# Design and Implementation of a Dynamic Metadata Editor

**Nor Adnan Yahaya[1] , Rosiza Buang[2] and Noor Hafizah Hassan[1]**

Malaysia University of Science and Technology
Petaling Jaya, Selangor, MALAYSIA

Petronas Berhad
Kuala Lumpur, MALAYSIA

**Summary**
This paper discusses the development of a web-based editor to support the authoring and management of Dublin Core (DC) metadata for web sources, using the Resource Description Framework (RDF) as the main representation scheme. The metadata editor is dynamic in the sense that it has the capability to automatically extract relevant content of DC metadata elements from the Dublin Core Metadata Initiative (DCMI) website as well as other relevant DC compliant metadata from the web source in question. While the first capability makes the editor responsive to any new relevant updates to the DC metadata scheme, the second one helps in reducing the manual input of actual metadata during the authoring and management process. The editor was implemented mainly in C#.Net language through the use of Microsoft Visual 2003. It uses SQL Server 7 for database management.

***Keywords:***

*Metadata authoring, Dublin Core, Resource Description Framework, Semantic Web.*

## 1. Introduction

Today thousands or even millions of web pages are being made available daily as the Internet becomes a favorite platform choice for transferring and sharing information among people globally. These voluminous pages of information are composed of text, static image, audio, movie and animation which are often being represented in different types of format. This, coupled with the lack of uniformity and standardized approach to web development, have made web processing becoming more complex. Existing index-based search engines are no longer capable of meeting the increasing demand for more accurate and meaningful search for information over the increasing volumes of web pages.

The Web was initially built for human interaction rather than to support machine to machine interaction [1]. The lack of processable knowledge associated with the Web generally has inhibited intelligent access and choice to accurate information by machines or any software. To overcome this, the use of metadata has become fundamental. In describing web sources, metadata, which is simply "data about data" [1] can include the authors of the web source in question, its date of creation or updating, the organization of the website sitemap, as well as key words representing the subject matter that can help to improvise searching.

The Semantic Web Initiative has developed RDF/XML language [1][2], which is expected to be the standard framework for representing metadata for Internet resources. In a parallel development, the Dublin Core Metadata Initiative (DCMI) [3] has taken the lead in developing metadata standard known as Dublin Core (DC) for describing cross-domain information resources. These two inter-related developments on RDF and DC respectively have motivated us to research on metadata development environment that takes advantage of the potentials offered by these two. Our initial effort resulted in the development of a dynamic metadata editor called D-DC/RDF-Editor, which is the main subject of this paper.

The paper is organized as follows. In the next section we present our observations on existing DC metadata editors by highlighting the main similarities and differences among representatives of them. Section 3 presents the salient features of D-DC/RDF-Editor by combining the strengths of these existing editors. Section 4 and Section 5 discusses the design and some implementation aspects of the editor, respectively. Section 6 provides an illustration of the usage of the editor and finally, Section 7 presents our conclusions.

Manuscript received March 5, 2008
Manuscript revised March 20, 2008

## 2. Observations on Some Existing DC Metadata Editors

Dublin Core (DC) Metadata Schema was designed to serve 2 levels of complexity, namely the Simple DC and Qualified DC. Simple DC comprises of 15 elements which can describe the basic metadata of any Internet resource. Qualified DC allows authors to refine some of these basic DC elements to make their meaning more specific as well as to provide additional information that can aid in the interpretation of the values of some these elements. Further description on these two levels of metadata schema can be found in [3].

Table 1 below provides a summary of comparison among three [4] [5] [6] of the existing DC metadata editors, in terms of the main features offered.

Table 1: Features comparison among DC metadata editors

| Features | DC-Dot | Reggie | DCMT |
|---|---|---|---|
| Auto Extraction of Metadata | Yes | No | No |
| Simple DC Elements | Yes | Yes | Yes |
| Qualified DC Elements | No | Yes | Yes |
| Presentation of DC Elements | Static | Dynamic | Dynamic |
| Metadata Format | HTML<br>XHTML<br>RDF/XML<br>XML<br>IEEE LOM<br>IMS<br>USMARC<br>Abbr RDF<br>etc | HTML<br>RDF/XML<br>Abbr RDF | HTML<br>XHTML |
| Default Template | No | No | No |
| Multiple Metadata Schemas | No | Yes | No |
| Suggests Additional Metadata | No | No | No |

There are two interesting observations that can be highlighted from the above comparison :

- the editors that support both simple and qualified DC elements do not provide any feature that does automatic extraction of metadata from the available sources.
- On the other hand, the editors that support automatic extraction of metadata supports only simple DC elements.

Hence, if these two powerful features were integrated into one metadata editor, then the resultant would be a powerful and effective tool. D-DC/RDF-Editor was intended to be one such a tool.

