УДК 004.428.4

А.Ю. Дорошенко, Л.Г. Молотієвський

ПРО УНІФІКАЦІЮ СПОСОБІВ ОБРОБКИ СТРУКТУРОВАНОЇ ІНФОРМАЦІЇ

Розглядаються проблеми пошуку та збору інформації з однотипних DOM-моделей. Запропоновано механізм збору структурованої інформації з релевантних джерел, що дозволяє будувати універсальні аналізатори даних з цих джерел.

Вступ

Інформація в всесвітній мережі Інтернет представлена переважно у вигляді сторінок, написаних мовою гіпертекстової розмітки HTML. Кожна сторінка має власну модель DOM, тобто об'єктну модель документу (від англ. Document Object Model[1]), яка є специфікацією прикладного програмного інтерфейсу для роботи зі структурованими документами.

З точки зору об'єктно-орієнтованого програмування, DOM визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленням документів у вигляді дерева. Все це призначено для того, аби надати комп'ютерній програмі доступ до DOM моделі документу та його структури, змісту та оформлення. Вибірка даних із такої структури можлива за допомогою запиту Xpath [2]. XPath (XML Path Language) – це мова виразів для адресації частин XML документу, або для обчислення величин (строкових, числових або булевих) на основі вмісту XML документа.

Основною проблемою аналізу сторінок є відсутність механізмів які дозволили б використовувати існуючі дані, для вибірки інформації з інших подібних (релевантних) сторінок. Релевантність – це властивість відповідності між бажаною інформацією та тою, що отримується з джерела.

В наш час всесвітня мережа Інтернет несе в собі велику кількість сервісів, що надають однакові послуги. З формальної точки зору, такі сервіси мають подібну структуру, функції та інтерфейс користувача. Будемо називати такі сервіси однотипними. Для прикладу достатньо розглянути кілька реальних однотипних сервісів, які займаються, оптовою торгівлею та надають сервіс пошуку туристичних турів: http://online.newstravel.com.ua/search_tour і http://online.coraltravel.com.ua/search_tour. Порівнюючи ці сервіси, можна зазначити, що вони:

- надають одні й ті ж самі послуги;
- мають подібні візуальні інтерфейси;
- мають одного й того ж розробника;
- мають однакові механізми створення запитів;
- несуть у собі взаємно релевантні дані.

Але, на жаль, ми не можемо використовувати один і той же XPath запит, який би дозволив отримати релевантні дані (наприклад, назву країни для подорожі), тому що DOM модель у кожного із сервісів – різна. Для створення раціонального підходу до розв'язання задачі уніфікації сервісів, потрібні механізми, які б дозволили автоматично адаптуватися до нових однотипних сервісів. З цією метою створюються аналізатори таких ресурсів. Проте існуючі механізми аналізу, такі як: BeautifulSoup, HTMLParser, Kizna HTML Parser та Jericho HTML Parser, часто не гнучкі та потребують доопрацювання при використанні для інших однотипних сервісів. Такий підхід веде до збільшення кількості аналізаторів.

Мета даної статті – оптимізація механізмів аналізу та вибірки інформації із однотипних сервісів шляхом побудови уніфікованих аналізаторів для таких сервісів. При написанні аналізаторів для однотипних сервісів виникають такі проблеми:

- аналіз подібних сервісів, потребує написання спеціальних механізмів вибірки даних, що часом мають складну структуру;
- перед написанням аналізатору, розробнику потрібно продивитися DOM модель, та скласти XPath запити для вибірки даних, що займає значний час;
- зміна DOM моделі джерела потребує зміну XPath запиту.

Метою системи збору інформації з однотипних інформаційних джерел є створення системи, яка може автоматично створювати XPath запити, використовувати вже існуючи дані для оновлення або створення нових аналізаторів однотипних інформаційних джерел.

