DICTIONARY MANAGEMENT SYSTEM FOR THE DEB DEVELOPMENT PLATFORM

Aleš Horák, Adam Rambousek
Faculty of Informatics, Masaryk University, Botanická 68a, 602 00 Brno, Czech Republic
hales@fi.muni.cz, xrambous@fi.muni.cz

Keywords: electronic dictionaries; dictionary management; dictionary writing system; DEB development platform.

Abstract: In the paper, we introduce new dictionary management interface for design, preparation and presentation of generic electronic XML dictionaries using the DEB (Dictionary Editing and Browsing) development platform. The DEB platform provides a strict client-server environment for general dictionary writing systems. So far several successful NLP tools have been implemented on this platform, one of the most known being the DEBVisDic tool for wordnet semantic network editing and visualization.
This paper describes a new part of the DEB platform – the Administration interface that is shared by all DEB applications running on one server machine.

1 INTRODUCTION

The usage of electronic dictionary resources in the computational linguistics has increased several times during the last decade due to evident reasons: 1) the computer equipment has reached such high state that it is feasible to process giga bytes of textual data (Graff, 2003), and 2) the effectiveness gain is enormous when compared to the previous types of slow "manual" processing of linguistic information.

Despite this fact, the list of generally available dictionary writing systems (DWS) is not very wide – we may refer to e.g. Longman Dictionary Publishing System (McNamara, 2003), TshwaneLex (Joffe and de Schryver, 2004) or the Dictionary Editor and Browser (DEB) (Horák et al., 2006; Horák and Pala, 2006). All the cited systems are based on XML databases that allow to capture practically any kind of structural data including monolingual and translational dictionaries, thesauri or encyclopediae. Longman DPS and TshwaneLex are self contained commercial applications that are designed for specific purposes – Longman DPS is used in the publishing house to bring out traditional paper based dictionaries as well as new electronic and on-line products. TshwaneLex on the other hand is a dictionary compilation system that allows to create and maintain several dictionary styles for various purposes. Both these systems are distributed on a commercial base.

The third system, the DEB platform, is an open source and freely available development framework developed at the Natural language processing Centre at Masaryk University, Czech Republic. The system provides strict client server architecture for design of completely versatile dictionary applications. In next sections, we shortly summarize the features of the system as a whole and then describe its new component, the Administration interface that is used by all client applications.

2 THE DEB DEVELOPMENT PLATFORM

The Dictionary Editor and Browser was first designed as a standalone program for writing dictionaries. After several problems with adaptation of the tool for coming new requirements, the second version, sometimes referred to as DEB II, became a complete rewrite of the system based on open standards.

2.1 General Features

The most important property of the system is the *client-server* nature of all DEB applications. This provides the ability of distributed authoring teams to work fluently on one common data source. The actual development of applications within the DEB platform can be divided into the server part (the server side functionality) and the client part (graphical interfaces with only basic functionality). The server part is built from small parts, called *servlets*, which allow a modular composition of all services. The client applications communicate with servlets using the standard web protocol HTTP.

Since the data on the server is stored in XML, the actual data storage backend is provided by Berkeley DB XML (Chaudhri et al., 2003), which is an open source native XML database providing XPath and XQuery access into a set of document containers.

The user interface, that form the most important part of a client application, usually consists of a set of flexible forms that dynamically cooperate with the server parts. According to this requirement, DEB has adopted the concepts of the Mozilla Development Platform (Feldt, 2007). Firefox Web browser is one of the many applications created using this platform. The Mozilla Cross Platform Engine provides a clear separation between application logic and definition, presentation and language-specific texts.

2.2 Current Client Applications

Current development of the DEB platform includes implementation of several real-life dictionary applications. We will shortly summarize the main ones here.

DEBDict general dictionary browser. This DEB client was first implemented as a demonstration application, however, during two year it has evolved into a central base of more than ten different large dictionaries with remote access users all over the world. The main features of DEBDict include:

- multilingual user interface (English, Czech, others can be easily added)
- queries to several XML dictionaries and encyclopediae (of different underlying structure) with the result passed through an XSLT transformation
- connection to Czech morphological analyzer
- connections to external websites (Google, Answers.com, Wikipedia)
- connection to a geographical information system (display of geographical links directly on their positions within a cartographic map) or any similar application

The current version of DEBDict provides a common interface to 11 dictionaries – 6 different dictionaries of Czech language, 3 Oxford English dictionaries, the Diderot Encyclopedia and the CIA World Factbook. All together these dictionaries offer searching of more then million dictionary entries of various styles.

