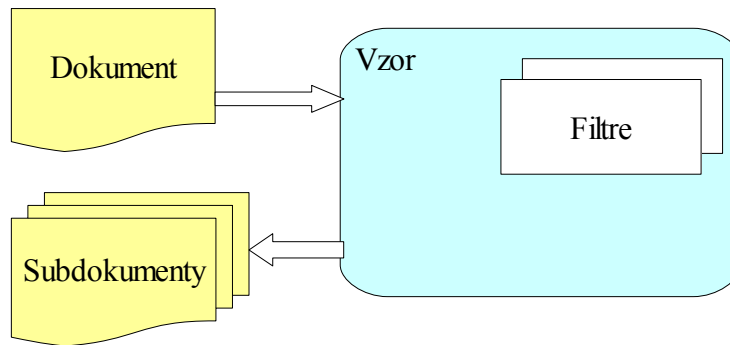
### 6.6.1 Vzor

Pri práci s dokumentom treba pristupovať postupne k jeho jednotlivým častiam, aby sa nad nimi mohli vykonávať konkrétne akcie. Tieto akcie spracovávajú určitý kontext, ktorý dostanú na vstupe. Na zmenu a získavanie nového kontextu slúžia vzory.

**Vzor** je objektom systému, ktorý sa aplikuje v rámci určitého kontextu obsahujúceho dokument (alebo časť dokumentu) na získanie žiadanej časti tohoto dokumentu (subdokument).

Za účelom výberu časti dokumentu obsahuje každý vzor jeden alebo viac filtrov, ktoré definujú tento výber. Výsledkom výberu môže byť súvislý subdokument, alebo viacero samostatných subdokumentov. Objekt vzor musí preto poskytovať vo svojom rozhraní metódy na získavanie jednotlivých inštancií vybraných subdokumentov. Schému aplikovania vzoru zobrazuje Obr. 6-3.



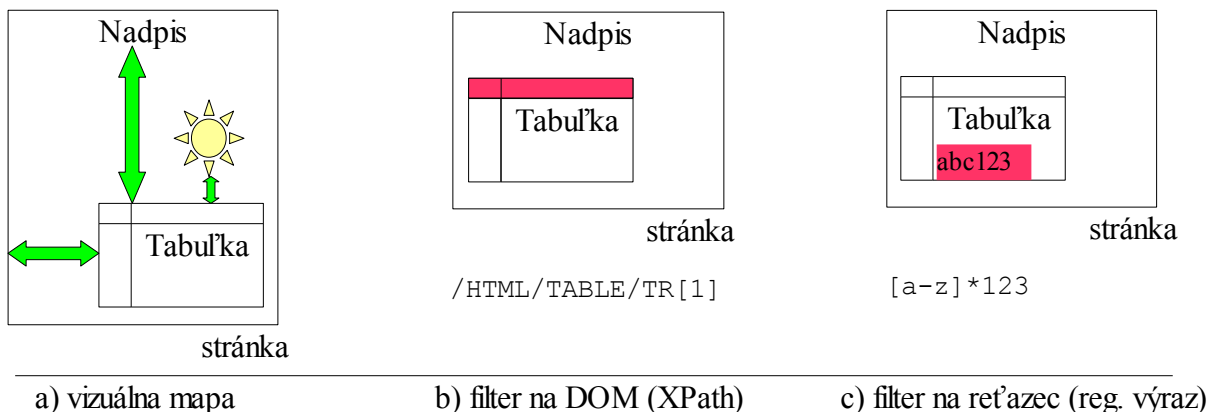


Obr. 6-3: Schéma aplikácie vzoru

## 6.6.2 Typy filtrov

V súčasnosti existuje viacero typov filtrov na výber informácií z webovej stránky. Jednotlivé typy sú zatiaľ rozpracované do rôznych úrovní. Pri výbere typov filtrov, ktoré má výsledný systém obsahovať, do úvahy pripadali najmä nasledovné typy:

- *vizuálna mapa* – Filter definuje požadovanú časť dokumentu na základe jej polohy na zobrazovanej stránke a tiež jej vzájomnej polohy na zobrazenej stránke vzhľadom k ostatným prvkom. Príklad - Obr. 6-4a).
- *filter na DOM* - Tento filter vychádza z hierarchickej stromovej štruktúry HTML kódu stránky. K jednotlivým častiam pri výbere pristupuje ako k elementom tohoto stromu alebo k podstromom. Pričom aj jeden element je vlastne podstromom stromu. Príklad - Obr. 6-4b).
- *filter na reťazce* – Filter pristupuje k dokumentu ako k reťazcu, pričom naň aplikuje kritériá vyhľadávania v reťazci. Špecifikácia vyberanej časti dokumentu môže byť realizovaná napríklad pomocou regulárnych výrazov. Príklad - Obr. 6-4c).



a) vizuálna mapa

b) filter na DOM (XPath)

c) filter na reťazec (reg. výraz)

Obr. 6-4: Príklady typov filtrov

Vzhľadom na požiadavky kladené na projekt boli do návrhu systému zaradené nasledovné dva typy filtrov – *filter na DOM* a *filter na reťazec*.

## Adapcia vzorov

Vzory pracujú s viacerými druhmi filtrov. Tieto filtre pristupujú k spracovávanému dokumentu rôznymi spôsobmi, a preto musia vzory vstupný dokument upraviť do tvaru vhodného na spracovanie daným filtrom. Takéto zobrazenie dokumentu medzi tvarmi vhodnými pre rôzne filtre a tým schopnosť spracovávať rôzne tvary dokumentu musia zabezpečiť metódy adaptácie vzorov.

### 6.6.3 Učenie

Každý vzor musí vybrať zo vstupného dokumentu práve tie prvky, ktoré si používateľ žiada. Spôsobom, akým sa vzoru odovzdajú požiadavky používateľa na výber elementov, bude učenie. Použijeme *učenie s učiteľom* (angl. supervised learning) – učenie sa pomocou pozitívnych a negatívnych príkladov.

Príkladom je časť dokumentu, ktorú špecifikoval používateľ. Vzor obdrží príklad od komunikačného modulu. Tento príklad môže byť:

- *pozitívny príklad* – predstavuje elementy, ktoré používateľ chce mať vo výstupnom dokumente, a teda sa musia vo výbere nachádzať
- *negatívny príklad* – predstavuje elementy, ktoré používateľ nechce mať vo výstupnom dokumente, a teda sa vo výbere nesmú nachádzať

Po odoslaní prvej sady pozitívnych a negatívnych príkladov systém začne generalizovať výber, čiže sa snaží získať čo najvšeobecnejšie kritérium výberu (t.j. riešenie, ktoré vo výbere zahŕňa čo najviac inštancií), pričom musia byť dodržané požiadavky kladené množinou pozitívnych a negatívnych príkladov.

Systém má byť pružný aj z hľadiska spôsobu učenia, aby bolo možné na učenie použiť rôzne metódy. Dosiachnutie splnenia tejto požiadavky bude zabezpečovať *stratégia učenia*, ktorú bude možné určiť / nastaviť pomocou rozhrania vzoru.

### Spravovanie príkladov

Systém má umožňovať prezeranie zadaných príkladov, ich editovanie, prípadne odstránenie nesprávnych príkladov alebo príkladov zadaných do zlej skupiny (pozitívne, negatívne). V dôsledku toho si musí každý vzor pamätať všetky príklady, ktoré boli zadané pre jeho filtre. Ďalej musí uchovávať aj priradenie jednotlivých príkladov k filtrom, ktoré boli pomocou nich odvodené, aby bolo možné jednoduchšie vykonávať editáciu filtrov a ďalšie narábanie s nimi.

### Odstraňovanie príkladov

Pri odstránení príkladu sa musia vykonať zmeny vo všetkých filtroch, ktoré boli týmto príkladom ovplyvnené. V prístupe k tejto situácii sa naskytujú 2 možnosti riešenia:

- *Roll-back* – t.j. vrátenie vzoru do stavu, v akom sa nachádzal pred vložením odstraňovaného príkladu. Táto možnosť by zahŕňala odstránenie všetkých príkladov zadaných po aktuálne odstraňovanom príklade a tiež odstránenie všetkých filtrov, ktoré boli vytvorené na základe

týchto príkladov.

- *Delete* – dôjde len k odstráneniu požadovaného príkladu. Po odstránení tohoto príkladu dôjde k opätovnému učeniu všetkých filtrov, aby sa vzor dostal do takého stavu, v akom by bol, ak by sa odstránený príklad nebol vôbec zadal.