1. Проблеми існуючих рішень

Існує досить багато рішень, які дозволяють аналізувати дані DOM, але більшість із них потребує ручного втручання розробника в процес аналізу. Розглянемо два існуючих рішення: Kizna HTML Parser та Jericho HTML Parser. Серед їхніх переваг слід відмітити:

- вибірка тексту з документу за запитом;
- вибірка з DOM електронних поштових адрес;
- захоплення ресурсів зі сторінки (картинки, Flash анімацію, JavaScript сценарії);
- обробка вихідних даних сторінок які показані на екрані, з метою виділити інтерфейсну інформацію. Основна відмінність такого виду аналізу від звичайного полягає в тому, що дані спочатку мали бути використані для сприйняття людиною, а не для машинної інтерпретації;
- перевірка Веб-посилань на сторінці;
- трансформація HTML сторінки в XML.

Головний недолік існуючих рішень полягає у тому, що вони орієнтовані лише на побудову та вибірку певної інформації з однієї сторінки а, отже, не забезпечують механізмів вибірки даних з релевантних сторінок.

2. Характеристика однотипних ресурсів

Для порівняння двох однотипних ресурсів потрібно дослідити код HTML сторінок які генеруються даними сервісами. Для прикладу візьмемо сервіси пошуку туристичних турів (http://online.coraltravel .com.ua/search_tour та http://online. newstravel.com.ua/search_ tour).

Виділимо групу релевантних елементів. Як приклад візьмемо країни для подорожі, та дослідимо HTML код даних елементів

Coraltravel:

```
<select name="STATEINC" class="STATEINC"
autocomplete="off">
<option value="104">China</option>
<option value="114">Cuba</option>
<option value="143">Dominican Repub-
lic</option>
<option value="58">Egypt</option>
<option value="95">India</option>
<option value="180">Indonesia</option>
<option value="181">Maldives</option>
<option value="182">Ukraine</option>
<option value="183">Korea</option>
<option value="96">Thailand</option>
<option value="51">Turkey</option>
<option value="151">UAE</option>
</select>
```

Newstravel:

```
<select name="STATEINC" class="STATEINC"
autocomplete="off">
<option value="25">Болгария</option>
<option value="27">Греция</option>
<option value="7">Кипр</option>
<option value="56">ОАЭ</option>
<option value="51">Таиланд</option>
<option value="8">Тунис</option>
<option value="54">Хорватия</option>
<option value="52">Черногория</option>
<option value="23">Чехия</option>
</select>
```

Порівнявши ці два коди, можна стверджувати, що вони несуть релевантні дані. Наприклад країна ОАЕ присутня в обох списках. Назва та клас даного компоненту сторінки однакові. Це трапляється в тому випадку, коли обидві пошукові сис-

теми були розроблені одним и тим самим розробником.

3. Засоби для аналізу

Для побудови DOM моделі будемо використовувати HTML Agility Pack [5]. Ця бібліотека несе в собі можливості які дозволяють:

- перетворювати будь який правильний HTML документ в DOM модель;
- для кожного елемента DOM моделі створюється XPath запит;
- мати зручну деревовидну систему переходів від одного вузла до іншого;
- не використовує ручну реєстрацію користувацьких атрибутів;
- мати функції редагування вмісту HTML

сторінки, що дозволяє видалити, редагувати та створити нові елементи HTML сторінки;

- конвертує HTML документ в XML.

Для вибірки об'єктів з DOM моделі. продумана реалізація декількох способів, як це зробити, а саме:

- Linq to XML;
- XPath;
- XSLT.

LINQ to DOM − технологія яка дозволяє відділити сутнісну об'єктну модель даних від фізичних даних, в даному випадку HTML файлу, вводячи логічне відображення між ними.

Language Integrated Query (LINQ) – це мова інтегрованих запитів, що допомагає отримати дані з джерела даних. Зазвичай, запити виражалися на спеціальній мові запитів. Для реляційних баз даних потрібно було використовувати SQL та XQuery для XML. Таким чином, розробники змушені вивчати нову мову запитів для кожного із джерел даних. LINQ спростив ситуацію, запропонувавши узгоджену модель, для роботи з даними в різних видах джерел даних таких як: XML, SQL, ADO .Net, та різноманітних колекціях даних.