DEBVisDic – complete new version of the successful wordnet semantic network editor and browser VisDic. Following its predecessor, DEBVisDic offers important qualities for design and preparation of multilingual WordNet semantic networks. VisDic was used as a main tool in the EuroWordNet (EuroWordNet, 1999) and Balkanet (Balkanet, 2002) projects where wordnets for 13 languages were developed.

DEBVisDic extends the list of functions necessary for wordnet editing like synset preview, two-directional graph browsing for synset relations or complex queries. New functionality offers a new windowed interface and remote team work capabilities. Important features are brought together with the new Administration interface described in this paper.

Currently, DEBVisDic is also used for preparation of new Polish, Hungarian, Slovenian and Afrikaans wordnets and it is proposed as the main tool for the prepared Global WordNet Grid.

PRALED is designed for the development of the Czech Lexical Database, CLD (Klímová et al., 2005). The PRALED client is used in the Institute of Czech Language, Czech Academy of Sciences (Prague) as a dictionary writing system for building CLD which is a large project planned for about 6 years from now. The goal is to develop a lexical database of contemporary Czech containing approximately 100.000 entries. An important new feature here is that PRALED will be linked to the Manatee/Bonito corpus manager and the Word Sketch Engine.

DEB CPA Corpus Pattern Analysis (CPA, (Hanks, 2004)) is a new technique for mapping meaning to words in text. No attempt is made in CPA to identify the meaning of a verb or noun directly, as a word in isolation. Instead, meanings are associated with prototypical sentence contexts. Concordance lines are grouped into semantically motivated syntagmatic patterns. Associating a "meaning" with each pattern is a secondary step, carried out in close coordination with the assignment of concordance lines to patterns.

CPA editing tool displays the list of verb entries, along with the information who and when updated