## Stratégie učenia

V prístupe k učeniu existujú viaceré stratégie. V navrhovanom systéme budú použité minimálne dva:

1. Prvou stratégiou je jednoduchá stratégia učenia určená na otestovanie funkčnosti prototypu. Bude založená na zovšeobecňovaní XPath výrazu, ku ktorému sa bude pristupovať ako k reťazcu. Napríklad, majme HTML zápis tabuľky:

```
<HTML>
<BODY>
<H1> ... </H1>
<TABLE>
<TR>
<TD>Cislo</TD>
<TD>Meno</TD>
</TR>
<TR>
<TD>1</TD>
<TD>Jano</TD>
</TR>
...
```

Ak používateľ vyberie 2 pozitívne príklady, ktorými budú elementy tabuľky „Cislo“ a „1“, vzor obdrží XPath výrazy:

```
/HTML/BODY/TABLE/TR[1]/TD[1]
/HTML/BODY/TABLE/TR[2]/TD[1]
```

Po zovšeobecnení by mal vzniknúť výraz:

```
/HTML/BODY/TABLE/TR/TD[1]
```

čím by vznikol výber 1. stĺpca tabuľky.

2. Ďalšou stratégiou môže byť priamo implementácia, prípadne upravená implementácia algoritmu učenia použitím zložených filtrov [8].

Tento algoritmus charakterizuje elementy pomocou referencií na iné elementy. Referencie sú reprezentované pomocou *cesty* od referenčného uzla k referencovanému (používa sa cesta v HTML strome). Cesta sa skladá z časti smerujúcej ku koreňu (po spoločný uzol oboch uzlov) a časti smerujúcej od tohoto uzla k referencovanému.

Algoritmus pri definícii filtra využíva *testy* (na prítomnosť atribútu, na jeho hodnotu, test na koreňový element, ...).

Z jednoduchých filtrov sa vytvárajú zložené ich spájaním pomocou logických operátorov  $\wedge$  (konjunkcia),  $\vee$  (disjunkcia),  $\neg$  (negácia). Napríklad pre filtre  $C$ ,  $C_1$ ,  $C_2$  a  $C_3$ :  $C = C_1 \wedge C_2 \wedge (\neg C_3)$ .

Algoritmus:

- Každý príklad sa popíše pomocou jednoduchého filtra. Popisujú sa aj pozitívne aj negatívne príklady, pričom sa vyznačí, či ide o pozitívny alebo negatívny.
- Na základe toho, ako jednotlivé filtre pokrývajú dokument sa urobí redukcia počtu filtrov, ak niektorý pokrýva aj prípady pokryté iným, alebo, ak sú filtre ekvivalentné.
- Zo vzniknutej skupiny filtrov sa vytvorí zložený filter, ktorého tvorba sa dá zredukovať na učenie disjunktívnej normálnej formy. Takto dostaneme výsledný naučený filter.

## 6.7 Aplikačný modul – Tvorba obalovača

Na základe vstupov od používateľa získaných z prezentačných modulov (prehliadača a aplikácie) modul na tvorbu obalovača vytvorí požadovanú inštanciu obalovača, ktorý pozostáva z:

- vykonateľných akcií
- relácií medzi akciami
- vstupných premenných
- štartovacej akcie
- HTTP klienta a parser dokumentu

Definície akcií, relácií medzi akciami a vstupných premenných sa využijú na tvorbu XML dokumentu, ktorý opisuje obalovač a následne sa využije rozhranie parsera programu obalovača nástroja Wrapper Suite (kapitola 2.3). Rozhranie parsera programu poskytuje možnosť vytvorenia inštancie obalovača na základe jeho opisu vo forme XML súboru. Posledným krokom v procese tvorby obalovača je nastavenie zvyšných charakteristík obalovača (výber štartovacej akcie a určenie HTTP klienta a parsera dokumentu).

### 6.7.1 Akcie obalovača

Akcie predstavujú elementárne operácie s presne definovanou činnosťou, ktoré pracujú nad lokálnym kontextom (časť 6.5). Akcie pozostávajú z 2 častí:

- typ akcie – navigačné, extrakčné akcie alebo odoslanie formulára
- atribúty akcie – vlastnosti akcie

#### Navigačné akcie

Navigačné akcie zabezpečujú prechod na stanovený dokument a jeho načítanie do lokálneho kontextu. Medzi navigačné akcie patrí:

- LoadPage
- FollowLink

### LoadPage

Načítanie požadovanej web stránky. Akciu charakterizujú nasledovné atribúty:

- *url* – adresa web stránky, ktorá sa má načítať
- *header* – hlavička HTTP požiadavky, ktorá je zaslaná na server s danou adresou

Akcia odošle HTTP požiadavku (s definovanou hlavičkou a URL adresou), pričom pri odosielaní požiadavky sa použijú cookies a autentifikačné údaje z kontextu. Následne sa vykoná transformácia dokumentu na validný XHTML dokument (ak je to možné) a uloží sa do dokumentu lokálneho kontextu ako objekt stromovej štruktúry (DOM), do cookies v kontexte sa uložia prijaté cookies.

### FollowLink

Zmena aktuálneho dokumentu na základe nájdeného odkazu na stránke. Akcia je určená atribútmi:

- *link* – určenie lokalizácie odkazu v dokumente pomocou identifikácie vzoru, ktorý vytvoril používateľ (modifikácia vzoru v lokálnom kontexte)
- *header* – hlavička http požiadavky, ktorá je zaslaná na server
- *repeat* – ohraničenie počtu nasledovaní odkazov

V aktuálnom dokumente sa lokalizuje požadovaný odkaz (určený atribútom *link*) a extrahuje sa z neho hodnota odkazu (adresa dokumentu). Ak atribút *link* nie je zadaný, akcia predpokladá, že v aktuálnom lokálnom kontexte sa bude nachádzať dokument, ktorého súčasťou je len požadovaný odkaz. Zo získanej hodnoty odkazu a adresy aktuálnej stránky sa vytvorí URL adresa a ďalej sa pokračuje rovnakým spôsobom ako pri akcii `LoadPage`. Používateľ má možnosť nastaviť počet opakovaní vykonávania nasledovania daného odkazu v dokumente prostredníctvom atribútu *repeat* (napr. pre prípad, že extrahované údaje sa nachádzajú v dokumentoch spojených odkazom „nasledujúci“). V takomto prípade sa zapamätá lokálny kontext obalovača po ukončení každej akcie `FollowLink`, ktorý bude predstavovať vstupný lokálny kontext nasledujúcej akcie `FollowLink` (pre ďalší počet nasledovania odkazu). Používateľ má tiež možnosť nastaviť neohraničený počet nasledovania odkazu, v takom prípade sa bude nasledovať odkaz, pokiaľ existuje.

### **Extrakčné akcie**

Extrakčné akcie zabezpečujú proces dolovania dát do štruktúrovaného výstupu. Prvým krokom extrakcie dát je aplikovanie filtra aktuálneho vzoru uloženého v lokálnom kontexte. Vzory uložené v lokálnom kontexte môže používateľ manuálne modifikovať, v tomto prípade sa do lokálneho kontextu akcie vloží vzor vytvorený používateľom. Pre všetky dáta, ktoré spĺňajú podmienky filtra (napr. všetky riadky tabuľky) sú vykonané nasledovné operácie:

1. získané dáta sú skonvertované do požadovaného formátu použitím regulárnych výrazov
2. uloženie dát do definovaného výstupného objektu, ak je to požadované

3. uloženie dát do definovaných premenných v kontexte, ak je to požadované
4. zmena aktuálneho dokumentu v lokálnom kontexte obalovača (dokument v lokálnom kontexte bude predstavovať extrahované dáta – strom s extrahovanými dátami)

Extrakčné akcie sú opísané nasledovnými atribútmi:

- *outputObject* – výstupný objekt (kontextový), ktorý určuje, kde sa majú extrahované dáta uložiť. Výstupný objekt je určený relatívnou cestou vo výstupnom strome pre daný obalovač vzhľadom na svojho predchodcu. Vo výstupnom objekte tiež možno určiť, či budú dáta uložené ako element alebo atribút
- *pattern* – identifikovanie vzoru v prípade, že používateľ vytvoril nový vzor pre danú akciu (modifikoval vzor uložený v lokálnom kontexte)
- *variable* – určenie premenných v kontexte, kde budú extrahované dáta uložené
- *format* – formát, do ktorého sa majú extrahované dáta transformovať, určený pomocou regulárnych výrazov

## Odoslanie formulára

Jedinou definovanou akciou, ktorá slúži na odoslanie formulára je akcia `SubmitForm`.