XSLT (eXtensible Stylesheet Languag e Transformations) – мова конвертації XML документів. Перетворення виразів через XSLT, описує правила перетворення вихідного документу в кінцеве дерево. Перетворення будується шляхом зіставлення зразків і шаблонів. Зразок порівнюється з елементами вихідного дерева, а шаблон використовується для створення частин кінцевого дерева. Кінцеве дерево відокремлено від вихідного дерева. Структура кінцевого дерева може повністю відрізнятися від структури вихідного дерева. У ході побудови кінцевого дерева елементи вихідного дерева можуть піддаватися фільтрації, також може бути додана нова структура.

Надважливим компонентом даної бібліотеки, є конвертація HTML документу в XML, та побудова DOM моделі. При побудові цієї моделі, для кожного із елементів генерується XPath вираз. Для прикладу візьмемо деяку HTML сторінку:

```
<html>
    <body>
        <div>Some content1</div>
        <div>Some content2</div>
    </body>
</html>
```

Для кожного з елементів, які несуть назву тегу div, буде створений XPath запит, який буде мати наступний вигляд:
html[0]/body[0]/div[0] та html[0]/body[0]/ div[1].

4. Підходи до екстрагування даних з веб-русурсів

Web Mining [3] − це процес отримання даних веб-ресурсів, який, як правило, має більше практичну складову ніж теоретичну. Основна мета Web Mining − це збір даних (аналіз) з наступним збереженням у потрібному форматі. Фактично, завдання зводиться до написання HTML аналізаторів. Є кілька підходів до вилучення даних:

- аналіз DOM дерева;
- використання XPath;
- візуальний підхід.

4.1. Аналіз DOM дерева. Використання XPath. Цей підхід ґрунтується на аналізі DOM дерева. Використовуючи цей підхід, дані можна отримати безпосеред-

ньо за ідентифікатором імені або інших атрибутів елемента дерева (таким елементом може служити параграф, таблиця, блок і т. д.). Крім того, якщо елемент не позначений яким-небудь ідентифікатором, то до нього можна дістатися за унікальним шляхом, спускаючись вниз по DOM дереву, наприклад:
body → p [10] → a [1] → текст посилання
або пройтися по колекції однотипних елементів:
body → links → 5 елемент → текст посилання.
Переваги цього підходу:

- можна отримати дані будь-якого типу і будь-якого рівня складності,
- знаючи розташування елемента, можна отримати його значення, прописавши шлях до нього.

Недоліки такого підходу:

- різні HTML / Javascript механізми та сервіси по-різному генерують DOM дерево, тому потрібно прив'язуватися до конкретного механізму;
- шлях елемента може змінитися, тому, як правило, такі аналізатори розраховані на короткочасний період збору даних;
- DOM-шлях може бути складний і не завжди однозначний.

Наступним еволюційним етапом аналізу DOM дерева є використання XPath – тобто шляхів, які широко використовуються при аналізі XML документів. Суть даного підходу в тому, щоб за допомогою деякого простого синтаксису описувати шлях до елемента без необхідності поступового руху вниз по DOM дереву.

4.2. Візуальний підхід. Сенс у даній концепції наступний - розглядати вебсторінку не як єдине ціле, а як набір інформаційних блоків, які будемо вважати одиницею, що відображається. Такий підхід дозволяє виділяти потрібні ділянки контенту і аналізувати їх у контексті всієї сторінки.

Основна ідея такого підходу це поділ сторінки на інформаційні блоки. Не зовсім тривіальне завдання, як може здатися на перший погляд. Ранні підходи були пов'язані з 1) аналізом DOM моделі; 2) аналізом великої кількості сторінок усередині сайту та визначення так званого «шаблону» сайту. Як правило, ці підходи не давали позитивного результату.

Рішенням даної проблеми – використання алгоритму VPSA (Vision-based Page Segmentation Algorithm) [4] від Microsoft Research Asia. В цьому алгоритмі використовується комбінований підхід, а саме аналіз DOM моделі і власні правила сегментації, виведені експертним або експериментальним шляхом.