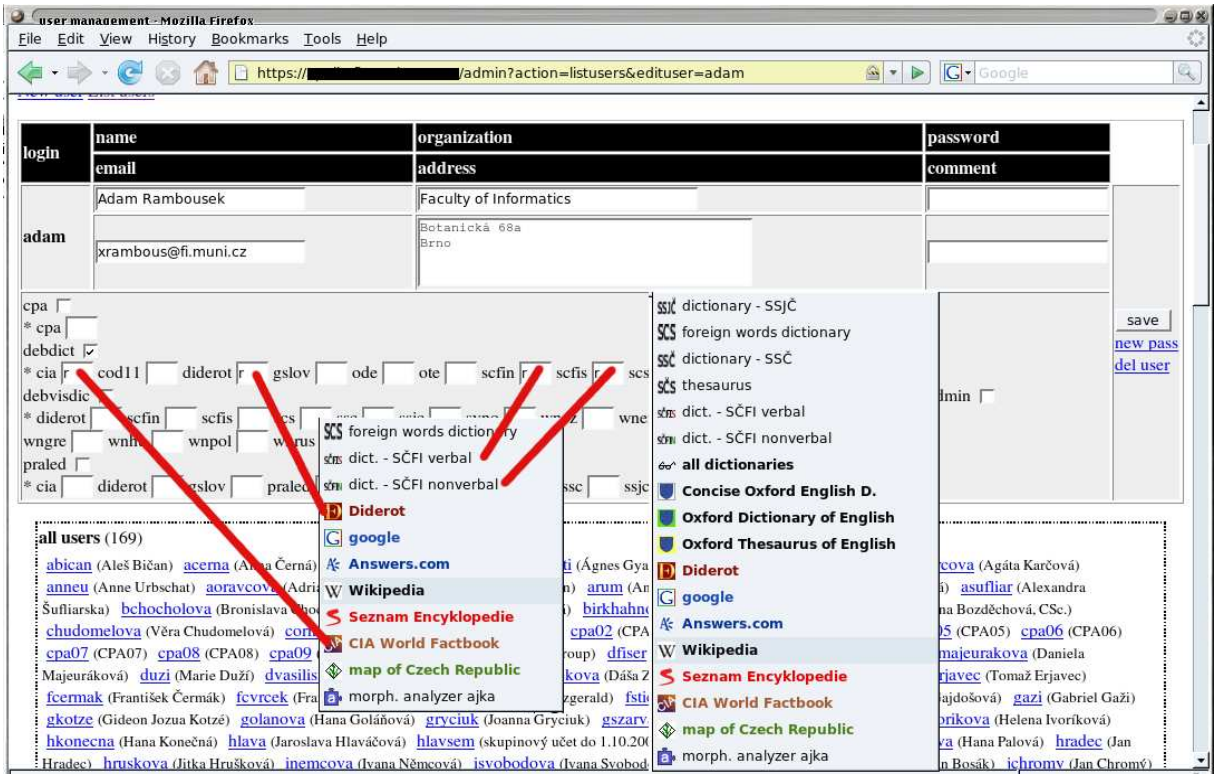


Figure 1: User management showing how access rights modify the dictionary list in DEBDict; list for selected user is on the left, list of all dictionaries is on the right.

the entry. Each entry consist of several patterns (the number of patterns is not limited) and it is possible to freely modify their order and content. The main part of the tool, the pattern editing window, allows to enter and modify all the information about one pattern. Examples documenting the pattern are taken from the British National Corpus using a modified version of Bonito corpus manager and the Word Sketch Engine (Kilgarriff et al., 2004) that is integrated to the DEB CPA tool.

Cornetto The Cornetto project (STE05039) is funded by the Nederlandse Taalunie in the STEVIN framework. The goal is to build a lexical semantic database for Dutch, covering 40K entries, including the most generic and central part of the language. Cornetto will combine the structures of both the Princeton Wordnet and FrameNet for English, by combining and aligning two existing semantic resources for Dutch: the Dutch wordnet and the Referentie Bestand Nederlands.

Since the lexicographic work on the Cornetto project consists in expert aligning and merging of two independent Dutch resource together with linking the data to the Princeton Wordnet, the project uses specific application implemented on top of the DEBVisDic tool.

3 THE NEW ADMINISTRATION INTERFACE

Initially, DEB server was developed with just command-line management of dictionaries and administration of user passwords for authentication. The configuration was realized by structured text files and data processing scripts.

After DEBVisDic has spread to more users worldwide and has been used for building several national Wordnets (Polish, Hungarian, Slovenian or Afrikaans), a more sophisticated administration interface for DEBVisDic users and dictionaries was created. Later on, this interface was transformed to more general and complex dictionary management application for the whole DEB server.

3.1 Overall Design Goals

The DEB server packages are currently being deployed on several servers in different organizations and often more than one user need to administer a single DEB server without having a direct server access. Thus, the administration interface must be accessible remotely and without any special tools. The best choice for this task is a web-based interface, where

the user needs just a web browser.

The interface should support easy administration of all the server areas. Of course, the main area of a dictionary management server is the dictionary management. Each dictionary is described with several basic attributes, like its name and code, the filename of its storage in the DB XML database, its dictionary type, the XML schema or indexed elements or XSLT templates for output displaying. Also, some projects may need extra specific settings – e.g. the DEBVisDic clients need to store information about the inter-dictionary links. After the dictionary is set up, the interface has to support import and export of XML data into and from the DB XML format.

When the administrator sets up the server dictionaries, these can be grouped to "services." A service is one individual part of the DEB server, usually used for one particular project. For example, DEBVisDic or DEBDict are separate services, but they share the same base libraries and management database. Several services can access the same dictionaries, each providing different view on the data.

The user accounts are shared between all the services. Thanks to the database sharing between services, each user needs just one account for all the services he or she may use. The administrator can restrict access to selected services and for each service, more detailed access permissions can be set for each dictionary (read-only, read-write, update, ...). The actual usage of the dictionary access permissions depends solely on the service. This means, one service can ignore permissions at all and another service can use complex access rights.

Apart from access rights, the user account management provides all the needed functions – it allows to create, modify and delete user accounts. Each user can log-in to the administration interface and change his or her password. In case the user forgets a password, he or sher can ask for a new random password.

To ease the deployment of the DEB platform, we are experimenting with automated creation of the client applications. Now, the server is able to create straightforward applications based on the Relax NG Schema (van der Vlist, 2003) of the dictionary, and we are aiming at automated creation of client packages for new national Wordnets.

Another very useful feature is uploading of files onto the server using the web interface. This way, the administrator can easily modify web page templates (XSLT) or other files without the need of direct (FTP, SSH) access to the server.

