

Different XML-based Search Techniques on Web

Sabin Corneliu Buraga and Mihaela Brut

Faculty of Computer Science, "A.I.Cuza" University of Iasi, Berthelot Street, 16 Iasi, Romania
Phone: (+40) 32 201090, E-Mail: {busaco, mihaela}@infoiasi.ro, WWW: <http://www.infoiasi.ro/~busaco/>

Abstract – *The broad promotion of XML as the standard meta-language applied to define markups for Web documents encouraged multiple researches on promising XML query languages. In this paper we study different approaches on defining XML-based query languages and we propose WQGL (Web Query Graphical Language) - an extension of WQFL (Web Query Formulating Language). WQGL gives support to construct graphical and intuitive Web interfaces for assisting users to build complex queries.*

Keywords: *World-Wide Web, Query, XML, Markup, Data Representation*

I. INTRODUCTION

The Web has grown to be a necessary tool for all researchers. Not only the hot news, but many consistent or semi-consistent collections of scientific data, in specific disciplines, are now available on Internet. The huge quantity of accessible data involves hundreds or even thousand pages to be returned, for a given problem/subject, by a traditional search engine, with its keywords based technique. Anybody requests assistance to select, among these, the most useful and consistent documents.

Search services usually can be distinguished according to how they gather and organize their meta-information: *automatic acquisition and indexing* (achieved by Web robots) and *manual acquisition and categorization* (accomplished by qualified information experts). Currently, each method is not sufficient enough.

The evolution of the World Wide Web (especially in the field of representation of data by means of XML-based markup languages) has led to many trials of defining appropriate languages for extracting information from XML content, like *XML-QL*, *XQuery*, *XQL*, or *XML-GL* [8, 9, 13]. The paper presents such of approaches and proposes *WQGL (Web Query Graphical Language)* – an XML-based language which suggests support to visually specify complex queries, using Web graphical user interfaces, to allow outlining the structure of the desired Web page. The WQGL language was regarded as an instrument to be used to build Web interfaces for ordinary users, to make

easy and stimulate them to navigate on Web in a clever mode. Augmented usage of online search by non-specialists has increased the requirements for a more useful and friendlier search skill. Also, WQGL provides support for Perl regular expressions in order to formulate complex queries.

II. RELATED WORK

Currently, there are some proposals of techniques to locate, index, query and restructure WWW sites' content. One prior significant direction was that to combine the keyword-based *searching* with database-style support for *querying* the Web [10]. Thus have been proposed Web3QL and WebSQL languages, modeled after SQL (Structured Query Language), or WebLog, inspired by the DataLog language.

According as XML [3, 4, 16] has been imposed as the standard for semi-structured documents on the Web, the research efforts have been concerted in defining markup languages – with suitable syntax and semantics – for querying XML documents, yet following the way that conventional query languages (notably SQL) have been used for extracting information from structured data, e.g. relational databases.

The query languages for XML documents are based on the following idea: they offer to the user the possibility to formulate a query in a specific manner and generate a new XML document as a pattern for this query. It shall be compared with the target XML document and shall be retained and returned only matched data, under the restructured form of a new XML document. The query languages themselves are implemented as XML documents.

XML-QL

XML-QL [9] is a query language which allows the extraction of information from XML documents by means of an implemented *WHERE-CONSTRUCT* command, analogous to the *SELECT-WHERE* construct from SQL or other query languages for semi-structured data. The "*WHERE*" part of the command defines the interrogation

and the "*CONSTRUCT*" part specifies the modality of taking over the result.

XQL

The XML Query Language (XQL) [13] offers the possibility of filtering and extracting information from XML documents using a pattern modeled after the directory notation. The relation between tags is referred as that between (parent) directories and their (child) sub-directories. Inspired by XPath [16], the XQL accepts the "/" character to be used to specify the sub-tags and "@" – for attributes. Complex queries could be constituted by using different Boolean and relational operators.

XML-GL

The XML-GL [8] is a graph-based query language with both its syntax and semantics defined in terms of graph structures and operations. Although the queries are formulated visually, the mechanism is too sophisticated for an ordinary user. An extension of XML-GL, named *XML-GLrec*, was developed. This approach allows moreover representing XML simple links and generic recursive queries. The language formalism is more complex and, actually, forbidden to non-specialists.