## 3. Salient Features of D-DC/RDF -Editor

In a nutshell, D-DC/RDF-Editor basically provides a single platform for a user to automatically generate metadata that conform to the enriched features of simple and qualified DC metadata standard and that are also representable in RDF language. Its main aim is also to ensure a consistent creation of metadata through maximizing the strengths offered by the DC and RDF standards. In general, it has the capability of capturing as many relevant metadata particularly for the DC *subject* element (or well known as keywords) where users are allowed to pick and choose the suggested keywords, plus the option to add more relevant keywords to their metadata as desired. This editor also has an added advantage in supporting future new elements that are related to Simple DC, Qualified DC, Encoding Schemes or Term Vocabularies without having to modify the programming code. This is achieved through automatic detecting of any new or obsolete DC elements based on the up-to-date definitions and schemas that are publicly available in Dublin Core Metadata Initiative(DCMI) website (http://dublincore.org).

Table 2 provides a summary of features of D-DC/RDF-Editor that combine the strengths of the existing editors as depicted in the previous section.

Table 2: Features of the D-DC/RDF-Editor

| Features | D-DC/RDF-Editor |
|---|---|
| Auto Extraction of Metadata | Yes |
| Simple DC Elements | Yes |
| Qualified DC Elements | Yes |
| Presentation of DC Elements | Dynamic |
| Metadata Format | RDF/XML<br>***XHTML***<br>***XML***<br>***Abbreviated RDF*** |
| Default Template | Yes |
| Multiple Metadata Schemas | ***Yes*** |
| Suggests Additional Metadata | Yes |

Representing metadata in the form that conforms to RDF schema ensures their consistency. This is particularly important in creating more complex and dynamic metadata which can be achieved through applying Qualified DC elements and the suggested Encoding Scheme. Furthermore, this can be carried out with minimal efforts and errors, via the application of default templates in order to avoid repetitive typing of the same data and rapid deployment of metadata. Last but not least, the editor also enables the system administrator to detect and update new updates of Dublin Core Metadata (DCM) only through the touch of a few buttons.

Dynamism and flexibility are the two main design goals of D-DC/RDF-Editor. The flexible design of the editor' s database has allowed for seamlessly transition from Simple to Qualified DC and vice versa without having losing its metadata information or duplicating the data entry. Overall the database subsystem is dynamic in the sense that changes in master reference data are done automatically through data wrapping of the schemas and related namespace maintained by DCMI, in order to ensure that they are up-to-date. Accuracy of metadata is also maintained through extraction of related metadata values from the web source itself as well as the use of default template for a standard set of metadata when extracting from multiple sources. Finally, the metadata that has been created can be generated into RDF/XML format files which can be made accessible by search engines.

With the auto extraction of metadata as the key feature, the functionality of D-DC/RDF-Editor can be organized into the following three main functions:

a) **Import Metadata Elements**: The main purpose is to populate the fields of the master reference tables with DC element names and other relevant names or information that can be extracted from the schemas that are available in DCMI website. There are five (5) major classes of names that need to be imported/extracted for the construction of these master reference tables. They are those pertaining to Simple DC, Qualified DC, Encoding Scheme Class, Encoding Scheme and DCMI Type Vocabulary. Due to dynamic and progressive development of DC standard, a mechanism was put in place to automatically detect any new, changes, or obsolete DC elements. Furthermore, Manual addition of DC elements is not allowed in order to maintain the conformance to the DC standard being used.

b) **Maintain Metadata Elements**: Once the master reference tables are fully populated with metadata-relevant names, it can then be edited and saved, and deleted to suit a user's requirement. The editor also enables the user to sort the appearance of these metadata elements by having the most used elements appear first and then followed by the least used ones. Once the order is confirmed and saved, it will be automatically reflected in further usage.

c) **Create Dublin Core Metadata**: This provides the support for the creation of the actual metadata itself which allows for two levels of sophistications, i.e the Simple and Qualified DC. The system is designed to offer flexibility to a user where he/she could toggle between these two without affecting the consistency and integrity of the metadata. For simplified metadata, Simple DC is a choice while full-blown Qualified DC would be favored by those who wish to create more complex metadata.

As will be described later, in the D-DC/RDF-Editor, the *Import Metadata Elements* function is fully automated. The other two functions require total and partial manual intervention respectively.

In creating metadata, a user is required to specify the URL address of the web source in question. When metadata are being constructed for the first time, the data value for certain DC metadata elements such as the *identifier*, *date*, *type* would be automatically extracted from the *title* element, *class* element, *meta* element of the corresponding HTML web page and its associated HTTP header. The resulting extracted data would then be displayed to the user for editing. Should the user select to use the default template, default values are also inserted in separate records. The sequence of the metadata elements would be according the sorting order that has been been set through the *Maintain Metadata Elements* functional module. Once the relevant metadata records are created, they are then stored in a database for future retrieval.

D-DC/RDF-Editor also extracts additional DC compliant metadata from other elements of HTML page such as *href*, *abbr*, *acronym*, *cite*, and *class* for further selection. Once the metadata of interest have been reviewed and edited, the user could generate them in RDF / XML compliant format that can also be displayed for final review. The resulting RDF/XML file could then be validated using a third party RDF Validation Services provided by W3C. Once validated, the user will be given the options to e-mail, or to copy and paste the metadata into an XML document.

## 4. Architectural Design

D-DC/RDF-Editor is basically based on the following simple architecture (Fig. 1).