3.2 The Implementation

The server administration interface is based on the same postulates as the other DEB server dictionaries and modules. The Berkeley DB XML database provides a storage backend for the administration metadata. The server-side scripts are developed in Ruby programming language.

All the data about users, dictionaries, permissions and other control data are stored in the DB XML database in the XML format. Each dictionary module of the DEB server uses a common interface to access data from this administration database.

Example 1 XML entry for the user from the Figure 1

```
<user>
 <login>adam</login>
 <name>Adam Rambousek</name>
 <email>xrambous@fi.muni.cz</email>
 <org>Faculty of Informatics</org>
 <addr>Botanicka 68a, Brno</addr>
 <pass>3Ja8ivX12OB0U</pass>
 <services><service code="debdict">
  <dict code="scs" perm="r"/>
  <dict code="scfis" perm="r"/>
  <dict code="cia" perm="r"/>
  <dict code="scfin" perm="r"/>
  <dict code="diderot" perm="r"/>
 </service></services>
</user>
```

The administration module provides several services – user authentication, access rights control, entry locking and journaling of dictionary changes.

The administration interface is a web-based application where the web pages are generated using an HTTP template which allows easy design and content modification and then served to the users by a lightweight web server – WEBrick (Santoso, 2004). The users are authenticated using standard HTTP authentication mechanism. The administration module extends the standard interface for passwords stored in a file and loads user's login and password from the XML database. Each change in user accounts or access rights is propagated to all DEB services in the real-time.

3.2.1 The Dictionary Management

For each dictionary, the administrator has to define several attributes (see the Figure 2). The minimal set of attributes contains a unique dictionary code, a database filename and a dictionary class (the implementation class), the other attributes are more or less

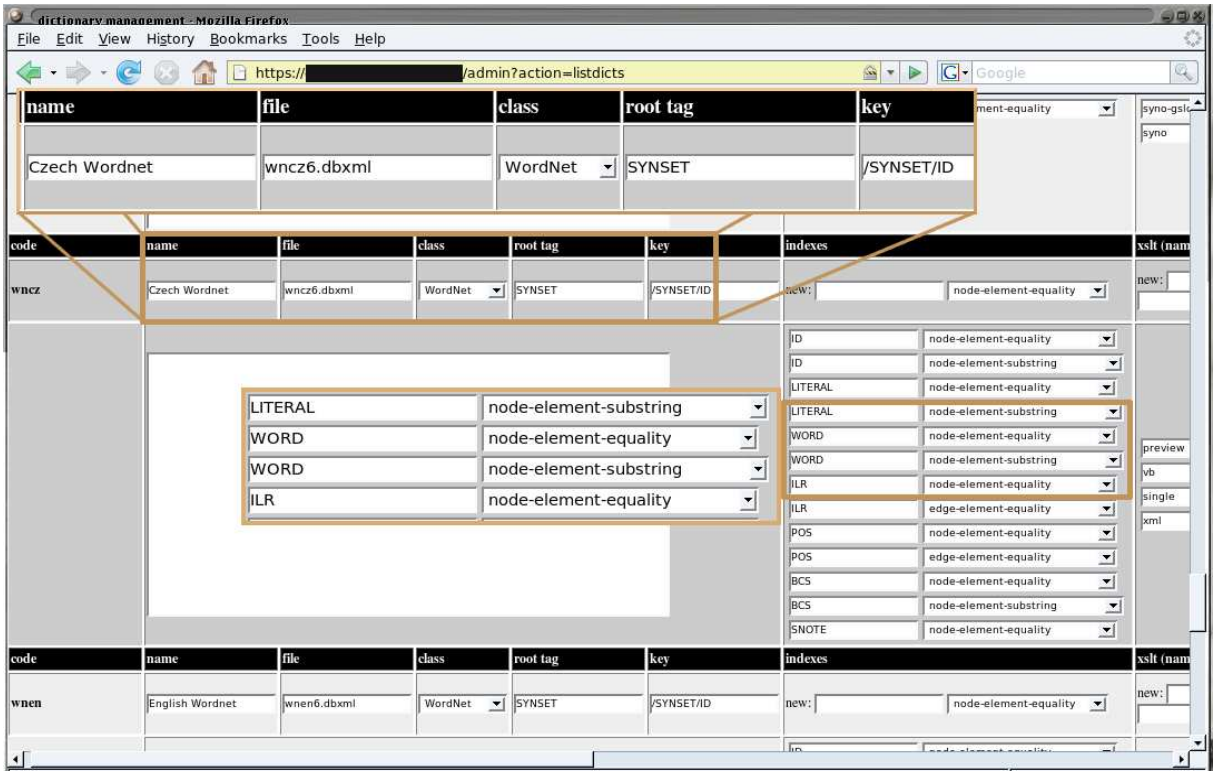


Figure 2: Dictionary management showing basic information and indexed elements for the Czech Wordnet dictionary.