Another interesting issue is given by querying the Web. Instead of classical interrogation of Web search engine, more complex and effective methods are developed.

SearchPad

SearchPad [2] is a Web tool which loads instantly (in parallel) with four important search engines (AltaVista, Excite, Google, and Hotbot) and explicitly maintain users search context. As a result, each link returned for a query by the search engine is accompanied by a "*Mark*" button. SearchPad organize automatically all the marked links into folders associated with users and queries, which are kept over the different sessions.

SearchPad has also a personal window which offer the possibility to explore (reorganize, rename, delete and even re-visit) the stored queries and links. Besides bookmarks, SearchPad maintains relationships between queries and links which the user would like to keep track of, as well as the time was spent for viewing the result of each query.

WebVCR

WebVCR [1] is a system conceived to create and replay smart bookmarks – shortcuts to Web content that require multiple browsing steps to be retrieved (login actions, filling some forms, clicking different buttons etc.). The navigation could be subsequently reiterated in the simple manner of record-playing specific to a CD-player.

Having a client-based implementation, WebVCR must be loaded in a browser window. Automatically shall be opened in a new window a HTML page containing the "*Record*", "*Play*" and "*Stop*" buttons and the WebVCR applet in the background. During the recording process, the applet memorizes the URL (Uniform Resource Locator) [4, 16] of each page reached and inserts event handlers on all active (clickable and changeable) objects in the page. Thus, each action of the user triggers the corresponding event handler that causes the applet to record it.

III. WEB QUERY GRAPHICAL LANGUAGE (WQGL)

Preliminaries

To assist users to prepare queries, we are designed an XML-based markup language, called Web Query Formulating Language (WQFL) [6, 7]. The main goal of WQFL is to permit obtaining Web matched-pages with complex and flexible queries. Each query shall be modeled by WQFL and a WQFL document will be created for each found page.

An example of WQFL query can be:

```
"complex controller" + with <4 paragraphs on top
+ with <2 images on bottom + without links
+ without multimedia content + without tables
```

Structural search activity of WQFL follows three phases [6, 7, 11]:

1. The keywords of the query expression (e.g. "complex controller") are given to a traditional search engine (e.g. AltaVista or Google) to perform a classical search activity. The search engines will return first *N* significant Web pages and the remaining expressions (i.e. *with <4 paragraphs on top*) will be parsed locally in the activity of semi-structural searching by a processing software tool;
2. The second phase's goal is to encode the Web pages structural information. According to the given potential of WQFL language, some users would like the graphical content to be found on top of the Web pages and to consist of maximum 4 paragraphs etc. That information is stored into WQFL documents. Each found Web document will be processed and it will be retained only the position (top, middle, and bottom) and the occurrences of some HTML (or XML) elements and attributes. For each element, we will retain three values that represent the occurrences of that element on top, middle and bottom of the Web page. For the entire user's query, a WQFL document (which will be locally stored) is generated. For each of the *N* found Web pages, a WQFL document is also created.
3. Every generated WQFL document is compared with the initial WQFL document (which models the user query). It is selected the best-matched one

using different Artificial Intelligence approaches, such as a self-organizing feature maps neural network based on the competitive learning [11].

Instead of HTML element names, the WQFL documents can include position occurrences information of any XML tags (such as SMIL, XHTML or MathML elements) [4, 16] and WQFL can be viewed as a query language for XML data.

Extending WQFL

Based on WQFL, the WQGL language can support new XML constructs to express more easily the structural information of the desired Web documents. The WQGL language can be used to build the graphical interfaces which will assist users to formulate complex queries. The components (widgets) of the graphical interface are defined by *XUL (Extensible User-interface Language)* [4, 12] elements. The XUL language is a platform-independent language used to represent the user interface elements within the framework of the Web browsers (i.e. Netscape or Mozilla).

The WQGL documents will consist of WQFL and XUL constructs and will be used in the first phase of searching process. Using XUL information embedded into WQFL documents, a software tool will create an intuitive interface which will help out the user to graphically choose the desired structure of found pages. Also, to build graphical interfaces for mobile and handheld devices and appliances, instead of XUL constructs can be considered *Wireless Markup Language (WML)* or *Handheld Device Markup Language (HDML)* elements and attributes.