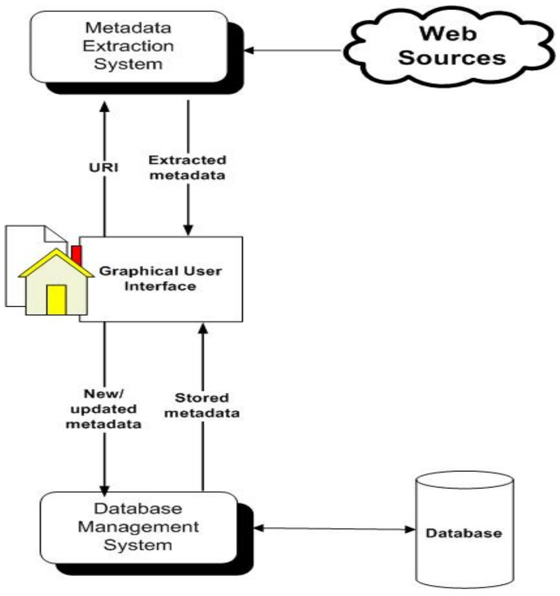


Fig. 1 Architecture of D-DC/RDF-Editor

The core component of the editor is the *Metadata Extraction System*. Generally, it has to carry out the following tasks :

- Import the relevant DC schema information that describe metadata elements found in DCMI website.
- Extract DC compliant metadata for the given web source from any of its associated RDF (*.rdf) file.
- Extract DC compliant metadata from relevant sections of the given web source.

Fig. 2 shows the design of *Metadata Extraction System.*

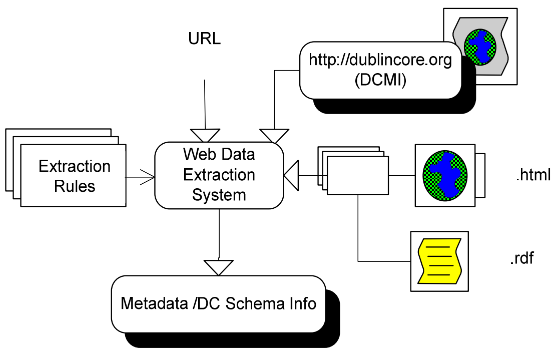


Fig. 2 Design of Metadata Extraction System

Another important component of the metadata editing system is The *Database Management System*. This can be any typical relational DBMS that can allow the constructed metadata be stored and managed as relational database records. In addtion, the same subsystem is also used to store and manage the master reference entities that represent the DC schemas.

Finally, the GUI system serves as an interface to the user in order to perform the various functions of the system and provide the windows for displaying the system's outputs in attractive manner.

## 5. Construction Model and Code Structures

Current prototype of D-DC/RDF-Editor implements the above architecture as a web-based system. The implementation structure is depicted in in Fig. 3.

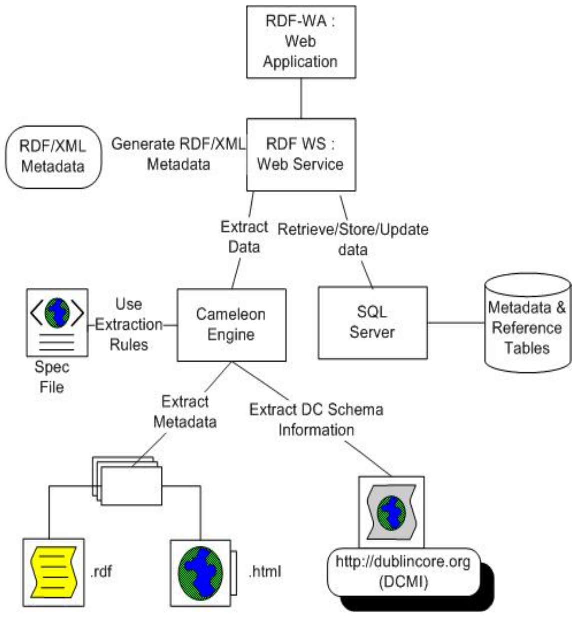


Fig. 3 Implementation structure of D-DC/RDF -Editor

### 5.1 Main Components

The main task of each component is described as follows :

a) The **RDF-WA** implements the GUI module of the system's architecture. It is implemented as an ASP.NET (using C#) web application and being the only component that has direct interface with the user. Its major function is to effect the extraction, creation and generation of metadata corresponding to a given URL address. The user is given the option to create a simple or complex (qualified) metadata. As mentioned earlier, one key advantage that D-DC/RDF-Editor has over other editors is its capability in extracting basic metadata or extras from a web source. User could enhance the metadata by manually inputting their metadata or deleting irrelevant metadata. To facilitate speedy but accurate data entry, this editor offers a default template facility for repetitive data entry. It also enables a user to update latest DC elements by wrapping latest elements from DCMI schemas.

b) The **RDF-WS** is an ASP.NET web service consisting of web methods (implemented in C#.NET) that support request for information or actions to be performed on them from the RDF-WA component. These methods could be dealing with either the SQL Server database or the web sources. For the latter case, the required data need to be extracted from the web source in question. This is done through the use of Cameleon Web Wrapper technology [7]. In general, RDF-WS is delegated to extract and store master reference data from DCMI schemas and namespaces, as well as metadata of a web source. It is also delegated to manage the these persistent data in the SQL Server database. In the current implementation, RDF-WS and RDF-WA reside within a same server.

