

A Proposal for a Web Structural Search Language Based on XML Technologies

Sabin Corneliu Buraga
Mihaela Brut

Faculty of Computer Science, "Al.I. Cuza" University of Iași
G-ral. Berthelot Str., 16, Iași, Romania
{busaco, mihaela}@infoiasi.ro

Abstract

The wide promotion of XML as the standard meta-language used to define markups for Web documents stimulated multiple researches on possible XML query languages. In this paper we propose *WQGL* (*Web Query Graphical Language*), an extension of WQFL (*Web Query Formulating Language*) presented in [4, 5], which was defined as a textual-based query language for extracting information from XML documents and for restructuring such information into novel XML documents. WQFL allows to formulate various textual queries to search hypermedia information. By combining WQFL and *XUL* (*Extensible User-interface Language*) facilities, WQGL offers support to build graphical and intuitive Web interfaces for assisting users to build queries, no matter of their computer skills. Also, to allow a more flexible way to formulate complex queries, WQGL offers support for Perl regular expressions.

1 Introduction and Motivations

The Web has become an indispensable tool for all specialities' researchers. Not only the hot news, but many consistent or semi-consistent collections of scientific data, in specific disciplines (e.g. Computer Science [26], Mathematics [30] etc.) are now accessible on Internet. The huge amount of available data implies hundreds or even thousand pages to be returned, for a given problem/subject, by a traditional search engine, with its keywords based method. Anybody needs help to select, among these, the most useful and consistent documents.

Search services generally can be distinguished according to how they accumulate and organize their meta-information: *automatic acquisition and*

indexing (performed by Web robots) and *manual acquisition and categorization* (accomplished by trained information experts). Each method is not enough presently. The first is prone to errors and implies high costs, natural language dependencies and difficulties in refining the results. The second method offers a better classification, but is very slow and requires qualified human resources.

According to the recommendations of the World Wide Web Consortium [32], the *Extensible Markup Language (XML)* [2, 3] is now spreading out as the standard meta-language to define markups for content publishing on the Web. As a result, the scientific data collections on Web need to comply with this standard for interoperability and, also, have to conform to a common semantics (e.g. each item of particular data needs a precise formal definition). This evolution have led to some accomplishments in defining appropriate languages for extracting information from XML content, *XML-QL* [10], *XQuery* [9], *XQL* [15, 23], or *XML-GL* [6].

The proposed *WQGL (Web Query Graphical Language)* offers support to visually specify complex queries, using Web graphical user interfaces, which allows to outline the structure of the desired Web page. The WQGL language was conceived as an instrument to build Web interfaces for the ordinary users, to facilitate and stimulate them to navigate on Web in a wise mode. The intensive usage of online search by non-specialists has increased the need for a more effective and friendlier search experience.

Also, WQGL extends WQFL by providing support for Perl regular expressions [24] in order to formulate complex textual queries.

2 Related Work

Presently, there are known different proposals of techniques to locate, index, query and restructure WWW sites' content (see also [6, 13]). One earlier important direction was to combine the keyword-based *searching* with database-style support for *querying* the Web. Thus have been proposed *Web3QL* [17] and *WebSQL* [20] languages modeled after standard SQL or *WebLog* [18] inspired by the DataLog language.

As XML have been imposed as the standard for semi-structured documents on the Web, the research efforts have been concentrated in defining languages – with appropriate syntax and semantics – for querying XML documents, yet following the way that traditional query languages (notably SQL) have been used for extracting information from structured data, e.g. relational databases.

2.1 XML-QL

The *XML-QL* language [10] provides a textual syntax for writing queries that construct new XML documents from target documents. This syntax is based on the use of pattern-matching expressions and variables ranging over content and tag names to extract content from target documents and embed it into the result of a query.

2.2 XQL

The *XML Query Language (XQL)* [23] is an extension to the XSL (Extensible Stylesheet Language) pattern syntax [7, 8], providing a syntax to locate nodes (elements and text) within an XML document and using a notation inspired from directory path expressions. There is another approach [15] with even the same name and proposed in the same year, but modeled after the SQL's querying style. For example, the second XQL proposal extends SQL's `select ... from ... where` construct with tag variables, path expressions and URI (Uniform Resource Identifier) [1] specifications.

2.3 XML-GL

The *XML-GL* language [6] 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* [22], was also developed, which allows moreover to represent XML simple links and generic recursive queries. The language formalism is rather complex and, actually, forbidden to non-specialists. The proposed WQGL language (described in section 4) puts a visual interface at user's disposal which lighten the query activity.