### SubmitForm

Odoslanie formulára, ktoré je špecifikované nasledovnými atribútmi:

- *form* – určenie formulára v dokumente pomocou identifikácie vzoru, ktorý vytvoril používateľ (modifikácia vzoru v lokálnom kontexte)
- *inputs* – parametre, ktorých hodnoty sú odoslané vo formulári. Parametre predstavujú premenné v kontexte
- *header* – hlavička HTTP požiadavky, ktorá je zaslaná na server

V aktuálnom dokumente sa lokalizuje požadovaný formulár (určený atribútom *form*) a z atribútu nájdeného formulára sa extrahuje odkaz a použitá metóda odoslania požiadavky (GET, POST). Ak atribút *form* nie je zadaný, akcia predpokladá, že v aktuálnom lokálnom kontexte sa bude nachádzať dokument, ktorého súčasťou je len požadovaný formulár. Z hodnoty získaného odkazu spracovania formulára a adresy aktuálnej stránky sa vytvorí URL adresa, vytvorí sa HTTP požiadavka so získanou metódou (GET, POST) a do vytvorenej požiadavky sa vloží hlavička (*header*) a obsah formulára (*inputs*). Vytvorená požiadavka sa následne odošle na server.

### 6.7.2 Ostatné charakteristiky obalovača

Po vytvorení akcií obalovača je potrebné vytvoriť relácie medzi akciami (stromovú štruktúru akcií), kde jedinou možnou reláciou je určenie nasledujúcich akcií, ktoré sa majú vykonať po skončení aktuálnej akcie. Týmto spôsobom sa vytvorí poradie vykonávania akcií. Zároveň je potrebné určiť štartovaciu akciu (koreň stromu akcií).

Na získanie dokumentov z webových systémov je potrebná komunikácia s webovými servermi prostredníctvom protokolu HTTP, ktorá je zabezpečená prostredníctvom existujúcich

knižníc prístupných pomocou rozhrania HTTP klienta v obalovači.

Úlohou parsera dokumentu je transformovať získaný HTML dokument DOM reprezentácie dokumentu.

## 6.8 Výstupné objekty

V súčasnej verzii aplikačného rámca predstavuje tvorba výstupných objektov samostatnú akciu. Pre zefektívnenie aplikácie sme sa rozhodli pre integráciu tohto procesu do akcie extrahovania dát. Počas vykonávania extrakčnej akcie (počas dolovania dát) sa bude priebežne vytvárať výstupný strom zo získaných údajov, pričom koreňový element je špecifikovaný v doménovej oblasti. Vetvy stromu sú určené relatívnou cestou vzhľadom na predchodcu. Získané dáta podľa svojho charakteru a potreby môžu byť ukladané nie len ako elementy, ale aj ako atribúty elementu.

Výstupné objekty budú ukladané do vyrovnávacej pamäte a odovzdávané jednotlivým zapisovačom, ktoré ich fyzicky zapíšu do konkrétneho cieľového prostredia (viac informácií v časti 2.3.6).

## 6.9 Implementačné prostriedky a vývojové nástroje

Vzhľadom na to, že systém na tvorbu obalovačov stavíme na existujúcom systéme, máme množstvo výhod, ale aj obmedzení. Medzi obmedzenia patrí najmä výber implementačných prostriedkov a vývojových nástrojov. Na druhej strane však vieme, že tieto prostriedky boli analýzou zvolené za vhodné a môžeme sa sústrediť na nové nástroje, ktoré sú potrebné na splnenie nových cieľov produktu. Prostriedky a nástroje existujúceho systému, ktoré použijeme:

- systém na správu verzií SVN
- platforma JDK 1.5, na ktorej bol systém vyvíjaný
- vývojové prostredie Eclipse, systém na správu verzií udržiava projekt pre toto vývojové prostredie vždy aktuálnym
- Jakarta commons HTTP klient - nie je potrebné modifikovať spôsob práce systému s protokolom HTTP
- HTML parser NekoHTML - použijeme existujúci modul na kontrolu a automatickú opravu chýb HTML dokumentov
- XML parser Xerces - štruktúru dokumentov nie je potrebné meniť
- XML spracovávanie dokumentov pomocou nástroja Jaxon
- balík na podporu Unit testovania v prostredí Java - naviažeme na vytvárané testy
- systém Maven2 na automatizovanú distribúciu, tvorbu balíkov, dokumentácie, štatistik.
- k týmto existujúcim prostriedkom sme pridali iba niekoľko rozšírení vo vývojovom prostredí, ktoré pomôžu členom tímu, ktorí sa s niektorými nástrojmi ešte nestretli:
  - Subclipse a JavaSVN – rozšírenia nástroja Eclipse, ktoré podporujú prácu s verzionovacím systémom SVN

- Maven 2.0 integration – rozšírenie integrujúce podporu systému Maven2 do prostredia Eclipse



## 7 Prototyp riešenia

V tejto časti dokumentácie je opísaný prototyp navrhovaného riešenia aplikácie tvorby obalovačov. V jednotlivých častiach sú rozobraté ciele, ktoré sme prototypom sledovali (časť 7.1), implementované zmeny v jadre (časť 7.2), implementované stratégie učenia (časť 7.3) a zmeny týkajúce sa používateľského rozhrania (časť 7.4).

### 7.1 Ciele

Náš produkt nadväzuje na prácu minuloročného tímu, ktorý vytvoril aplikačný rámec na tvorbu obalovačov. Cieľom nášho prototypu je rozšírenie existujúceho systému v nasledovných bodoch:

- úprava jadra modulu *Tvorba vzorov*, konkrétne:
  - overenie návrhu učenia sa obalovača
  - návrh a implementácia vybraných tried modulu
  - tvorba rozhraní
- implementácia jednoduchých stratégií učenia sa
- úpravy v *Prezentačnom* module aplikácie – doplnenie potrebných súčastí

Hlavným cieľom, ktorý sme si stanovili v prototypu, je overenie návrhu učenia sa obalovača. V súčasnosti sú dáta zo stránky získané pomocou filtra (určený XPath výrazom), ktorý musel používateľ presne zadefinovať. Naším cieľom je prídanie učenia pomocou pozitívnych príkladov. Pozitívne príklady od používateľa sú generalizované do filtra, ktorého podmienkou je, že aplikovaním na dokument musí vrátiť všetky zadané príklady (pozitívne). Táto požiadavka je realizovaná v module *Tvorba vzorov*, kde sa pri vytvorení novej akcie, ktorá pri svojej činnosti vyžaduje lokalizovať dáta na stránke, vytvorí vzor s priradenou stratégiou učenia sa (o stratégií rozhodne používateľ).

V module *Tvorba vzorov* bolo ďalej potrebné vytvoriť triedy, ktoré budú zabezpečovať požadovanú funkcionálnosť:

- filter – v súčasnosti XPath filter
- príklady – pozitívne a negatívne príklady potrebné pri vytváraní filtra
- subdokument – výsledok filtrovania aktuálneho dokumentu, v súčasnosti XML subdokument
- vzor – zahŕňa filter a stratégiu učenia, ktorá generuje filter
- stratégia učenia – v prototypu implementované 2 stratégie učenia:
  - filter priamo vytvorený používateľom
  - filter naučený z pozitívnych príkladov

Keďže v požiadavke na systém je umožnenie viacerých stratégií učenia a tiež viacerých

typov filtrov, triedy pre filter a stratégiu učenia sú reprezentované ako rozhrania. V prototypy sú vytvorené triedy, ktoré implementujú tieto rozhrania (XPath filter implementuje filter a stratégia učenia je implementovaná dvoma triedami, ktoré realizujú stanovené stratégie učenia). Keďže do konečnej aplikácie implementujeme aj filter regulárnych výrazov nad reťazcami, je potrebné transformovať dokument v XML formáte na reťazec. V prototypy sme sa preto zaoberali aj možnosťami existujúcich riešení transformácie dokumentov (adaptačné rámce).

Vzhľadom na vytvorené zmeny v existujúcom systéme je potrebné aj rozšírenie prezentačného modulu:

- umožnenie voľby stratégie učenia pri vytváraní akcie,
- rozhranie na pridávanie príkladov pri učení pomocou pozitívnych príkladov.