c) The **Microsoft SQL Server** is used to manage and maintain master reference data extracted from DCMI schemas and namespaces, as well as the metadata that has been confirmed by the user through RDF-WA. The database for D-DC/RDF-Editor is centered on one entity, *Metadata* which is supported by another entity called *Default Template*. Both entities have an associated detail record and are dependent on DC Schemas and other references. Five master reference entities are identified to represent DC Schemas : *Simple DC*, *Qualified DC*, *Encoding Scheme*, *Classes of Encoding Schemes*, and *Vocabulary Term*. These entities are connected by one central table containing all its details.

d) The **Cameleon Web Wrapper Engine** is used to extract data from the given web source based on the extraction patterns given in the associated specification file(s). In this case, we use Cameleon# [8] (in .dll format) which is has been developed as Web Wrapper engine within the MIT's Context Interchange System (COIN) [9]. The Specification File (abbreviated as Spec file) is an XML file (therefore with xml extension) that contains a series of regular expressions that specify the extraction pattern of what to be extracted from a web source. In this context, the spec file corresponds to the file containing extraction rules as specified in the design for Metadata Extraction System (Fig. 2). In the current implementation, 18 Spec files are developed and used. The Spec files are categorized into two: Extraction of reference data and metadata. As mentioned earlier, reference data provides information on DC schemas that are found in DCMI website and therefore commonly performed by the System Administrator. Metadata extraction techniques being used are discussed in the next section.

There are four main types of data that are being defined under this implementation :

a) The **target web pages** are fed into the system for automatic extraction of its metadata. The extraction is limited to pages with HTML representation only.

b) The **Dublin Core Schemas** provide descriptions of DC elements. This comprise of element names and other information that make up the master reference tables. All these are represented in XML and extracted using the Cameleon engine.

c) The **RDF/XML Metadata File** contains metadata for the web source in question. This is expressed in RDF/XML to by the RDF-WA component. The metadata will either be e-mailed to a user or available for cut/copy and paste into an application.

d) The **Metadata Tables** contain the metadata for the various web sources that have undergone the metadata authoring process. These tables are managed by the SQL Server.

## 5.2 Metadata Extraction

Automated metadata extraction from web sources relies mainly on the use of the Cameleon# Web Wrapper [8] which is essentially a web data extraction tool. This tool allows selected web contents to be extracted through the creation of relational abstractions of the corresponding web sources. This act, in essence, wraps the web source in question by associating it with a Spec file(s) that contain virtual schema declarations together with rules for extracting virtual attribute values as well as some other related rules.

Analysis of HTML pages and DC Schemas has contributed to the development of generic Spec files to support automatic metadata extraction. Analysis of HTML pages reveals that HTTP Header and several HTML elements carry metadata or potential information as metadata. Analysis of DC Schemas allows for extraction of metadata element descriptions from DC schemas to build the DC master reference tables.

Every Spec file usually contains at least three elements: RELATION, SOURCE, ATTRIBUTE. The RELATION element specifies the name given to the relational abstraction of the resource pointed to by the URI given in the SOURCE element. Each ATTRIBUTE element corresponds to an attribute identified with the relational abstraction in question. The ATTRIBUTE element contains three sub-elements BEGIN, END, and PATTERN. The BEGIN and END elements each contains a regular expression (regex) describing the string that respectively denotes the beginning and the end of the block inside the target web source that contains the value for the attribute in question. The PATTERN element is actually the one that contains the regex describing the pattern for extracting the string that corresponds to the attribute value that needs to be extracted. In above example, the string to be extracted is described by the regex inside the round parenthesis pair, which is supposed to be the title of the web source in question. Multiple SOURCE elements could exist in a spec file to enable data extraction from several web sources. The URI address could either be hard-coded or be treated as a parameter (variable) where its value could be passed from a system application. A variable is an identifier enclosed within a pair of #.

Metadata extraction is done automatically for a new URL page and upon request subsequently. This extraction is performed both on the HTML page and associated RDF/XML page if available. The extracted values from basic elements such as title, meta, and class are inserted as default metadata for the web source in question. Values that are extracted from acronym, cite, abbr, and href elements are to be treated as extra metadata and generated only upon request. At this stage, even though elements such as link and img may contain potential metadata, they are not to be extracted.

In the following subsections, we provide code snippets from [10] for extracting data from RDF page, HTTP Header, and some of HTML elements.

### 5.2.1 RDF page

An RDF page may be associated with a web source as a result of certain metadata development process. Normally, the URL for this type of page contains the domain name of its parent page and with an .rdf extension. For example, the URL address for the RDF page for DCMI Website is http://dublincore.org/index.shtml.rdf. This page contains DC metadata describing the website http://dublincore.org. Metadata extraction from RDF page is performed through the use of two Spec files. The first Spec file (Fig. 4) is consulted to determine existence of the RDF page. Once the RDF page is found to exist, the second spec file (Fig. 5) is consulted to extract the metadata it contains.

```
<ATTRIBUTE name="RDFURLAdd" type="String">
   <BEGIN><![CDATA[<body]]></BEGIN>
   <END><![CDATA[</body>]]></END>
   <PATTERN>
      <![CDATA[<a href="(http://.*.rdf)"]]>
   </PATTERN>
</ATTRIBUTE>
```