Also, WQGL language provides support for Perl-syntax regular expressions [5, 14] that can be used to formulate user queries. Instead of directly sending the query string to a Web search engine, the query can be pre-processed or post-processed if a Perl regular expression is present into that query. The post-processing activity consists of textual analysis of the content of each found page by resolving the regular expression.

Examples

An example of the generated WQFL document fragment follows:

```
<!DOCTYPE webquery PUBLIC "-//WQFL 1.0//EN">
<?xml version="1.0" ?>
<webquery
  timestamp="18.03.2002 10:33"
  location=
    "http://www.infoiasi.ro/~busaco/index.html">
  <engine url="http://www.google.com">
    Google
  </engine>
  <engine url="http://www.altavista.com">
    Altavista
  </engine>
  <query>micro-controller</query>
```

```
<structure>
  <element name="p">
    <occur top="4" />
  </element>
  <element name="img" order="2">
    <occur middle="5" bottom="3" />
  </element>
  <element name="a" appear="no">
    <occur top="0" middle="0" bottom="0" />
  </element>
  <element name="table">
    <occur top="1" />
  </element>
</structure>
</webquery>
```

The found page contains 4 paragraphs on top, 5 images on bottom, no other hyperlinks and only one table.

To specify a tabbed control for a graphical user interface written in XUL, a generated WQGL document can contain:

```
<?xml version="1.0" ?>
<webquery>
  <engine url="http://www.google.com">
    Google
  </engine>
  <query>complex controller</query>
  <interface target="Mozilla"
    language="XUL" type="text/xul">
    <window title="WQGL" style="width: 400px">
      <tabcontrol align="vertical">
        <tabbox value="Paragraphs" />
        <tabbox value="Images" />
        <tabbox value="Tables" />
      </tabcontrol>
    </window>
  </interface>
  <structure>
    <element name="img" appear="no">
      <occur top="0" middle="0" bottom="0" />
    </element>
  </structure>
</webquery>
```

To find all documents which contain "complex" followed at any distance by one or more occurrences of "controller" or "control" terms, we can use the following Perl regular expression (this expression can be partially pre-processed):

```
<webquery>
  ...
  <query regexp="yes">
    complex+.*(controller|control)+
  </query>
  ...
</webquery>
```

Document Type Definition for WQGL

To formally describe the WQGL elements and attributes we give the following *Document Type Definition (DTD)* [4] constructs:

```
<!DOCTYPE webquery [
  <!-- WQGL elements -->
  <!ELEMENT webquery (engine+, query, interface+
    structure, page*)>
  -- web query information --
  <!ELEMENT engine (#PCDATA)>
```

```

-- search engine --
<!ELEMENT query      (#PCDATA)>
-- query expression --
<!ELEMENT structure (element*)>
-- structural data --
<!ELEMENT element   (occur*)>
-- elements data --
<!ELEMENT occur     EMPTY>
-- position occurrences --
<!ELEMENT page      (#PCDATA)>
-- found page(s) information
-- (metadata and content) --
<!ELEMENT interface ANY>
-- will include XUL or
-- other language's elements --

<!-- WQFL attributes -->
<!ATTLIST webquery
    timestamp CDATA #IMPLIED
    -- query timestamp --
    maxpages  NUMBER #IMPLIED
    -- maximum number of found pages --
    language  CDATA #IMPLIED
    -- Web pages language(s) --
>
<!ATTLIST engine
    url      CDATA #REQUIRED
    -- search engine URL --
    info     CDATA #IMPLIED
    -- additional information
    -- (e.g. use a specific language) --
>
<!ATTLIST query
    regexp   ("yes"|"no") "no"
    -- query is using Perl
    -- regular expressions --
>
<!ATTLIST element
    name     CDATA #REQUIRED
    -- stored element name (e.g. <a>) --
    order    NUMBER #IMPLIED
    -- order of significance --
    appear   yes|no yes
    -- the element must appear
    -- (position don't matters) --
    context  CDATA #IMPLIED
    -- context of element occurrence
    -- (e.g. parent tag) --
>
<!ATTLIST occur
    -- position occurrences --
    top      NUMBER #IMPLIED
    middle   NUMBER #IMPLIED
    bottom   NUMBER #IMPLIED
>
<!ATTLIST page
    url      CDATA #REQUIRED
>
<!ATTLIST interface
    target   CDATA #IMPLIED
    -- target platform (browser) --
    language CDATA #IMPLIED
    -- used language (e.g. XUL) --
    type     CDATA #IMPLIED
    -- MIME type of used language
    -- (e.g. text/xul)
>
]>