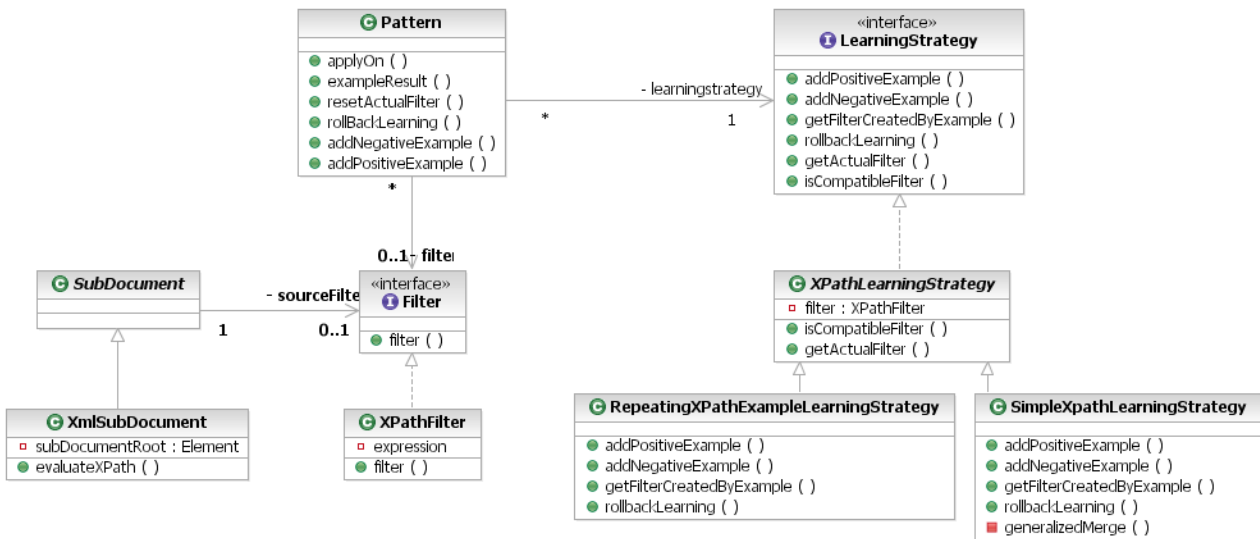
Umožnenie voľby stratégie učenia sa je implementované v existujúcom rozhraní vytvárania a modifikovania akcií obalovača. Na základe voľby používateľa sa danej akcii priradí do vzoru zodpovedajúca stratégia učenia. Rozhranie na pridávanie príkladov je riadené kontrolórom, ktorý na základe požiadavky používateľa na pridanie príkladu vytvorí dialóg a po zadaní príkladu pridáva do vzoru akcie vytvorený príklad. Vo finálnej verzii budú príklady v kontrolóri získavané cez komunikačný modul, a preto kontrolór predstavuje rozhranie, ktoré je v prototypy implementované triedou na riadenie pridávanie príkladov z dialógu *Prezentačného* modulu aplikácie.

## 7.2 Zmeny v jadre

Keďže projekt vychádza z existujúceho rámca na tvorbu obalovačov, prototyp riešenia integrujeme priamo do tohto prostredia. Cieľom prototypu je ukážka alebo naznačenie prístupu, ktorý chceme použiť pri realizácii učenia vzorov, konkrétne vzorov typu XPath. Implementácia prototypu si preto vyžaduje niektoré nevyhnutné úpravy jadra existujúceho rámca na tvorbu obalovačov.

### 7.2.1 Návrh modulu „Tvorba vzorov“

Učenie vzorov realizuje modul *Tvorba vzorov* (pozri časť 6.1). Keďže v neskorších fázach projektu je potrebné kompletne zmeniť existujúce jadro systému, modul *Tvorba vzorov* musí byť navrhnutý tak, aby bolo možné tento modul použiť čo najjednoduchším spôsobom v existujúcom systéme (ukážka prototypu) a rovnako aj v novom systéme. Logický diagram tohto modulu sa nachádza na Obr. 7-1.



Obr. 7-1: Diagram tried modulu „Tvorba vzorov“ (neobsahuje signatúry metód)

Opis jednotlivých entít diagramu:

- **Pattern** – fasáda nad filtrom a stratégiou učenia. Vzor deleguje príklad na stratégiu učenia, ktorá na základe príkladu vytvorí nový filter.
- **Filter** – rozhranie, ktoré reprezentuje komunikáciu s filtrom. Obsahuje metódu filter.
- **XPathFilter** – konkrétny filter. Filtrovanie realizuje aplikovaním XPath výrazu na uzol v DOM strome.
- **LearningStrategy** – rozhranie, ktoré reprezentuje stratégiu učenia. Stratégia učenia umožňuje pridávať pozitívne a negatívne príklady, udržiava históriu učenia (vie vrátiť filter, ktorý bol naučený vstupným príkladom) a vie zistiť, či je filter kompatibilný so stratégiou učenia.
- **XPathLearningStrategy** – abstraktná stratégia učenia, ktorá sa učí filtre typu XPathFilter.
- **RepeatingXPathExampleLearningStrategy** – stratégia učenia, obsahuje učenie sa filtra typu XPath tak, že naučený filter je iba kópiou vstupného pozitívneho príkladu. Stratégia si neudržiava históriu učenia a nevie sa učiť pomocou negatívnych príkladov.
- **SimpleXPathLearningStrategy** – stratégia učenia, obsahuje učenie sa filtra typu XPath tak, že zadaný príklad sa generalizuje s doteraz naučeným filtrom. Stratégia spracováva len pozitívne filtre. Udržiava si históriu príkladov a aj z nich vytvorených filtrov.
- **SubDocument** – abstraktná trieda, ktorá reprezentuje subdokument. Subdokument obsahuje filter, ktorým bol vytvorený.
- **XmlSubDocument** – subdokument, ktorý reprezentuje uzol v DOM strome.

## 7.2.2 Integrácia modulu „Tvorba vzorov“ do jadra systému

Úlohou modulu *Tvorba vzorov* je vytváranie nových vzorov pomocou príkladov, a teda vytváranie nových filtrov a filtrácia subdokumentov pomocou týchto filtrov. Podobná zodpovednosť prislúcha v existujúcom systéme triede `TagFinder`. Trieda `TagFinder` si udržuje XPath výraz vo forme reťazca (filter) a filtruje prichádzajúce DOM dokumenty, takže vracia DOM uzly, ktoré vzniknú aplikovaním XPath výrazu na vstupný dokument.

Trieda `TagFinder` je silne zviazaná s existujúcim jadrom systému, preto jej nahradenie nie je vhodné a vykoná sa až v ďalšej fáze projektu. Existujúci `TagFinder` bol upravený tak, že jeho XPath výraz bol nahradený vzorom, pričom filtrácia je delegovaná na vzor.

## 7.3 Stratégie učenia

Hlavnou úlohou stratégie učenia v rámci modulu „Tvorba vzorov“ je vytvoriť filter pre daný vzor. Filter sa vytvára postupne na základe zadávaných príkladov. Musí byť vytvorený tak, aby subdokument (prípadne subdokumenty), ktoré sa pomocou neho získajú z pôvodného dokumentu, pokrývali všetky pozitívne a žiadne negatívne príklady.

K úlohám stratégie učenia sme pridali aj možnosť získať filter, ktorý bol vytvorený na základe zadaného príkladu a možnosť vrátiť učenie do stavu, v akom bolo pred zadaním určitého príkladu.

Stratégia učenia charakterizuje, akým spôsobom sa bude filter vytvárať, čiže na základe tých istých príkladov môžu dve rôzne stratégie vytvoriť dva rôzne filtre. Pre výsledný produkt sme špecifikovali požiadavku, aby podporoval viaceré stratégie učenia. Prototyp obsahuje 2 stratégie: opakujúcu stratégiu učenia sa XPath príkladov a jednoduchú stratégiu učenia sa XPath príkladov. Obe zatiaľ pracujú iba s pozitívnymi príkladmi.

### 7.3.1 Opakujúca stratégia učenia XPath príkladov (RepeatingXPathExampleLearningStrategy)

Ide o stratégiu použiteľnú na spracovanie príkladov zadávaných pomocou XPath výrazu. Je schopná pamätať si len jeden zadaný príklad, ktorý potom poskytuje ako vytvorený filter. Jej cieľom je poskytovať filter, ktorý vyberá z dokumentu práve tie elementy, ktoré vybral používateľ. Uplatní sa v ďalšej etape projektu, konkrétne v časti vytvárania prepojenia medzi internetovým prehliadačom a aplikáciou. Tu posluží na overenie správneho odovzdávania príkladov z prehliadača a ich správneho zobrazovania naspäť v prehliadači.

#### Opis činnosti:

- **pridanie pozitívneho príkladu** – stratégia zadaný príklad nijak nespracováva, iba si ho zapamätá a tento príklad (XPath výraz) poskytuje ako filter
- **pridanie negatívneho príkladu** – stratégia zadanie príkladu ignoruje, t.j. dodaný príklad si nikam nezaznamená a naďalej poskytuje doterajší filter

- **vrátenie filtra vytvoreného pomocou zadaného príkladu** – ak sa príklad zhoduje s filtrom, vráti ten istý filter (ktorý je v tomto prípade totožný s príkladom), inak vráti null – čiže, na základe zadaného príkladu nebol vytvorený filter
- **návrat v učení** – ak je zadaný príklad zhodný s filtrom, odstráni sa filter. Inak sa používa aj naďalej doterajší

### 7.3.2 Jednoduchá stratégia učenia XPath príkladov (SimpleXPathExampleLearningStrategy)

Stratégia predstavuje jednoduchý spôsob učenia. Zo zadaných príkladov sa usiluje vytvoriť filter, ktorý ich bude všetky zahŕňať (generalizovať). Keďže spracováva iba pozitívne príklady, stačí pri zadaní každého príkladu iba zovšeobecňovať doterajší filter.