Fig. 4 Specification for extracting URL address of RDF Page

```
<SOURCE URI="#RDFURLAdd#">
   <ATTRIBUTE name="DCElement" type="String">
      <BEGIN><![CDATA[<rdf:RDF]]></BEGIN>
      <END><![CDATA[</rdf:RDF>]]></END>
      <PATTERN>
           <![CDATA[<dc:([^>]*)>]]>
      </PATTERN>
   </ATTRIBUTE>

   <ATTRIBUTE name="DCMetadata" type="String">
      <BEGIN><![CDATA[<rdf:RDF]]></BEGIN>
      <END><![CDATA[</rdf:RDF>]]></END>
      <PATTERN>
         <![CDATA[<dc:[^>]*>([^<]*)<]]>
      </PATTERN>
   </ATTRIBUTE>
</SOURCE>
```

Fig. 5 Specification for extracting metadata from RDF Page

### 5.2.2 HTTP Header

HTTP header contains fixed information which include *date* and *format* elements. As illustrated in Fig. 6, we use HttpWebRequest class of C# (.Net) to extract these two elements.

```
// Initialize variables
HttpWebRequest HttpWReq =
(HttpWebRequest)WebRequest.Create(txtURL.Text);
HttpWebResponse HttpWResp = (HttpWebResponse)HttpWreq.GetResponse();
HttpWReq.KeepAlive = false;
```

```
int i = dgDCMI.Items.Count;
//for every record displayed on datagrid / or for every simple dc elements
for (int j=0; j<i; j++)
{
    //Search for Date element and Extract it from HTTP header
    if(dgDCMI.Items[j].Cells[2].Text.Trim().ToString().ToLower()
                                        ==“date”)
    {
        dgDCMI.Items[j].Cells[3].Text=
            HttpWresp.LastModified.ToString();
        dgDCMI.Items[j].Cells[4].Text = “Text”;

    }
    //Search for Format element and Extract it from HTTP header
    else if(dgDCMI.Items[j].Cells[2].Text.Trim().ToString().ToLower()
                                    ==“format”)
    {
        dgDCMI.Items[j].Cells[3].Text=
            HttpWresp.ContentType.ToString();
        dgDCMI.Items[j].Cells[4].Text = “Text”;
    }
}
```

Fig. 6 Code snippet for extracting from *date* and *format* elements

### 5.2.3 Title Element

The title element under DC Metadata standard refers to the name given to the web source in question. For HTML document, this normally appears on the <head> section of a page. Title extraction also carries dual purposes: (1) to determine the existence of the web source (2) to get a value assigned to title element. Fig. 7 below shows a snippet of the generic spec file required by the Cameleon wrapper engine for extracting the content of the title element needed.

```
<RELATION name="HTMLTitleWrap">
  <SOURCE URI="#HTMLURLAdd#">
    <ATTRIBUTE name="metaTitle" type="String">
      <BEGIN>
          <![CDATA[<[Hh][Ee][Aa][Dd]>]]>
      </BEGIN>
       <END><![CDATA[</[Hh][Ee][Aa][Dd]>]]></END>
      <PATTERN>
         <![CDATA[<[Tt][Ii][Tt][Ll][Ee]>
         ([^<]*)</[Tt][Ii][Tt][Ll][Ee]>]]>
       </PATTERN>
    </ATTRIBUTE>
  </SOURCE>
</RELATION>
```

Fig. 7 Specification for extracting from Title element

### 5.2.4 Meta Elements

Similar to RDF page, the **meta element** specification in the Header section of an HTML page also contains metadata for the page itself. Meta element does not contain a standard meta name but may provide a link to its schema the link element. To simplify the process, the extracted meta name is matched with user-defined names from DDC/RDF-Editor database. If a matched is found, it is assigned to its respective DC element. To start with, most common meta names are pre-defined in DDC/RDF-Editor database and could be continuously updated of any new repeat meta names. For example, most common meta name is “keywords”. Thus “keywords” is then stored in DDC/RDF-Editor database associated with the DC element, *subject*.

Unlike other extractions as described in title element and RDF page, the extraction of meta element is done by line where the whole meta element is extracted. The extraction is further refined at the system application where its name, value and scheme are segregated and processed as mentioned earlier. Fig. 8 shows a snippet of the Spec file for this purpose.

```
<ATTRIBUTE name="metaLine" type="String">
  <BEGIN><![CDATA[<head]]></BEGIN>
  <END><![CDATA[</head>]]></END>
  <PATTERN>
      <![CDATA[<meta([^>]*)>]]>
   </PATTERN>
  <PATTERN>
     <![CDATA[<link rel="meta" href="([\0-\377]*?)"]]>
  </PATTERN>
</ATTRIBUTE>
```