Основною складністю використання даного алгоритму є процес визначення критеріїв оцінки і те, за якими правилами розрізняються різні блоки. Або більш конкретно: за якими ознаками можна відрізнити набір посилань і основний контент? Щоб це зробити потрібно виконати аналіз основних параметрів (характеристик) блоків.

Є одна реалізація даного алгоритму, вона представлена як SmartBrowser [5].

5. Проблеми аналізу контенту веб-ресурсів

Проблеми при парсингу HTML даних − використання Javascript / AJAX / асинхронних завантажень, які значно ускладнюють написання аналізаторів а саме: різні методи візуалізації HTML можуть видавати різні DOM дерева. Крім того, ці методи можуть мати помилки, які потім впливають на результати роботи аналізаторів а саме: великі обсяги даних вимагають написання розподілених аналізаторів, що тягне за собою додаткові витрати на синхронізацію.

Не можна однозначно виділити підхід, який буде 100 % застосовуватись у всіх випадках. Тому сучасні бібліотеки для аналізу HTML даних, як правило, комбінують різні підходи. Наприклад, HtmlAgilityPack [6] дозволяє аналізувати дерево DOM (використовувати XPath), а також з недавніх пір підтримується технологія LINQ to XML. Вилучення даних SDK використовує аналіз дерева DOM, містить набір додаткових методів для аналізу рядків, а також дозволяє використовувати технологію LINQ для запитів в DOM моделі сторінки.

6. Алгоритм та опис його роботи

Будь яка правильна HTML сторінка конвертується в XML-документ, за яким згодом будується DOM-модель. Будь яка DOM-модель, що побудована на основі HTML сторінки представляє собою правильне дерево, що значно полегшує аналіз.

Початковими даними для аналізу сторінки мають бути дані вилучені будь-яким способом із іншого або цього-ж самого ресурсу. Чим більше даних, які використовуються для визначення релевантності, буде мати аналізатор, тим точніше буде результат отриманий на виході.

Слід також зазначити, що деякі сервіси використовують системи захисту від автоматичного аналізу. Можна навести декілька способів захисту:

- використання AJAX для отримання даних, після завантаження HTML ссторінки;
- використання однакових атрибутів, що ідентифікують елементи(id);
- використання зображень, які несуть в собі дані.

Даний алгоритм (рис. 1) виконує функцію пошуку в будь-якій правильній DOM моделі HTML сторінки, груп вузлів які несуть в собі групи, які містять в собі релевантні елементи. Потім, для кожної з даних груп створюється свій унікальний XPath запит. Алгоритм потребує змін, у випадку якщо використовуються системи захисту від аналізу даних.

Крок 1. Аналіз та створення DOM. Першу частину роботи виконує HTML Agility Pack. Він формує DOM модель для HTML документу, та створює для кожного вузла окремий XPath запит.

Крок 2. Створення додаткових даних для кожного вузла дерева. Для кожного вузла створюється окремий ідентифікатора, який формується на основі батьківського ідентифікатора, назви тегу. Для прикладу візьмемо HTML сторінку:

```
<html>
<head>
…
</head>
    <body>
    <div class=”info”>
        <ul>
            <li>Text1</li>
            <li>Text2</li>
            <li>Text3</li>
            <li>Text4</li>
        </ul>
    </div>
    </body>
</html>
```

Елемент дерева <li> буде мати ідентифікатор “htmlbodydivclassinfoulli”. Далі формується XPath запит: html[0]/body[0] /div[0]/ul[0]/li[0], а для елемента списку з вмістом «Text2» буде сформований інший XPath html[0]/body[0] /div[0]/ul[0]/li[1]. Далі дані вузли додаються до загального списку елементів. Далі потрібно перевірити чи є в даного вузла дочірні елементи, якщо такі існують, то повторити дії описані вище(крок 1-2).

Крок 3. Формування релевантних груп. Коли всі вузли були додані до загального списку, виконується групування елементів. Для кожної групи елементів створюється окремий список. З загальним XPath який дозволяє отримати цю групу з DOM моделі даного документа.