#### Opis činnosti:

- **pridanie pozitívneho príkladu** – stratégia zovšeobecni doterajší filter pomocou zadaného príkladu, čím dostane nový filter, ktorý bude ďalej poskytovať. Ak sa pridáva príklad, ktorý sa už predtým pridal, ignoruje sa, pretože by aj tak výsledný filter neovplyvnil.
- **pridanie negatívneho príkladu** – pridávaný negatívny príklad sa ignoruje (podobne ako u predchádzajúcej stratégie)
- **vrátenie filtra vytvoreného pomocou zadaného príkladu** – ak bol zadaný príklad už pri učení použitý, vráti sa filter, ktorý vtedy pomocou neho vznikol. Ak nebol použitý, vráti sa null.
- **návrat v učení** – keďže stratégia použije každý príklad práve raz, dá sa jednoznačne na základe príkladu určiť filter, ktorý mala stratégia pred použitím tohoto príkladu. A pretože stratégia udržuje históriu použitých príkladov a filtrov, ktoré pomocou nich vznikli v časovom poradí, môže sa učenie vrátiť do stavu pred naučením zadaného príkladu.

#### Princíp zovšeobecňovania

Do zovšeobecňovania vstupujú dva XPath výrazy. Stratégia s nimi pracuje ako s reťazcami, pričom sa berie do úvahy, že ide o XPath výrazy. Východiskom je rozdelenie oboch reťazcov na elementy podľa oddeľovača. Hlavnou myšlienkou je na základe týchto dvoch výrazov vytvoriť nový výraz, ktorý pokrýva (aj) všetky elementy, ktoré pokrývali tieto výrazy. Zároveň sa snažíme neurobiť výsledný výraz príliš všeobecný.

Základným princípom je prechádzanie elementov oboch reťazcov od ich začiatku a nájdenie najdlhšej spoločnej časti, pričom jednotkou pri tomto porovnaní je práve element. Nasledujúca ukážka predstavuje zovšeobecnenie dvoch výrazov do tretieho (pod čiarou):

/HTML/BODY/TABLE[1]/TR[1]/TD[2]/P

/HTML/BODY/TABLE[1]/TR[1]/TD[2]/IMG

---

/HTML/BODY/TABLE[1]/TR[1]/TD[2]

Táto základná myšlienka je v tejto stratégii doplnená o ďalšie zovšeobecnenie, a to

odstránením predikátu. Aby sa obmedzilo nadbytočné zovšeobecnenie v tomto smere, odstraňujú sa iba rozdielne predikáty.

```
/HTML/BODY/TABLE[1]/TR[1]/TD[2]/
```

```
/HTML/BODY/TABLE[1]/TR[2]/TD[2]/
```

---

```
/HTML/BODY/TABLE[1]/TR/TD[2]
```

## 7.4 Používateľské rozhranie

Ako používateľské rozhranie sa v našom prototypy použilo rozhranie systému, ktoré vytvorili členovia minuloročného tímu. Toto rozhranie bolo modifikované a doplnené o ďalšie ovládacie prvky, ktoré používateľovi sprístupňujú funkcionality stratégie učenia a tvorby vzorov.

Prvá zmena používateľského rozhrania sa týka zmeny dialógových okien akcií `FollowLink`, `ExtractData`, `ForEachTag` a `DoWhileNextLink`. Tieto dialógové okná umožňovali jednotlivým akciám vkladať regulárne výrazy, ktoré slúžili pri hľadaní značiek v dokumente. Funkciu regulárnych výrazov sme v prototypy nepoužili, a preto sme odstránili ovládacie prvky, ktoré tieto výrazy umožnili vkladať. Naopak v týchto dialógoch pribudla položka, ktorá zobrazuje zoznam všetkých dostupných stratégií učenia vzorov a umožňuje výber jednej z nich. Vybraná stratégia sa potom použije pri tvorbe vzoru (`RepeatingLearningStrategy` – filter je zadaný priamo používateľom a `SimpleLearningStrategy` – jednoduché učenie generalizáciou pozitívnych príkladov).

Z funkcionality tvorby vzorov pri jednoduchom učení sa obalovača vznikla aj požiadavka na vkladanie pozitívnych a negatívnych príkladov vo forme XPath výrazov (v prototypy sa používajú len pozitívne príklady). Bolo potrebné vytvoriť dialógové okno, ktoré používateľovi umožňuje vkladať tieto príklady a rozlišovať medzi nimi. Na vkladanie príkladu sa použil ovládaci prvok textové pole. Do tohto prvku používateľ môže zapísať požadovaný príklad. Musí taktiež určiť, o aký typ príkladu sa jedná (pozitívny alebo negatívny). Na to slúžia dva prepínače, z ktorých je vždy aktívny práve jeden (podľa typu príkladu, ktorý si používateľ vyberie). V našom prototypy využívame na učenie obalovača iba pozitívne príklady (vloženie negatívnych príkladov nemá žiadny dopad na učenie sa obalovača). Dialóg je ovládaný kontrolórom, ktorý na žiadosť používateľa otvára nový dialóg, získava údaje od používateľa a zabezpečuje vytvorenie nových príkladov, ktoré sú vložené do pozitívnych príkladov vzoru.

Dialógové okno vkladania príkladov sa používa len pri akciách `FollowLink`, `ExtractData`, `ForEachTag` a `DoWhileNextLink`, kde je potrebné zadávať príklady na tvorbu vzorov. Aktiváciu tohto okna je možné vykonať z menu aplikácie alebo z rolovacieho menu danej akcie (v položke menu `Add Example`).

Viac o používateľskom rozhraní je možné nájsť v prílohe A – Používateľská príručka prototypu.

## 8 Použité pramene

- [1] Kapow Technologies. *RoboSuite Quick Start Guide* [online]. Verzia 6.1. 2006 [cit. 2006-11-11]. Dostupné na: [http://kdc.kapowtech.com/documentation\\_6\\_1/robosuite/RoboSuiteQuickStartGuide.pdf](http://kdc.kapowtech.com/documentation_6_1/robosuite/RoboSuiteQuickStartGuide.pdf).
- [2] Kapow Technologies. *ModelMaker User's Guide* [online]. Verzia 6.1. 2006 [cit. 2006-11-11]. Dostupné na: [http://kdc.kapowtech.com/documentation\\_6\\_1/robosuite/ModelMakerUsersGuide.pdf](http://kdc.kapowtech.com/documentation_6_1/robosuite/ModelMakerUsersGuide.pdf).
- [3] Kapow Technologies. *RoboMaker User's Guide* [online]. Verzia 6.1. 2006 [cit. 2006-11-11]. Dostupné na: [http://kdc.kapowtech.com/documentation\\_6\\_1/robosuite/RoboManagerUsersGuide.pdf](http://kdc.kapowtech.com/documentation_6_1/robosuite/RoboManagerUsersGuide.pdf).
- [4] Lixto Software GmbH. *Lixto Visual Wrapper: User Manual*. Verzia 2.2. 2006. Dostupné ako súčasť inštalačného balíka Lixto Visual Wrapper.
- [5] BERTA, I., JANŽO, A., JEMALA, M. et al. *Analýza textov pracovných ponúk z prostredia webu*. Bratislava, 2006. Tímový projekt na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave. Vedúci projektu Mgr. György Frivolt. 158 s.
- [6] XUL Planet. *XPCOM Interfaces* [online]. 2006 [cit. 2006-11-11]. Dostupné na: <http://www.xulplanet.com/tutorials/xultu/xpcom.html>.
- [7] KOSEK, J. *XUL* [online]. 2002 [cit. 2006-11-13]. Dostupné na: <http://www.kosek.cz/clanky/xul/index.htm>.
- [8] CARME, J., CERESNA, M., GOEBEL, M.: *Web Wrapper Specification Using Compound Filter Learning*. [s.l.] : [s.n.], 2006. 8 s.
- [9] SZINEK, P. *W3C Mozilla DOM Connector will be released soon!* [online]. 2006 [cit. 2006-11-14]. Dostupné na: <http://www.rubyrailways.com/w3c-mozilla-dom-connector-will-be-released-soon/>.