Fig. 8 Specification for extracting from Meta element

### 5.2.5 Class Elements

**Class element** is potentially related to the DC *subject* element. Typically, a class element is associated with menus or submenus of the web sources and appears within a Body section. The result of an extraction could be as few as none or exceeding 30. The DDC/RDF-Editor picks up to first 30 values and assigns them to a single subject element. The remaining values are made available upon request. Fig. 9 shows the code snippet of the spec file for extraction.

```
<SOURCE URI="#HTMLURLAdd#">
 <ATTRIBUTE name="metaClass" type="String">
  <BEGIN><![CDATA[<body]]></BEGIN>
  <END><![CDATA[</body>]]></END>
  <PATTERN>
    <![CDATA[<a class=[^>]*>\s* ([\0-\377]*?)\s*</a]]>
  </PATTERN>
  <PATTERN><![CDATA[<td class=[^>]*><a[^>]*>
   \s*([\0-\377]*?)\s*</a]]>
  </PATTERN>
 </ATTRIBUTE>
</SOURCE>
```

Fig. 9 Specification for extracting Class element

### 5.2.6 Other Elements

**Other elements** such as *cite*, *acronym*, *abbr*, and *href* are extracted upon request. Although the values of most of these elements are normally keywords for the web source, a user may opt to assign to other DC elements as an in one record where the extractions are separated with a user-defined separator, or individually in separate records. Fig. 10 illustrates the code snippet of the spec file for extracting of the acronym element.

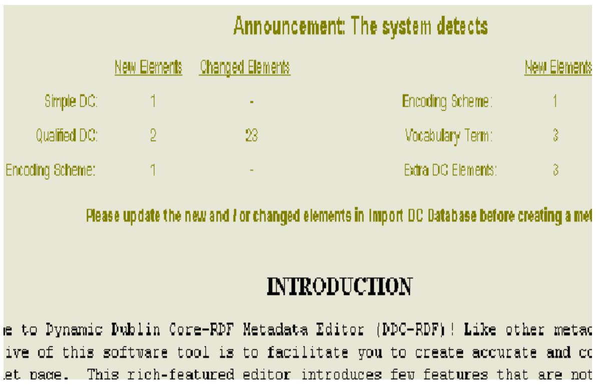
```
<ATTRIBUTE  name="bodyAcronymFull" type="String">
  <BEGIN>
    <![CDATA[<[Bb][Oo][Dd][Yy]]]>
  </BEGIN>
  <END><![CDATA[</[Bb][Oo][Dd][Yy]>]]></END>
  <PATTERN><![CDATA[<[Aa][Cc][Rr][Oo][Nn][Yy]
       [Mm][^>]*[Tt][Ii][Tt][Ll][Ee]="([^"]*)"[^>]*>]]>
   </PATTERN>
</ATTRIBUTE>

<ATTRIBUTE name="bodyAcronym" type="String">
  <BEGIN>
      <![CDATA[<[Bb][Oo][Dd][Yy]]]>
  </BEGIN>
  <END><![CDATA[</[Bb][Oo][Dd][Yy]>]]></END>
   <PATTERN><![CDATA[<[Aa][Cc][Rr][Oo][Nn]
      [Yy][Mm][^>]*[Tt][Ii][Tt][Ll][Ee]=[^>]*>
      ([^<]*)</[Aa][Cc][Rr][Oo][Nn][Yy][Mm]]]>
   </PATTERN>
 </ATTRIBUTE>
```

Fig. 10 Code Snippet of Extraction of Acronym Element

## 6. Illustration

In this section we show portions of screen samples illustrating a typical scenario in authoring metadata through D-DC/RDF-Editor.

Fig. 11 shows a portion of the main page of D-DC/RDF-Editor. The upper part of the main page contains an announcement regarding any new changes to the elements in the schemas that has been detected by the system from the Dublin Core website.



Announcement: The system detects

| | New Elements | Changed Elements | | New Elements |
|---|---|---|---|---|
| Simple DC: | 1 | - | Encoding Scheme: | 1 |
| Qualified DC: | 2 | 23 | Vocabulary Term: | 3 |
| Encoding Scheme: | 1 | - | Extra DC Elements: | 3 |

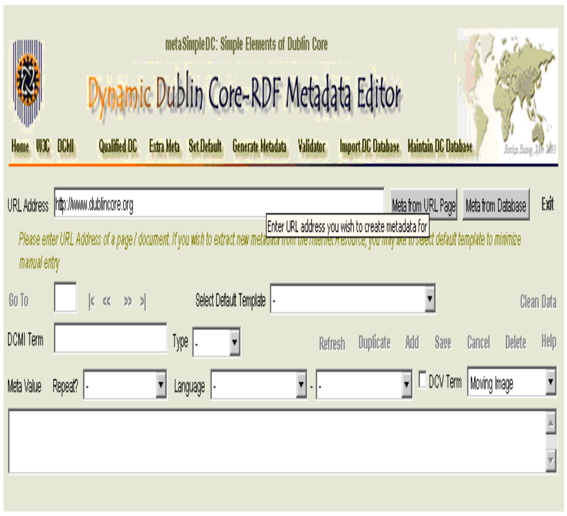
Please update the new and / or changed elements in Import DC Database before creating a met

INTRODUCTION

e to Dynamic Dublin Core-RDF Metadata Editor (DDC-RDF)! Like other metac
ive of this software tool is to facilitate you to create accurate and cc
et page. This rich-featured editor introduces few features that are not

Fig. 11 Main Page of D-DC/RDF-Editor

Fig. 12 shows the screen that is typically used in the first step of Simple DC metadata creation. The user starts by typing the URL address of the website that he would like to create the metadata for.