optional. The meaning of the dictionary attributes is:

- The dictionary name is displayed to users by the client application.
- The definition of the XML entry root tag and its key element are needed for XML import and for searching (in case, the application does not have its own, more complex search method).
- Indexes speed up search operations, so each element or attribute that is used in user queries should be indexed.
- The XSLT templates transform XML data to another form suitable for presentation or machine processing.

Extra dictionary attributes are required for the DEBVisDic dictionaries:

- Each DEBVisDic dictionary is linked to the client software by the client package code.
- The DEBVisDic Dictionaries can reference to each other using “equivalence tags.”
- In the next field, the administrator can enter dictionaries that should be reloaded after an edit action in the client (usually in another dictionary).
- And the last option specifies related dictionaries – for example, several national Wordnets linked with ILI (Inter-Lingual Index). It is possible to display the same entry in different languages or to copy entries between languages.

3.2.2 Import and Export

The import function takes an XML file and stores the data into the DB XML database. The XML file has to be uploaded to the server (it is possible to upload it through web interface). All entries must share the same root tag (specified in the dictionary management), entries with different root tags are ignored. The administrator can choose if he or she wants to delete all the entries from database before the import or just add the new entries. The import utilizes two methods for XML reading. The first method loads the whole XML file into memory and uses an XML parser on the big document. This method is accurate, unfortunately it has exponential time complexity, so it can take hours for large XML files (over 10 MB). The second method uses regular expressions to read entries one by one from the XML file and then each single entry is parsed. Entries are stored in the database with value of the specified key tag as a unique key. The administrator is informed about the import progress on the web page – a number of processed entries, a total number of entries, an estimated time till the end and last ten entry keys are displayed.

The administration module also supports export

from database to plain XML file, the output files may be compressed to save disk space. The export also has an option to save the file in the form of a Ruby language script that will setup the database and import initial data. This is needed for the administration database itself. The output files are saved in a specified directory on the server and the administrator is informed about the export progress. Once the export ends, the administrator is offered a link to download the file through the web interface. The same function is used also for daily database backup.

3.2.3 Locking and Sequences of Identifiers

The administration interface offers entry locking management to other DEB server modules. If multiple users can edit the database at the same time (which is one of the basic advantages of the client-server architecture), it is crucial to provide exclusive write locking of entries so that two users are not able to edit the same entry at a time. Decisions about entry locking depends on each application design:

- when should an entry be locked and unlocked?
- should only the edited entry be locked or should the locking affect other entries too?

An application then sends the request to the administration module which updates the lock database. The administration module provides several functions – besides simple lock and unlock functions, it can tell which user has locked a given entry, return the list of locks for selected user and/or dictionary or group several locks together if they are related. The administrator has access to the list of all locks and he or she can also delete chosen locks if the application did not release them correctly.

Newly created entries should have a unique identifier. If the application does not generate its own identifiers, the administration module can provide such service. It is possible to set an identifier pattern for each dictionary – this pattern looks like `CZE-[id]` and [id] will be replaced with sequentially increased number. The administrator can also affect the number used.

3.2.4 The Installation Packages

The administration interface supports automated creation of Firefox Extension installation packages (XPI). If the administrator specifies a Relax NG schema for the dictionary, it is possible to transform this schema to an application design description in the XUL description language and the supporting code in JavaScript. The application created in this way supports basic forms – single and multiple text fields, select-boxes of specific values or relational links to other dictionaries. It can serve as basis for custom modifications. Of course, the application is able to connect to server, load data from server and save a modified entry back. We are currently working on more complex support for creation of new packages, mainly for the DEBVisDic client packages.