## Príloha A: Používateľská príručka prototypu

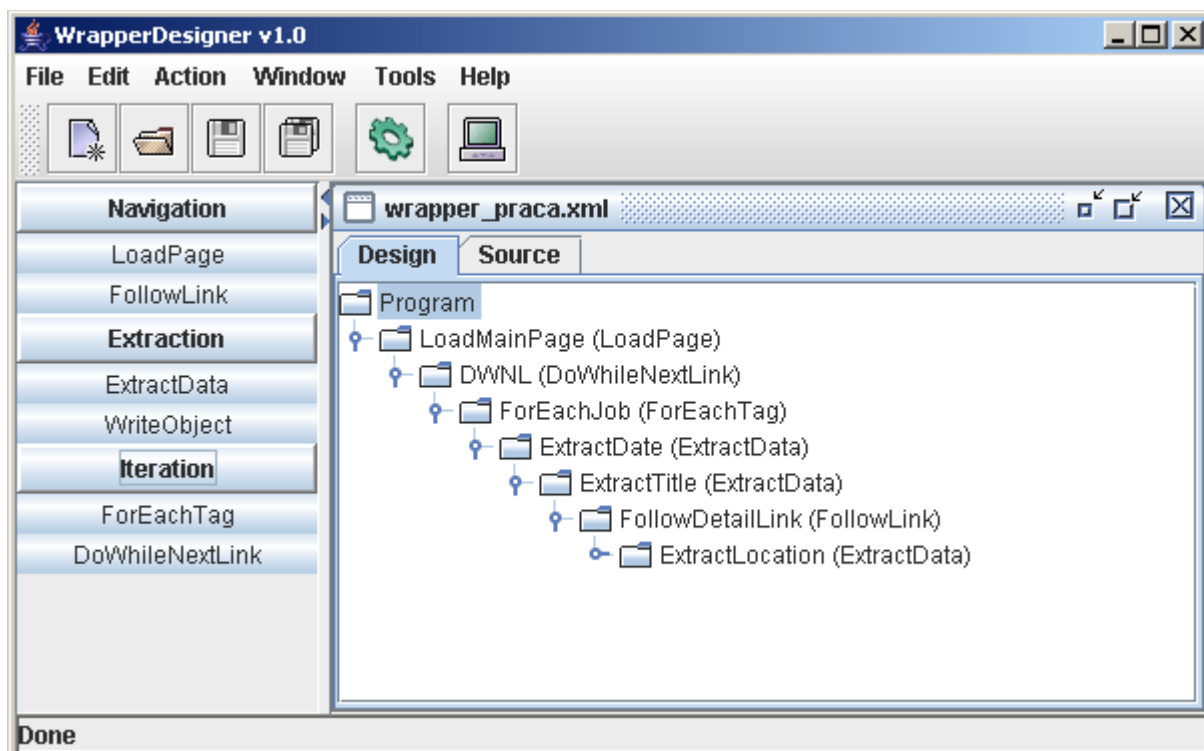
Prototyp aplikácie tvorby obalovača nadväzuje na aplikačný rámec WrapperSuite, ktorý bol implementovaný tímom č. 10 v minulom akademickom roku. Zmenená bola iba kľúčová funkcionálna, prostredníctvom ktorej realizujeme ciele, ktoré sme si vytýčili v časti 7.1. Keďže vlnajší aplikačný rámec obsahoval kompletnú príručku používateľa, v nasledujúcej časti kladieme dôraz najmä opis ovládania aplikácie súvisiaceho s funkcionálnosťou, ktorá úzko súvisí s demonštrovanými vlastnosťami prototypu.

### A.1 WrapperDesigner

Na interakciu s používateľom slúži prostredie WrapperDesigner. Je rozdelené do nasledovných funkčných celkov:

1. menu,
2. panel nástrojov,
3. panel obsahujúci jednotlivé akcie, ktoré môžu byť použité pri tvorbe obalovača,
4. návrhový a zdrojový pohľad na zdrojový súbor vytváraný obalovač,
5. interpret.

Snímok obrazovky je zobrazený na Obr. A-1.



Obr. A-1: Snímok obrazovky prostredia WrapperDesigner

### A.1.1 Menu

Menu prostredia WrapperDesigner je rozdelené na nasledovné položky:

- File – umožňuje prístup k príkazom pre prácu s aktuálne spracovávaným zdrojovým kódom obalovača,
- Edit – umožňuje prístup k príkazom pre prácu s textom umiestneným v schránke,
- Action – obsahuje možnosti práce s akciami, jednotlivé položky budú bližšie opísané v nasledujúcich častiach
- Window – umožňuje prístup k príkazom pre prepínanie medzi oknami jednotlivých otvorených zdrojových súborov obalovača,
- Tools – obsahuje nástroje aplikačného rámca WrapperSuite, ako aj možnosti nastavenia vlastností aplikácie
- Help – umožňuje prístup k oknu s informáciami o tejto aplikácii.

### A.1.2 Panel nástrojov

Možnosti panela nástrojov prakticky zodpovedajú najčastejšie používaným položkám menu. Išlo nám hlavne o pohodlné používateľské prostredie a intuitívne ovládanie aplikácie prototypu.



*Obr. A-2: Panel nástrojov prostredia WrapperDesigner*

Panel nástrojov prostredia WrapperDesigner obsahuje postupne (sprava doľava) položky umožňujúce prístup k nasledujúcim funkciám:

- Vytvorenie nového zdrojového súboru obalovača,
- Otvorenie existujúceho zdrojového súboru obalovača,
- Uloženie aktuálne spracovávaného zdrojového súboru obalovača
- Uloženie všetkých otvorených zdrojových súborov obalovačov,
- Interpretácia aktuálne spracovávaného zdrojového súboru obalovača,
- Zobrazenie konzoly umožňujúcej prehliadanie výstupu obalovača.

### A.1.3 Panel akcií

Panel akcií prostredia WrapperDesigner, ktoré môžu byť použité pri vytváraní obalovača je rozdelený do troch celkov, podľa typu akcie:

- navigačné akcie,
- extrakčné akcie,
- iteračné akcie.

Po kliknutí a rozbalení jednotlivých skupín akcií je možné priamo vybrať požadovanú akciu. V prípade navigačných akcií sú to nasledujúce:

- akcia LoadPage,
- akcia FollowLink.

V prípade extrakčných akcií ide o:

- akcia ExtractData,
- akcia WriteObject.

V prípade iteračných akcií sú to nasledujúce:

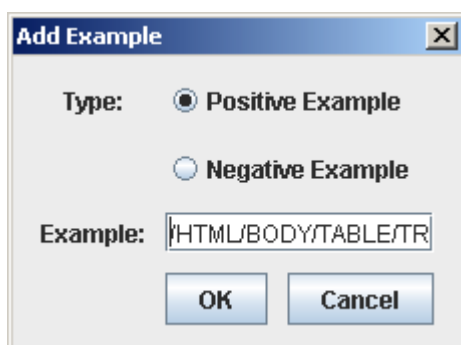
- akcia ForEachTag,
- akcia DoWhileNextLink.

Z pohľadu prototypu sú kľúčové akcie LoadPage, ForEachTag, ExtractData a WriteObject. Ich konkrétne použitie bude opísané v časti A.2.

#### A.1.4 Návrhový pohľad na zdrojový súbor – karta Design

Pri tomto pohľade je zdrojový súbor vytváraného obalovača zobrazený ako strom na seba nadväzujúcich, postupne vykonávaných akcií (pozri Obr.A-1 - kartu Design). V tomto pohľade je možné editovanie samotného programu obalovača. Editovanie realizujeme pridávaním alebo odstraňovaním akcií. Všetky úpravy je možné robiť nielen pomocou panela nástrojov alebo prostredníctvom menu, ale tiež použitím pravého tlačidla myši, čo zabezpečuje okamžitý prístup k vlastnostiam akcie, na ktorú sme klikli.

Z pohľadu prototypu je dôležité všimnúť si možnosť pridávania príkladov (položka menu Actions, resp. kliknutím pravého tlačidla a následne Add Example). Používateľ môže pre danú stratégiu učenia zadať pozitívny alebo negatívny príklad v podobe XPath výrazu (Obr. A-3).