metaSimpleDC: Simple Elements of Dublin Core
Dynamic Dublin Core-RDF Metadata Editor
Home W3C DCMI Qualified DC Extra Meta Set Default Generate Metadata Validator Import DC Database Maintain DC Database
Meta from URL Page
Meta from Database
Exit
Go To
Clean Data
Refresh Duplicate Add Save Cancel Delete Help

URL Address http://www.dublincore.org

Enter URL address you wish to create metadata for

Please enter URL Address of a page / document. If you wish to extract new metadata from the internet resource, you may like to select default template to minimize manual entry

Select Default Template -

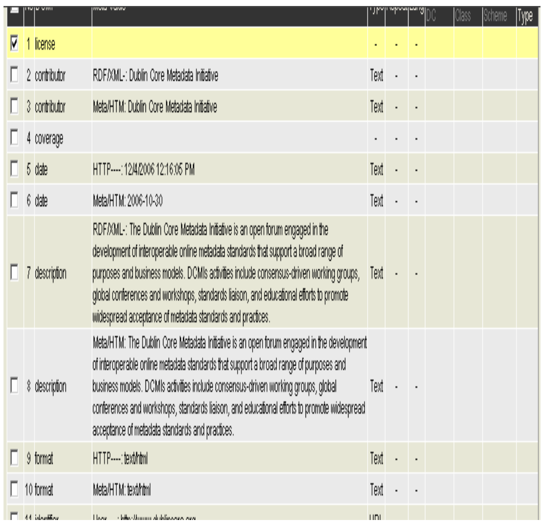
DCMI Term

Type -

Meta Value Repeat? - Language - - - DCV Term Moving Image

Fig. 12 Simple DC Metadata Screen

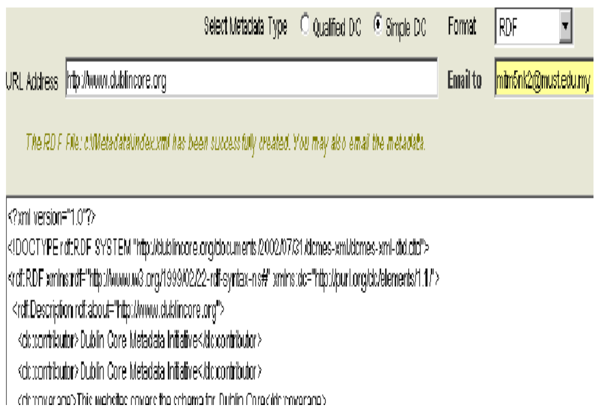
Fig. 13 then shows the display of the extracted metadata content from the target website.



| | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | DC | Class | Scheme | Type |
|---|---|---|---|---|---|---|---|---|---|---|
| ☑ | 1 | license | | - | - | - | | | | |
| ☐ | 2 | contributor | RDF/XML-: Dublin Core Metadata Initiative | Text | - | - | | | | |
| ☐ | 3 | contributor | Meta/HTM: Dublin Core Metadata Initiative | Text | - | - | | | | |
| ☐ | 4 | coverage | | - | - | - | | | | |
| ☐ | 5 | date | HTTP----: 12/4/2006 12:16:05 PM | Text | - | - | | | | |
| ☐ | 6 | date | Meta/HTM: 2006-10-30 | Text | - | - | | | | |
| ☐ | 7 | description | RDF/XML-: The Dublin Core Metadata Initiative is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models. DCMIs activities include consensus-driven working groups, global conferences and workshops, standards liaison, and educational efforts to promote widespread acceptance of metadata standards and practices. | Text | - | - | | | | |
| ☐ | 8 | description | Meta/HTM: The Dublin Core Metadata Initiative is an open forum engaged in the development of interoperable online metadata standards that support a broad range of purposes and business models. DCMIs activities include consensus-driven working groups, global conferences and workshops, standards liaison, and educational efforts to promote widespread acceptance of metadata standards and practices. | Text | - | - | | | | |
| ☐ | 9 | format | HTTP----: text/html | Text | - | - | | | | |
| ☐ | 10 | format | Meta/HTM: text/html | Text | - | - | | | | |
| ☐ | 11 | [illegible] | [illegible] | [illegible] | | | | | | |

Fig. 13 Metadata extracted from the URL address

After the extracted metadata has been displayed and edited according to the user's preferences, the updated metadata are then generated automatically in RDF/XML format as shown in Fig. 14.



Select Metadata Type: [✓ Simple DC] Format: RDF

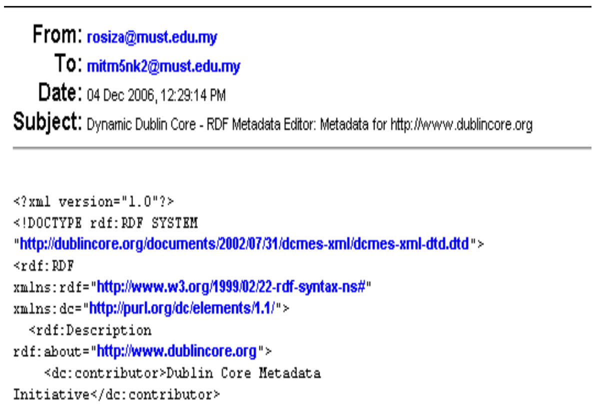
URL Address: http://www.dublincore.org Email to: mitm5nk2@must.edu.my