Крок 4. Формування масиву даних для кожної із груп. У ході операцій нам доступні групи елементів які несуть у собі деякі дані. Для кожної групи створюються списки даних. Тобто, кожен елемент у групі несе в собі дані які містяться в самому тезі та його атрибутах. Наприклад, для даного елемента Html сторінки:

```
<label>
    <input type="checkbox" addition=
    ”Duna” value="412" >
    Дюны
</label>
```

будуть сформовані 3 списки даних для атрибутів для Value, addition та для вмісту тегу label. Після чого, потрібно для кожної із груп провести операцію перетину з вже існуючими даними для визначення релевантності. Якщо такі дані знайдені, автоматично формується XPath запит.

Крок 5. Оптимізація XPath запитів. Кожен обраний елемент у групі має власний XPath запит.

Причому цей запит буде відрізнятися лиш порядковим номером елемента

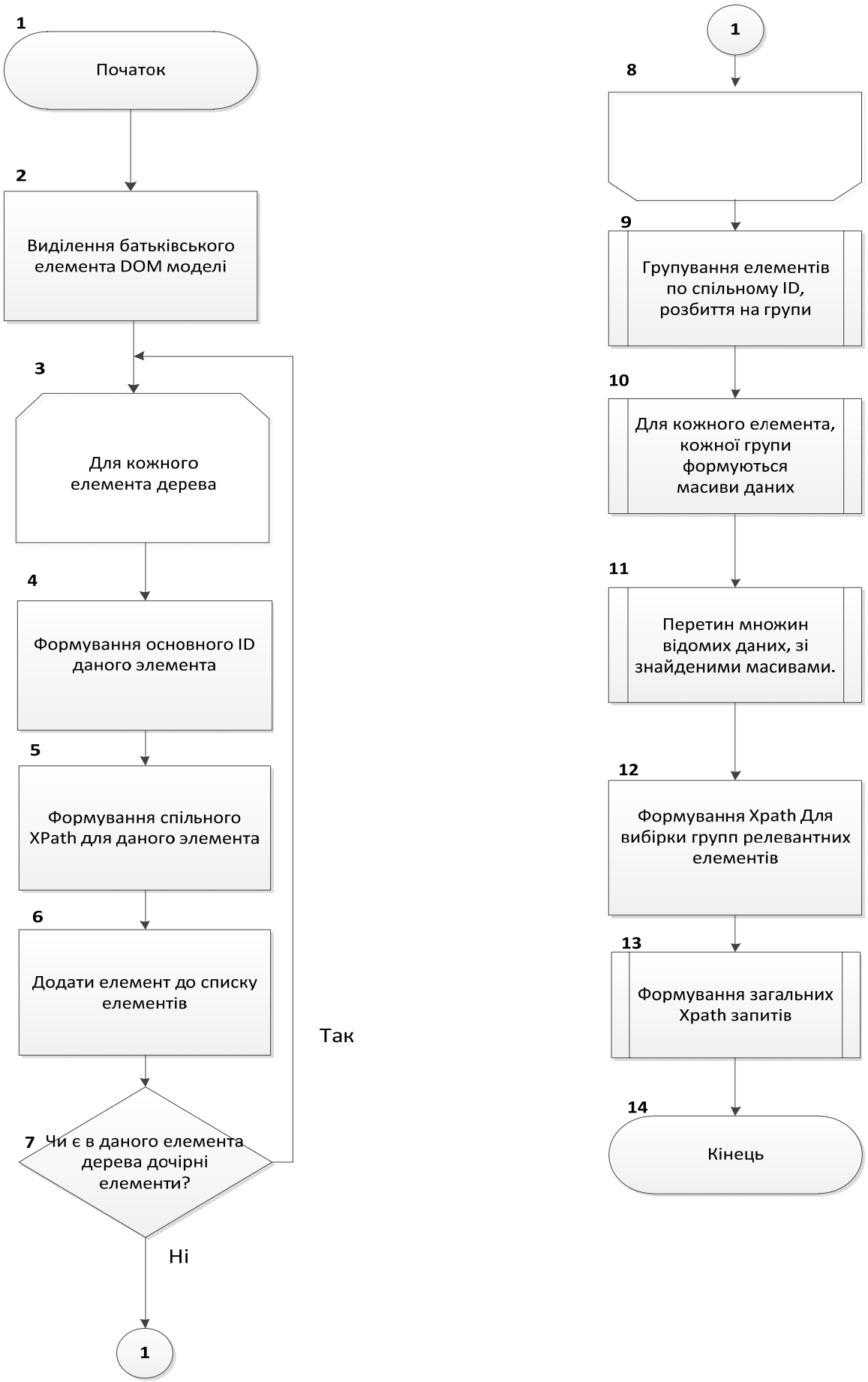

Рис. 1. Алгоритм роботи аналізатора однотипних сервісів

(li[1]) тому, щоб отримати запит для групи елементів, нам лише потрібно видалити цей номер. Тоді *X*path запит буде вибирати з DOM моделі відразу групу. Аналіз, проведений у попередньому пункті, дасть інформацію про те, який із атрибутів елемента несе в собі необхідну інформацію.

Якщо одна множина даних має дві або більше релевантних груп, то тоді потрібен перетин запитів, наприклад:

```
<html>
  <body>
  <div class="info">
  <ul>
        <li>Text1</li>
        <li>Text2</li>
  </ul>
  </div>
  <div class="info2">
  <ul>
        <li>Text1</li>
        <li>Text2</li>
        <li>Text3</li>
        <li>Text4</li>
  </ul>
  </div>
   <body>
</html>
```

Буде сформовано 2 групи XPath запитів які наведені далі:

html[0]/body[0] /div[1]/ul[0]/li,
html[0]/body[0] /div[0]/ul[0]/li.

Після чого проводиться порівняння таких XPath та виділення єдиного для вибірки обох груп html[0]/body[0] /div/ul[0]/li. Якщо такий перетин зробити неможна, оскільки XPath запити сильно відрізняються, то тоді вони зберігаються окремо.