3 WQFL

3.1 Preliminaries

In the designing phase of the WQFL language, the following practical situations were observed [4]:

- compound queries employ Boolean connectors (such as *and*, *or*, *near* or *not* operators used by all actual search Web engines);
- users would prefer the access to particular Web documents that have diverse data structures (e.g. without tables, only seven pictures to be placed on bottom of the desired Web page and so on);

- the search activity could be more productive if different Artificial Intelligence approaches were used.

The goal of WQFL is to allow obtaining Web matched-pages with complex and flexible queries (that should be formulated by using different Web interfaces) such as:

```
"temporal Petri nets" + with <7 paragraphs on top
+ with <3 images on bottom + without links
+ without multimedia content + without tables
```

Each query should be modeled by WQFL and for each found page a WQFL document should be generated.

Structural search activity of WQFL consists of the following phases:

1. The keywords of the query expression (e.g. "temporal Petri nets") are given to a traditional search engine (such as AltaVista or Google) to perform a classical search activity. The search engine will return the first N significant Web pages and the remaining expressions (e.g. **with <7 paragraphs on top**) will be processed locally in the activity of semi-structural searching by a processing software tool.
2. The goal of the second phase 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 the textual content to consist of maximum 7 paragraphs, etc. That information is stored into WQFL documents. Each found Web document will be processed and only the position (top, middle, and bottom) and the occurrences of some HTML elements and attributes (e.g. `<p>`, `<table>`, ``, `<applet>` and so on) will be retained.

For example, we can use a subset \mathcal{S} of the set \mathcal{H} of all legal HTML elements (tags) defined by World Wide Web Consortium. The following HTML elements can be included in \mathcal{S} to perform the semi-structural search activity [14]:

- `<p>` (paragraph),
- `` (still image; JPEG, GIF, or PNG file image formats),
- `<object>` (multimedia – sound, movie, animation, 3D virtual worlds – or generic object, such as an ActiveX component),
- `<table>` (tabular data),
- `<a>` (anchor),
- `<script>` (script code, such as JavaScript or VBScript programs),
- `<applet>` (Java applet).

Depending on the user's needs, the set $\mathcal{S} \subset \mathcal{H}$ of the selected HTML elements can vary.

For each element, we will retain three values that represent the occurrences of that element on top, middle and bottom of the Web page. According to the Document Type Definition (DTD) for WQFL (defined in section 3.2), it is generated a WQFL document (which will be locally stored) for the entire user's query, that shall be processed further on.

For each of the N found Web pages, a WQFL document is also created.

3. In the third phase, every generated WQFL document $w \in \mathcal{WQFL}$ (see section 3.2) 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 competitive learning [14].

3.2 Document Type Definition for WQFL

This subsection formally describes the WQFL elements and attributes by using the DTD formal rules.

The following DTD defines the constituents of the WQFL language:

```
<!DOCTYPE webquery [
<!-- WQFL elements -->
<!ELEMENT webquery (engine+, query, 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) --

<!-- 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 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
>
]>

```

The process of building WQFL documents can be viewed as a function $g : Pages \rightarrow WQFL$, where *Pages* is the set of found Web pages (resources) and *WQFL* is the set of corresponding WQFL documents.

The `<webquery>` document root element of a WQFL document will include at least one `<engine>` element, a `<query>` element, a `<structure>` element, and a zero or more `<page>` elements.

The `<query>` element will store the effective query expression to be send to a search Web engine and the `<structure>` element will contain the semi-structural information of the found page. Some attributes (e.g. `name`

or `url`) are mandatory and others can optionally appear (such as `maxpages`, `language` or `context`). The order of significance of an element is given by the value of `order` attribute of `<element>` tag.

Instead of DTD, it can be used an XML Schema [3, 11] to formally declare the grammar which defines the class of WQFL documents.