The RDF File: c:\MetadataIndex.xml has been successfully created. You may also email the metadata.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF SYSTEM "http://dublincore.org/documents/2002/07/31/dcmes-xml/dcmes-xml-dtd.dtd">
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/">
 <rdf:Description rdf:about="http://www.dublincore.org">
  <dc:contributor> Dublin Core Metadata Initiative</dc:contributor>
  <dc:contributor> Dublin Core Metadata Initiative</dc:contributor>
  <dc:coverage>This websites covers the schema for Dublin Core</dc:coverage>
```

Fig. 14 Generated metadata in RDF/XML format

The user can then "cut and paste" the newly generated RDF/XML formatted metadata into another document or e-mailed to someone as shown in Fig. 15.



**From:** rosiza@must.edu.my
**To:** mitm5nk2@must.edu.my
**Date:** 04 Dec 2006, 12:29:14 PM
**Subject:** Dynamic Dublin Core - RDF Metadata Editor: Metadata for http://www.dublincore.org

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF SYSTEM
"http://dublincore.org/documents/2002/07/31/dcmes-xml/dcmes-xml-dtd.dtd">
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description
rdf:about="http://www.dublincore.org">
    <dc:contributor>Dublin Core Metadata
Initiative</dc:contributor>
```

Fig. 15 Sample of RDF/XML File sent to an Email

## 7. Concluding Remarks

The need to have a powerful metadata editor is important in order to create good metadata for Internet resources that conforms to the standards supported by W3C as well as the industry. We have presented the salient features, design and implementation aspects of a metadata editor that uses automated metadata extraction from web sources as the basis. This automated metadata extraction feature not only alleviates the tedious manual process of creating metadata for any given web source but also ensures metadata consistency and integrity. In addition, through information extraction from metadata schemas, updates on metadata element names and other relevant information are also achieved dynamically.

Currently the prototype implementation of the editor focuses on supporting the authoring of DC metadata and their generation into RDF/XML format. However, the design of the editor is flexible enough to accomodate any future expansion to support other metadata schemas such as VCard, VCalendar, Content Standards for Digital Geospatial, and most importantly any user-defined schema.

**Acknowledgments**

We acknowledge the assistance given by the Context Interchange Systems Laboratory (CILS) at MIT Sloan School of Management in the use of Cameleon# within the work reported here.

## References

[1] O. Lassila, and R.R, Swick, "Resource Description Framework (RDF) Model and Syntax Specification", 1999. Available at : http://www.w3.org/TR/PR-rdf-syntax/
[2] D. Becket (Ed), "RDF/XML Syntax Specification (Revised)", 2004. Available at : http://www.w3.org/TR/rdf-syntax-grammar/
[3] Dublin Core Metadata Initiative Website, http://dublincore.org.
[4] P. Andy, "Dublin Core Metadata Editor", 2000. Available at : http://www.ukoln.ac.uk/metadata/dcdot
[5] DSTC Pty Ltd, "Reggie, The Metadata Editor Website", 1998. Available at : http://metadata.net/dstc
[6] K. Traugott, B. Mattias, and B. Mårten, "Dublin Core Metadata Template", 1998. Available at : http://www.lub.lu.se/cgi-bin/nmdc.pl
[7] A. Firat, S.E. Madnick, and M. Siegel, "The Cameleon Web Wrapper Engine", *Proceedings of the VLDB2000 Workshop on Technologies for E-Services*, September 2000.
[8] A. Firat, S. E. Madnick, N.A.Yahaya, W.K. Choo, and S. Bressan, "Information Aggregation Using Cameleon# Web Wrapper", *The 6th International Conference on Electronic Commerce and Web Technologies*, EC-Web 2005, Copenhagen, Denmark, 2005.
[9] C. Goh, S. Bressan, S. Madnick, and M. Siegel, "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information", *ACM Transactions on Information Systems*, Vol. 17, No. 3, (1999) 270-293.
[10] N.A. Yahaya, and R. Buang, "Automated Metadata Extraction from Web Sources", *The 3rd International Workshop on Web-Based Support Systems*, (WSS '06), Hong Kong, China, 2006.

**Nor Adnan Yahaya** obtained his PhD in Computer Science from Northwestern University, USA in 1987. He is an Associate Professor of IT at the Malaysia University of Science and Technology (MUST). His current research activities are focused on the development of tools and innovative applications related to emerging Web technologies such as web aggregation, web services, web agents, and the Semantic Web.

**Rosiza Buang** obtained her Master of Science (Information Technology) from Malaysia University of Science and Technology (MUST) in 2006. She is working as Knowledge Management (KM) Senior Executive at Technology, Capability & Data Management Department in Petroleum Management Unit, PETRONAS. Currently, she is involved in many KM initiatives including preservation of corporate knowledge and instillation of KM habits and mindset among others.

**Noor Hafizah Hassan** was a Research Officer at the Malaysia University of Science and Technology (MUST). She obtained her Bachelor of Science (Computer Science) from the University of Malaya, Malaysia in 2006 and currently pursuing her Master of Software Engineering degree from the same university.