Крок 6. Відображення результатів. У випадку виявлення конфліктів, якщо дві множини даних входять в одну множину даних отриманих з однієї групи елементів DOM моделі, такі XPath мають бути збережені, щоб користувач після повного аналізу вирішив даний конфлікт.

1. *DOM* (Document Object Model) http://www.w3.org/TR/REC-DOM-Level-1/
2. *XPath* http://www.w3.org/TR/1999/REC-xpath-19991116/
3. *Степанов Р.Г.* Технология Data Mining: Интеллектуальный анализ данных. – 2008. – http://m8.ksu.ru/EOS/dm.pdf
4. *VPSA* (Vision-based Page Segmentation Algorithm) http://research.microsoft.com/apps/pubs/default.aspx?id=70027
5. *Smart*Browser http://smartbrowser.codeplex.com/
6. *HTML* Agility Pack Project home page. – http://htmlagilitypack.codeplex.com/
7. *Smith K.A.* Cross-Disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv., 41, 1, Article 6 (December 2008). – 2008. – 25 p.
8. *Wang X., Smith K.A., Hyndman R.* Characteristic-Based clustering for time series data. Data Mining Knowl. Discov. 13. – 2006. – P. 335–364.
9. *Adelberg B.* NoDoSE*:* A tool for semiautomatically extracting structured and semistructured data from text documents, In Proceedings of ACM SIGMOD Conference on Management of Data, 1998. – P. 283– 294.
10. *Bar-Yossef Z.* and *Rajagopalan S.*, Template Detection via Data Mining and its Applications, In Proceedings of the 11th International World Wide Web Conf. (WWW2002), 2002.
11. *Richter J.* CLR via C#. – Microsoft, 2010. – 896 с.
12. *Durstenfeld R.* Algorithm 235: Random permutation. Communications of the Association for Computing Machinery, 7:420. – 1964.

Отримано 16.05.2011

Про авторів:

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділом теорії комп'ютерних обчислень Інституту програмних систем НАН України.

Молотієвський Леонід Геннадійович,
студент 5 курсу магістратури.

Місце роботи авторів:
Національний технічний університет України «КПІ», 03056, м. Київ,
проспект Перемоги, 37.
Тел.: 044 4068610,
e-mail: dor@isofts.kiev.ua,
kpi.leon@gmail.com