3.3 Examples

Let be the following user query:

```
"temporal Petri nets"  
+ with <7 paragraphs on top  
+ with <3 images on bottom  
+ without links
```

The corresponding WQFL document which model this query is:

```
<!DOCTYPE webquery PUBLIC "-//WQFL 1.0//EN">  
<?xml version="1.0" ?>  
<webquery>  
  <engine url="http://www.google.com">  
    Google  
  </engine>  
  <engine url="http://www.altavista.com">  
    Altavista  
  </engine>  
  <query>temporal Petri nets</query>  
  <structure>  
    <element name="p">  
      <occur top="7" />  
    </element>  
    <element name="img">  
      <occur bottom="3" />  
    </element>  
    <element name="a" appear="no">  
      <occur top="0" middle="0" bottom="0" />  
    </element>  
  </structure>  
</webquery>
```

After the textual query is submitted to a search engine, a number of found page is returned. For each page is generated a WQFL document which express the structure of that page. An example of a generated WQFL document fragment follows:

```

<!DOCTYPE webquery PUBLIC "-//WQFL 1.0//EN">
<?xml version="1.0" ?>
<webquery
  timestamp="18.10.2001 10:33"
  location="http://www.infoiasi.ro/~jucan/index.html">
  <!-- location provided by the search engine -->
  <engine url="http://www.google.com">
    Google
  </engine>
  <engine url="http://www.altavista.com">
    Altavista
  </engine>
  <query>temporal Petri nets</query>
  <structure>
    <element name="p">
      <occur top="7" />
    </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 7 paragraphs on top, 5 images on bottom, no other hyperlinks and only one table.

3.4 Benefits

The WQFL documents can store (within `<page>` element) the additional information about a certain Web page: location, size, metadata (creator, copyright, owner and so on), language etc. Also, to express relations between different locations of found Web documents, *RDF (Resource Description Framework)* [3, 19] assertions or *RSS (RDF Site Summary)* [31] statements could be used.

The WQFL language can be used as a standard format for interchange HTML and XML information between Web robots and agents.

Instead of HTML element names, the WQFL documents can include position occurrences information of any XML tags (such as SMIL, XHTML

or MathML elements) and WQFL can be viewed as a query language for XML data.

4 Web Query Graphical Language (WQGL)

Instead of textual queries, to help users to formulate graphical queries, we can extend WQFL with new XML constructs to express more easily the structural information of the desired Web documents. The new defined language is the *Web Query Graphical Language (WQGL)* language.

The WQGL language can be used to build the graphical interfaces which will assist the users to formulate complex queries. The components (widgets) of the graphical interface are defined by XUL elements.

4.1 Extensible User-interface Language (XUL)

The *Extensible User-interface Language (XUL)* [3, 21] is an XML-based language used to represent the user interface elements within the framework of the Web navigators (i.e. Netscape or Mozilla). Because XUL is platform-independent, the graphical user interfaces designed in XUL shall have the same look and behavior on different operating systems or graphical architectures (e.g. Motif, GTK).

XUL provides various types of widgets used to build complex graphical interfaces. These widgets are very similar with the current approaches used in graphical user interface development environments such as Borland Delphi, Glade, KDevelop or Microsoft Visual Studio .NET.

An XUL document consists of two parts. The first part includes the XML processing instructions which specify the styles that shall be used for displaying the widgets and, also, the namespaces for XUL elements. The second part defines the proper structure of the interface. It consists of an `<window>` (root) element of the XUL document, which contains the definitions of all widgets that constitute the desired interface.

Example

An example of an XUL document follows:

```
<?xml version="1.0" ?>
<?xml-stylesheet href="chrome://global/skin/global.css"
                 type="text/css" ?>

<!-- main window -->
<window title="Color Picker"
        xmlns:html="http://www.w3.org/1999/xhtml"
```

```

xmlns="http://www.mozilla.org/keymaster/
      gatekeeper/there.is.only.xul"
align="vertical"
style="width:200px">

<!-- JavaScript code for displaying color code -->
<html:script>
  function setColors(cp) {

    var color = cp.color;
    document.getElementById("input").value = color;
    document.getElementById("mydiv").style.backgroundColor
      = color;
  }
</html:script>

<colorpicker id="cp" onclick="setColors(this);"
             style="background-color: #CCCCCC;
                 text-align: center; margin: 1em"/>
<html:p />
<!-- selected color code -->
<html:input id="input" style="font-size: 12pt"
           readonly="readonly" />
<html:p />
<!-- selected color -->
<html:div id="mydiv"
         style="width: 100px; height: 100px;
             text-align: center;
             background-color: white" />

<html:hr />
</window>

```

This XUL document will build an interface used to easily select a desired color without knowledge about color numerical code.

4.2 Using XUL to build WQGL documents

The proposed idea is to extend the functionality of WQFL documents by including within the new `<interface>` element different XUL constructs. The `<interface>` element will contain all XUL elements which can be used to build the Web graphical user interface. The WQGL documents will consists of WQFL and XUL constructs and will be used in the first phase of the searching process. Using information included into `<interface>` element, a

software tool will generate an intuitive interface which will assist the user to graphically choose the desired structure of found pages.

At the DTD of WQFL documents (see section 3.2) we must add these constructs:

```
<!ELEMENT interface ANY>
  -- will include XUL or other language's elements --
<!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)
>
```

We'll change the `<webquery>` element's formal definition into:

```
<!ELEMENT webquery
  (engine+, query, interface+, structure, page*)>
```

We give the possibility to specify multiple interfaces constructs, eventually using different user interface languages (instead of XUL the XML and XSL/CSS approaches can be considered).

Also, to build graphical interfaces for mobile and handheld devices and appliances, the WQGL `<interface>` element can contain *HDML* (*Handheld Device Markup Language*) or *WML* (*Wireless Markup Language*) constructs.

Example

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

```
<?xml version="1.0" ?>
<webquery>
  <engine url="http://www.google.com">
    Google
  </engine>
  <query>temporal Petri nets</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>
</webquery>
```

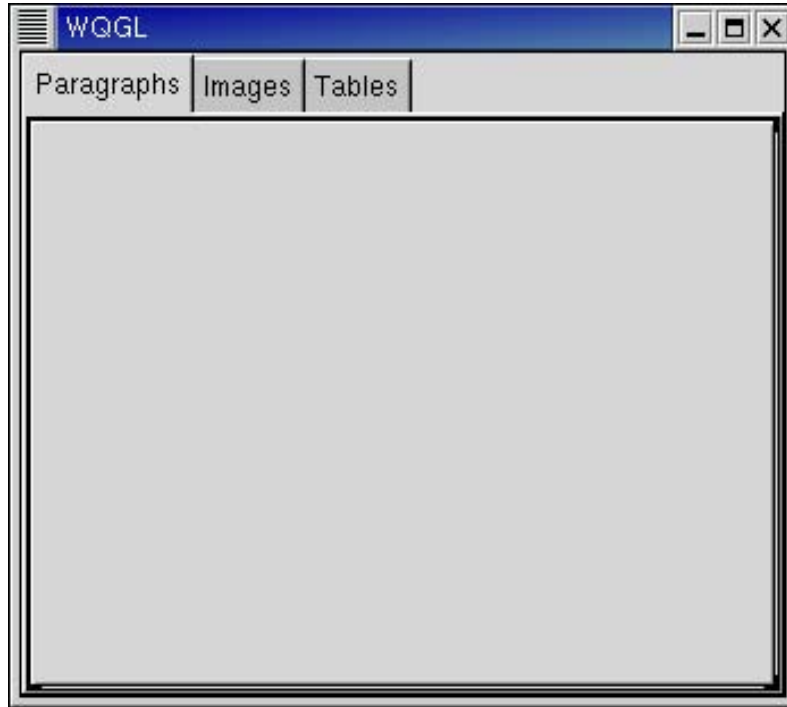


Figure 1: An XUL interface

```

    </tabcontrol>
  </window>
</interface>
<structure>
  <element name="img" appear="no">
    <occur top="0" middle="0" bottom="0" />
  </element>
</structure>
</webquery>

```

The XUL interface is illustrated in figure 1.

4.3 Extending WQFL to support Perl regular expressions

Definition 1 Let be $\Sigma = a_1, a_2, \dots, a_n$. We define the set of regular expressions [16] as follows:

1. ϕ, λ and a are regular expressions, for each $a \in \Sigma$,
2. if E_1 and E_2 are regular expressions, then $(E_1 + E_2)$ and $(E_1 \cdot E_2)$ are regular expressions,
3. if E is a regular expression, then E^* is a regular expression,

4. all regular expressions can be generated by applying the 1, 2 and 3 rules only.

Definition 2 *The attached language of a regular expression is:*

1. $L(\phi) = \phi$, $L(\lambda) = \lambda$, $L(a) = \{a\}, \forall a \in \Sigma$,
2. $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$,
3. $L((E_1 \cdot E_2)) = L(E_1) \cdot (E_2)$,
4. $L(E^*) = (L(E))^*$.

Regular expressions [16] can describe \mathcal{L}_3 -type grammars. A regular expression is equivalent to a finite (deterministic, non-deterministic, or transitional system) automaton.

A Perl *regular expression* [24] is a pattern – a template – to be matched against a string of characters; it is used by many programs, such as the UNIX commands (i.e., `grep`, `sed`, or `emacs`) or the various shells (e.g. `bash`). Each program has a different set of template meta-characters. The Perl language offers a superset of all of these tools and any complex regular expression can be written in Perl by using the same set of meta-characters. To permit at the implementation phase pattern-matching and different substitutions on a textual Web query, the WQGL extends WQFL by providing support for Perl-syntax regular expressions.

We'll add to the `<query>` element a new `regexp` attribute to indicate if the user query contains Perl regular expressions. The WQFL's DTD presented in section 3.2 is extended with the following definition:

```
<!ATTLIST query
    regexp    ("yes"|"no")    "no"
    -- query is using Perl regular expressions --
>
```

Instead of directly sending the query string to a Web search engine indicated by a `<engine>` element, the query can be pre-processed or post-processed if the `regexp` attribute has value “yes”. The post-processing activity will consist of the textual analysis of the each found page's content by verifying the matching with the regular expression.

Examples

To find all documents which contain “Petri” followed at any distance by one or more occurrences of “nets” or “network” terms, we can use the following Perl regular expression (this expression can be partially pre-processed):

```

<webquery>
  ...
  <query regexp="yes">
    /Petri*.*|(net|(s|work))+/i
  </query>
  ...
</webquery>

```

To find an expression only within a word, we can use the \B operator (provided by Perl language).

```

<webquery>
  ...
  <query regexp="yes">
    /\BTeX\b/
  </query>
  ...
</webquery>

```

The presented expression needs to be post-processed.

5 Design and Implementation Proposal

The application will consist of several modules used for:

- building the graphical Web user interface which will assist user to easily formulate complex graphical queries. Also, this module will generate the textual query and the desired structure of Web documents. The software module will process WQGL documents and can be located on client side, server side or both;
- sending to a Web engine the keywords of the user query, receiving the list of the addresses of the found Web pages and generating the corresponding WQFL documents. The generated WQFL documents can be locally or remotely stored (on another machine which can be viewed as a Web proxy to further optimize user queries). Sending and receiving data activities will be conformed to the HTTP (HyperText Transfer Protocol) protocol specifications [12].
- 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 process 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) [24] 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* [28] – a *SAX (Simple API for XML)* library available on Linux systems. Another parsing and further processing solution is given by *JDOM (Java Document Object Model)* library [27], a Java open-source implementation of World-Wide Web Consortium’s DOM [3, 25] – level 1 and level 2 recommendations.

To design the graphical user interface we intend to adopt an user-centered interface design approach. A solution is given by *LUCID (Logical User Centered Interaction Design)* [29].

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)* [3, 21] open-source technology.

A proposal for such interface is the following:

- the interface should be intuitive enough to be handled by the users without experience;
- the interface should provide a set of toolbars which will contain different widgets to easily express the graphical constituents of the queries (such as target elements, position and number of occurrences of these elements, etc.);
- the interface should provide, also, a set of layouts (like those from presentation software, e.g. Power Point) to illustrate different types of Web pages structures. For example, some layouts could model the following particularities noticed at the Web pages containing scientific data collections:
 - at the beginning of the document there is a list of internal links to the proper content, which is spread entirely in the same page;
 - it is also possible the list of anchors to consist of external links; in both cases, there can be noticed hypertext navigation facilities;
 - usually, such pages do not include script codes, Java applets or multimedia content.
- the interface should be modular in order to be easily extended in the future.

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

6 Conclusions and Further Work

The proposed WQGL language is an extension of the WQFL language which can be an useful instrument for building Web query interfaces for the ordinary users. The main goal of the WQGL is to express, by using XUL constructs, a friendly interface which allows the user to easily design himself the structure of the desired Web page.