```

IV. DESIGN AND IMPLEMENTATION PROPOSAL

The application will consist of several modules used for:

- Building the graphical Web user interface which will help user to easily prepare complex graphical queries. Also, this module will create the textual query and the preferred structure of Web documents. The software module will process WQGL documents and can be placed on client side, server side or both;
- Sending to a Web engine the keywords of the user query, receiving the list of the URL addresses of the found Web pages and generating the corresponding WQFL documents. The produced WQFL documents can be locally or remotely stored (on another machine which can be considered as a Web proxy used to optimize further user queries). Sending and receiving data activities will be conformed to the HTTP (HyperText Transfer Protocol) protocol requirements [4, 16].
- Performing the structural search to obtain the best solution by comparing the structural information of the found pages with the structure defined by the user. This module will parse the WQFL documents and will return to the user the URI(s) of best solution(s). This module will include a Perl regular expression processor (engine) and can be located on client side or server side, too.

To validate the document an XML processor it can be used (e.g. *Expat*, *MSXML*, or *JAXP*). To easily process the structure of the WQFL and WQGL documents it can be used *Libxml* – a SAX (Simple API for XML) [4, 15] library available on Linux systems. For building the user interface it can be considered the existing XUL processing engine of Mozilla/Netscape 6 browser, a cross-platform module using *XPCOM* (*Cross-Platform Component Object Model*) open-source technology.

The Perl regular expression engine can be easily written in Perl [14]. Another solution is to develop the engine in Java, C++ or even JavaScript languages.

REFERENCES

- [1] V. Anupam et al., "Automating Web Navigation with the WebVCR", Proceedings of the Eight International World Wide Web Conference WWW9, ACM Press, 2000.
- [2] K. Bharat, "SearchPad: Explicit Capture of Search Context to Support Web Search", Proceedings of the Eight International World Wide Web Conference WWW9, ACM Press, 2000.
- [3] T. Bray et al. (eds.), Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, Boston, 2000: <http://www.w3.org/TR/REC-xml>
- [4] S. C. Buraga, Web Technologies (in Romanian), Matrix Rom Publishing House, Bucharest, 2001.
- [5] S. C. Buraga et al., Web Programming in bash and Perl. Polirom Publishing House, Iasi, 2002.
- [6] S. C. Buraga, T. Rusu, "An XML-based Query Language Used in Structural Search Activity on Web", in The Fourth Conference on Technical Informatics – CONTI 2000 Proceedings, Timisoara, 2000.
- [7] S. C. Buraga, T. Rusu, "Search Semi-Structured Data on Web", in The Seventh International Symposium on Automatic Control and

- Computer Science – SACCS 2001 CD-ROM Proceedings, Iasi, 2001
- [8] S. Ceri et al., "XML-GL: A Graphical Language for Querying and Restructuring XML Documents", Proceedings of the Eight International World Wide Web Conference WWW8, Toronto, 1999.
- [9] A. Deutsch et al., "XML-QL: A Query Language for XML", in Proceedings of QL'98 – The Query Languages Workshop, Cambridge, Mass., 1998: <http://www.w3.org/TR/1998/NOTE-xml-ql-19980819/>
- [10] D. Florescu, A. Levy, A. Mendelzon, "Database Techniques for the WWW: a Survey", SIGMOD Record, ACM, 1998.
- [11] O. Gogan, S. C. Buraga, "The Use of Neural Networks for Structural Search on Web", in The 10th Edition of The International Symposium on System Theory – SINTES10 Proceedings, Craiova, 2000.
- [12] I. Oeschger, XUL Programmer's Reference Manual (Third Edition), Mozilla.Org, 2000: <http://www.mozilla.org/xpfe/Xulref.zip>
- [13] J. Robie, J. Lapp, D. Schach, "XML Query Language (XQL)", in Proceedings of QL'98 – The Query Languages Workshop, Cambridge, Mass., 1998.
- [14] L. Wall et al., Programming Perl (Third Edition), O'Reilly and Associates, Cambridge, 2000.
- [15] * * *, Libxml: <ftp://ftp.gnome.org/pub/GNOME/sources/libxml/>
- [16] * * *, World Wide Consortium's Technical Reports, Boston, 2002: <http://www.w3.org/TR/>