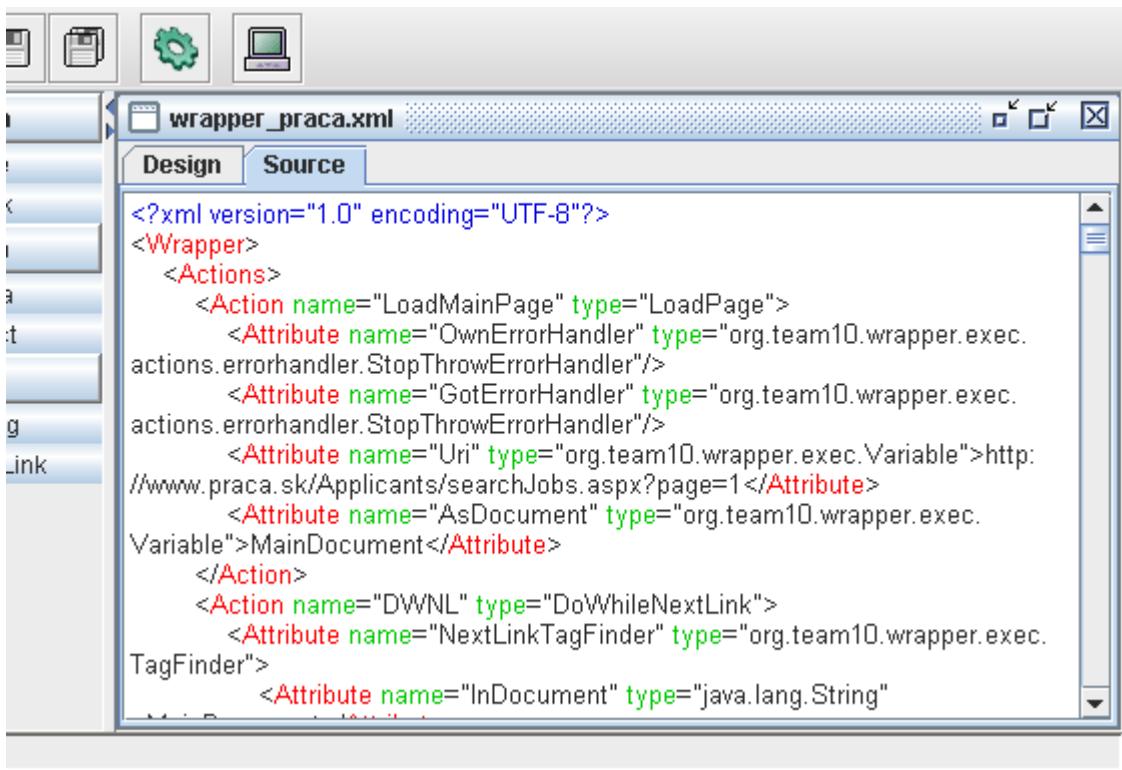


Obr. A-3: Pridanie príkladu

#### A.1.5 Zdrojový pohľad na zdrojový súbor – karta Source

Okrem návrhového pohľadu na vytváraný zdrojový súbor obalovača je možné prepnutie sa aj do karty Source (Obr. A-4), ktorá poskytuje pohľad priamo vytváraný XML súbor obalovača tak, ako bude uložený do výstupného súboru. Kvôli prehľadnosti a väčšej flexibilitě kontroly

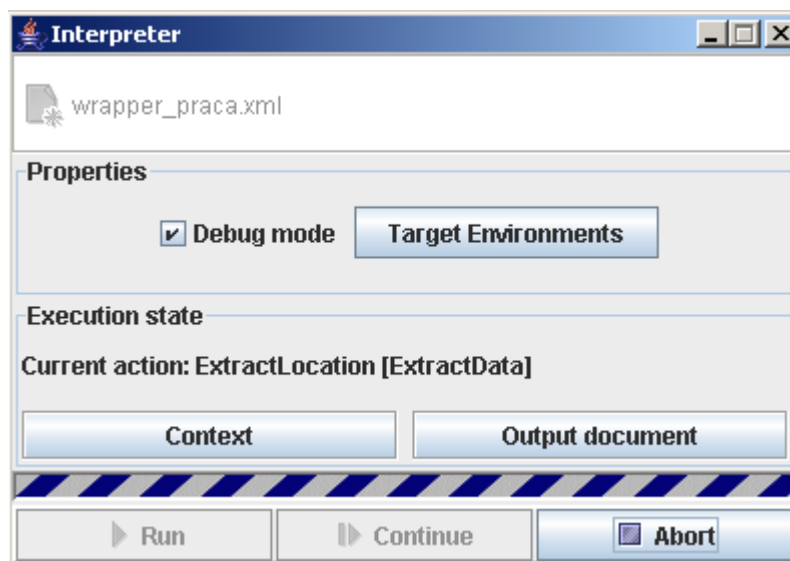
vytváraného zdrojového súboru používateľom bolo použité zvýraznenie syntaxe.



Obr. A-4: Snímok časti obrazovky, kde sa nachádza tzv. zdrojový pohľad na zdrojový súbor

### A.1.6 Interpreter

Okrem možnosti vizuálneho vytvárania a editovania zdrojového súboru obalovača poskytuje prostredie WrapperDesigner aj prostriedky pre interpretáciu takto vytvoreného zdrojového súboru (Obr. A-5).



Obr. A-5: Snímok obrazovky interpretera

Keďže práca s interpretom sa od predchádzajúcej verzie nezmenila, nemá zmysel rozoberať detaily možností použitia v prípade prototypu znovu.

## A.2 Demonštrácia prototypu

Nasledujúca časť opisuje postup použitia aplikácie WrapperSuite na docielenie a ukázanie takého správania, na ktorom bude možné demonštrovať splnenie cieľov, ktoré boli kladené na prototyp. Uvidíme, ako sa na praktickom príklade uplatní jednoduchá stratégia učenia sa.

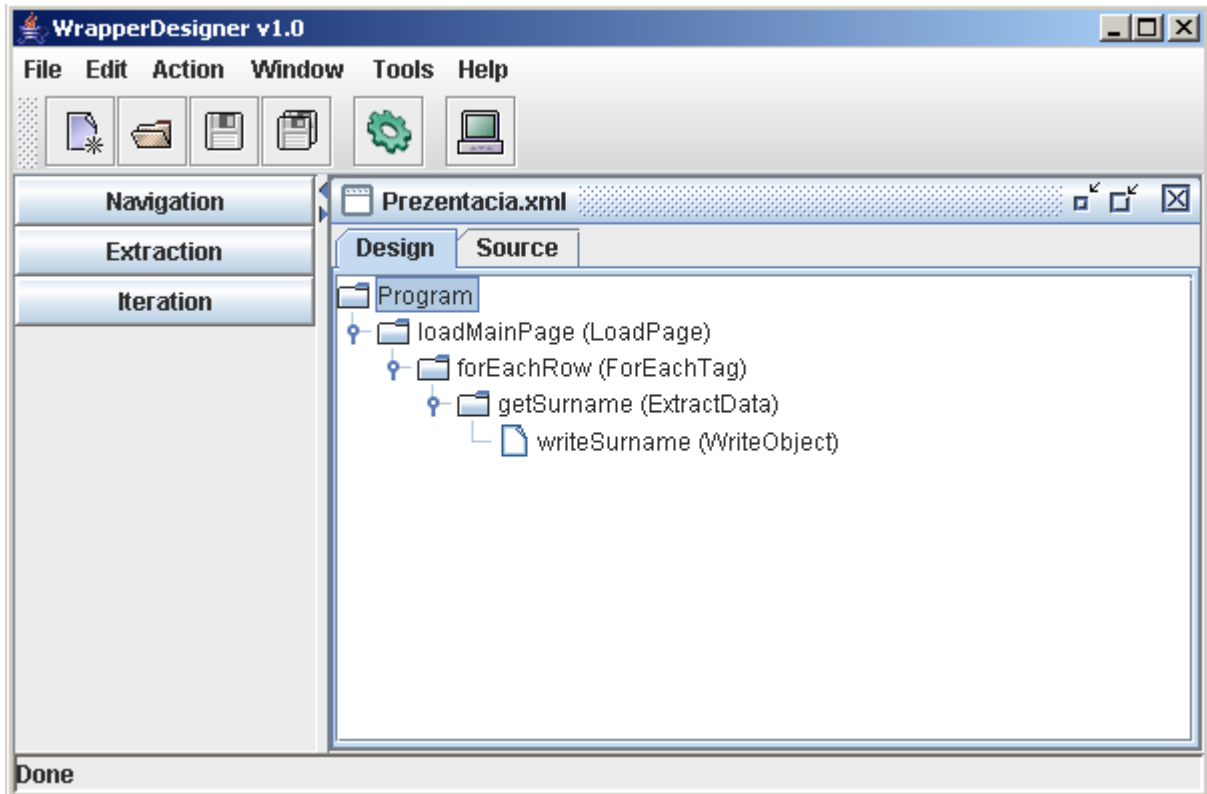
Postup vytvorenia obalovača a nasledovného spustenia učenia sa a extrahovania potrebných dát pozostáva z nasledujúcich krokov:

1. Vytvorenie nového obalovača kliknutím na ikonu New alebo výberom tejto položky z menu File
2. Pridanie navigačnej akcie LoadPage. Do zobrazeného dialógového okna je potrebné vložiť parametre akcie:
  1. Name – názov akcie: *loadMainPage*;
  2. URI – adresu nahrávanej stránky – *http://osiris.yweb.sk/tabulka.html* – webová stránka vytvorená pre testovacie účely
  3. asDocument – názov dokumentu, pod ktorým bude vystupovať v kontexte: *MainDocument*.