The possibilities of the WQGL language make it very useful even to the professional users of the Internet. For example, the Web designers which have in mind a particular structure for a future page could use a WQGL-based tool to find out the existent pages with a similar structure and with the same subject as content. WQGL could be extended from the particular structural search activity to a more pronounced semantic search activity. For example, if the user places on the interface's page the widgets corresponding to a title, 10 paragraphs and 3 tabels, it could means that he needs the Web pages which effectively contain the specified keywords in the title and inside of at least 10 paragraphs and 3 tabels. In this case, the position of the consistent information (on top, in the middle or at the bottom of the page) doesn't matter, but the generated WQGL documents must taken into account only the tags which mark up a text containing the specified keywords.

As the case of WQFL, the WQGL documents could be also used as a standard format for interchange HTML and XML annotated information between different Web applications.

References

- [1] T. Berners-Lee *et al.* (eds.), *Uniform Resource Identifiers (URI): General Syntax*, Internet Standard, RFC 2396, IETF, 1998
- [2] T. Bray *et al.* (eds.), *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, Boston, 2000:
<http://www.w3.org/TR/REC-xml>
- [3] S. C. Buraga, *Web Technologies* (in Romanian), Matrix Rom Publishing House, Bucharest, 2001
- [4] 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, Timișoara, 2000
- [5] 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, Iași, 2001

- [6] 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
- [7] J. Clark (ed.), *XSL Transformations*, W3C Recommendation, Boston, 1999:
<http://www.w3.org/TR/xslt>
- [8] J. Clark, S. DeRose (eds.), *XPath*, W3C Recommendation, Boston, 1999:
<http://www.w3.org/TR/xpath>
- [9] S. DeRose, *XQuery: An Unified Syntax for Linking and Querying General XML Documents*, in Proceedings of QL'98 – The Query Languages Workshop, Cambridge, Mass., 1998
- [10] 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/>
- [11] D. Fallside (ed.), *XML Schema*, W3C Recommendation, Boston, 2001:
<http://www.w3.org/TR/xmlschema-0/>
- [12] R. Fielding *et al.* (eds.), *Hypertext Transfer Protocol – HTTP/1.1*, Internet Standard, RFC 2068, IETF, 1997:
<http://www.ietf.org/rfc/rfc2068.txt>
- [13] D. Florescu, A. Levy, A. Mendelzon, *Database Techniques for the WWW: a Survey*, SIGMOD Record, ACM, 1998
- [14] 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
- [15] H. Ishikawa, K. Kubota, Y. Kanemasa, *XQL: A Query Language for XML Data*, in Proceedings of QL'98 – The Query Languages Workshop, Cambridge, Mass., 1998
- [16] T. Jucan, *Formal Languages and Automata* (in Romanian), Matrix Rom Publishing House, Bucharest, 1999
- [17] D. Konopnicki, O. Shmueli, *W3QL: A Query System for the World Wide Web*, in Proceedings of the 21th International Conference on Very Large Databases, Zurich, 1995
- [18] L. Lakshmanan, F. Sadri, I. Subramanian, *A Declarative Language for Querying and Restructuring the Web*, in Proceedings of RIDE-NDS, IEEE Computer Society Press, 1996

- [19] O. Lassila, R. Swick (eds.), *RDF Model and Syntax Specification*, W3C Recommendation, Boston, 1999:
<http://www.w3.org/TR/REC-rdf-syntax/>
- [20] A. Mendelzon, G. Mihailă, T. Milo, *Querying the World Wide Web*, in Proceedings of the Conference on Parallel and Distributed Information Systems, Toronto, 1996
- [21] I. Oeschger, *XUL Programmer's Reference Manual* (Third Draft Edition), Mozilla.Org, 2000:
<http://www.mozilla.org/xpfe/Xulref.zip>
- [22] B. Oliboni, L. Tanca, *Querying XML Specified WWW Sites: Links and Recursion in XML-GL*, Technical Report, University of Milano, 2000
- [23] J. Robie, J. Lapp, D. Schach, *XML Query Language (XQL)*, in Proceedings of QL'98 – The Query Languages Workshop, Cambridge, Mass., 1998
- [24] L. Wall *et al.*, *Programming Perl* (Third Edition), O'Reilly and Associates, Cambridge, 2000
- [25] L. Wood (ed.), *Document Object Model (DOM) Level 1 Specification*, W3C Recommendation, Boston, 1998:
<http://www.w3.org/TR/REC-DOM-Level-1/>
- [26] * * *, *Computer Science Resources*:
<http://citeseer.nj.nec.com/cs>
- [27] * * *, *JDOM*:
<http://www.jdom.org>
- [28] * * *, *Libxml*:
<ftp://ftp.gnome.org/pub/GNOME/sources/libxml/>
- [29] * * *, *Logical User Centered Interaction Design*, Cognetics, 1999:
<http://www.cognetics.com/lucid/>
- [30] * * *, *Mathematics Resources*:
<http://mathbookshelf.fullerton.edu>
- [31] * * *, *RDF Site Summary (RSS) 1.0*:
<http://purl.org/rss/1.0/>
- [32] * * *, *World Wide Consortium's Technical Reports*, Boston, 2001:
<http://www.w3.org/TR/>