### Prezentácia učenia – najprv pomocou RepeatingXPathLearningStrategy:

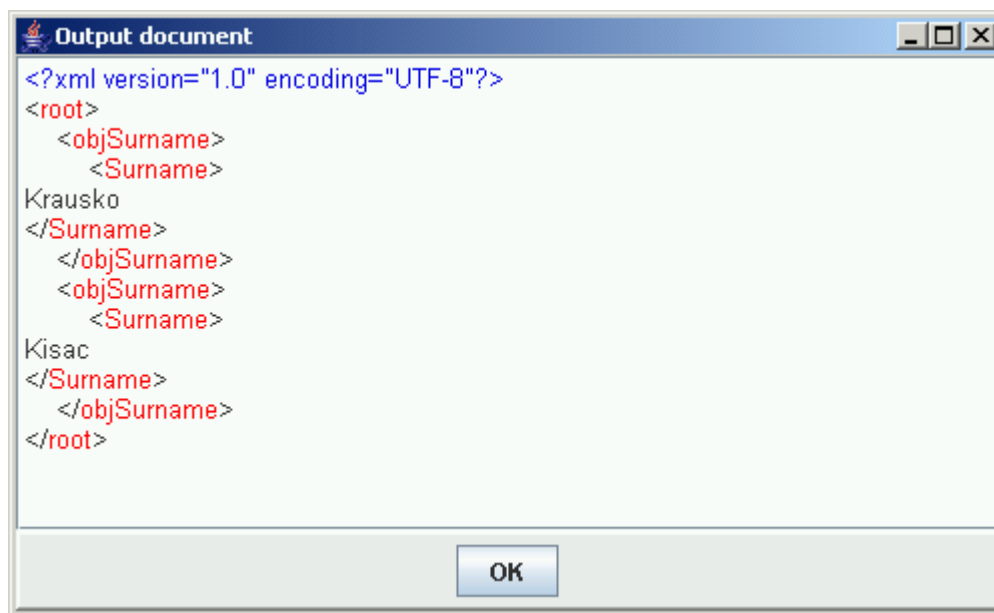
3. Pridanie iteračnej akcie ForEachTag. Do zobrazeného dialógového okna je potrebné vložiť parametre akcie:
  1. Name v karte General: *forEachRow*
  2. LearningStrategy v karte General: *RepeatingLearningStrategy*
  3. XPathExp v karte TagFinder: *necháme prázdne*
4. Pridanie extrakčnej akcie ExtractData. Do zobrazeného dialógového okna je potrebné vložiť parametre akcie:
  1. Name v karte General: *getSurname*
  2. LearningStrategy v karte General: *RepeatingLearningStrategy*
  3. inDocument v karte Source: *MainDocument*
  4. XPathExp v karte Source: *necháme prázdne*
  5. OutObject v karte Destination: *objSurname*
  6. OutObjectPath v karte Destination: *Surname*
  7. ContVariable v karte Destination: *necháme prázdne*
5. Pridanie extrakčnej akcie WriteObject. Do zobrazeného dialógového okna je potrebné vložiť parametre akcie:
  1. Name: *writeSurname*
  2. ObjectName: *objSurname*

6. Zadáme prvý pozitívny príklad.
  1. Kliknutím pravým tlačidlom myši na akciu *forEachRow*
  2. V kontextovom menu kliknutie ľavým tlačidlom na *Add Example*
  3. Typ príkladu: *Positive Example*.
  4. Do políčka Example: */HTML/BODY/TABLE/TR[2]/TD[2]*
  5. Potvrdenie OK  
Aktuálny stav v návrhovom pohľade by mal zodpovedať Obr. A-6.



Obr. A-6: Obaľovač používajúci stratégiu *RepeatingLearningStrategy*

7. Spustíme interpret. Spustíme obaľovač – tlačidlom Run.
8. Výstup skontrolujeme tlačidlom Output document, kde vidíme jedno priezvisko z druhého riadku a druhého stĺpca tabuľky.
9. Zatvorenie interpreta (tlačidlo Close) a dokumentu (tlačidlo OK).
10. Pridanie ďalšieho pozitívneho príkladu. Postupujeme ako v kroku 6, ale za Example zadáme: */HTML/BODY/TABLE/TR[3]/TD[2]*
11. Overenie výstupu – body 7., 8., 9. Tu vidíme, že pribudlo 1 priezvisko, teraz z riadku 3.  
Zvolená stratégia učenia teda nie je schopná generalizovať, filter prispôsobí iba poslednému zadanému príkladu (Obr. A-7).



Obr. A-7: Podoba Output Document-u po učení stratégiou *RepeatingLearningStrategy*

### Prezentácia učenia pomocou *SimpleXPathLearningStrategy*:

#### 12. Zmena stratégie učenia

1. Po pravom kliknutí na `forEachTag` a vybraním položky `Edit` sa otvorí dialógové okno, kde na karte `General` zmeníme `Learning Strategy` na `SimpleLearningStrategy`.
2. Potvrdíme pomocou `OK`.

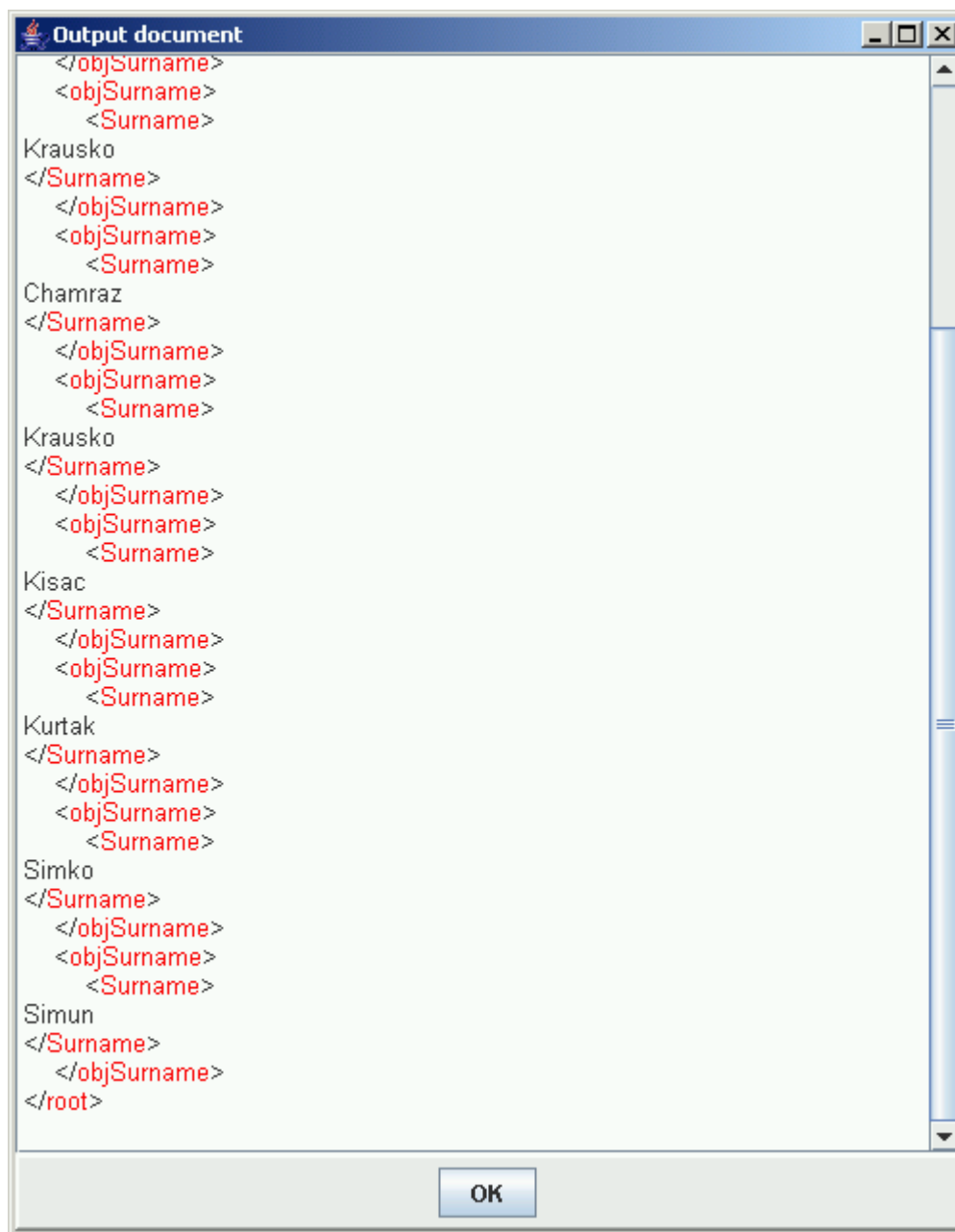
13. Zadanie prvého pozitívneho príkladu – rovnako ako v bode 6.

14. Overenie zopakovaním bodov 7., 8., 9. s rovnakým výsledkom.

15. Zadanie druhého pozitívneho príkladu – rovnako ako v bode 10.

16. Overenie ako v bode 11., ale do výsledku pribudol celý stĺpec s priezviskami.

Zvolená stratégia generalizovala 2 zadané príklady, z ktorých odvodila výber celého stĺpca (Obr. A-8).



Obr. A-8: Podoba Output Document-u po učení stratégiou *SimpleLearningStrategy*