4 USAGE SCENARIO – HOW TO MAKE A SAMPLE DICTIONARY

4.1 New Dictionary Definition

As a first step, the administrator needs to provide basic information about the dictionary. It does not matter if there is already an existing dictionary full of data, or whether the dictionary is going to be built from scratch. The administrator must specify an entry root element, where to find the unique key, several indexes and an XML schema of the entry.

Let us create a demonstration dictionary from scratch, we will name the root element `entry` and have the unique key identifier in the element `/entry/headword`. The corresponding Relax NG schema is given in the Example below.

Example 2 Relax NG schema

```
<element name="entry">
  <element name="headword">
    <attribute name="pos">
      <text/>
    </attribute>
    <text/>
  </element>
  <oneOrMore>
    <element name="sense">
      <text/>
    </element>
  </oneOrMore>
</element>
```

This schema describes entry with one `headword` element, with `pos` attribute, and one or more `sense` elements. Of course, Relax NG supports description of much more complex XML structures.

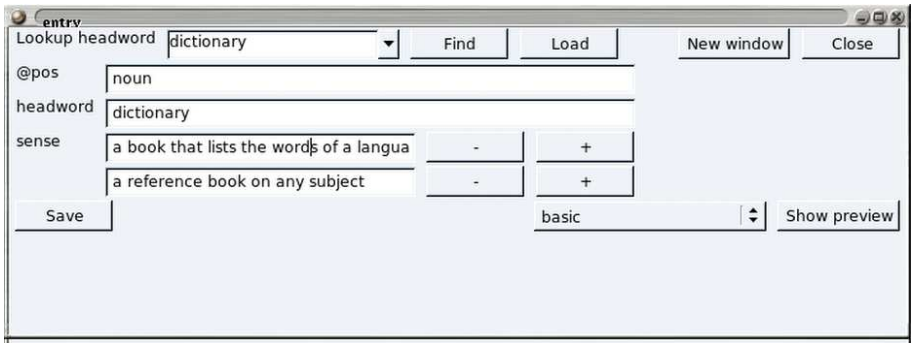


Figure 3: Sample automatically build client application

4.2 Preparation of an Installation Package

The preparation of a new basic client application package requires selection of a dictionary and running the package generation. The administration module checks the Relax NG schema and finds all elements or attributes that contain `text` child element. All such elements and attributes are transformed to XUL textbox fields with the respective name as a label describing the field. If an element can occur multiple times in the entry (like `sense` in our Example), buttons for adding and removing the textbox are added to the application form, too.

The created JavaScript supports loading and saving documents and also searching for documents. The application thus enables querying each indexed field specified in the dictionary management interface. For example, users can easily find all nouns.

All the created application files are then packaged into the Firefox extension installation package (XPI). Users can download this package for installation or individual files for editing. An example of the resulting application is shown on the Figure 3.

For the new client, there are also two basic preview templates (in XSLT) saved on the server side. One provides basic entry preview displaying all the data and the second displays raw XML data.

4.3 Application Customization

Thanks to the design of applications based on the Mozilla development platform, these applications are easily customizable.

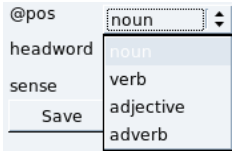
Any change in the layout and design of the form is done by editing the XUL (XML User-interface Language) files accompanied with standard CSS stylesheets. The application logic (i.e. procedures implemented in JavaScript) stays the same for a new layout. Combination of XUL and CSS languages is very powerful and supports long list of features that are commonly used in desktop applications. For example, we can change PoS textbox field into a drop-down list.

Example 3 Produces textbox field

```
<textbox id="entry.headword.@pos"/>
```

Example 4 Produces drop-down list

```
<menulist id="entry.headword.@pos">
  <menupopup>
    <menuitem label="noun"/>
    <menuitem label="verb"/>
    <menuitem label="adjective"/>
    <menuitem label="adverb"/>
  </menupopup>
</menulist>
```



As we can see, the field labels contain element names only. This allows the application designer to change them to something human-readable. The actual texts are stored in a DTD (Document Type Definition) file as XML entities, so they can be adjusted to any texts in one place. Moreover, this mechanism is also used for localization of the application. It is possible to include several DTD files for different languages into installation package and (automatically) switch between them.

After all the application source files are modified to meet the designer's requirements, he or she can upload them using the administration interface and let it build a new version of the installation package.

The application designer can also supplement the dictionary editor with more preview templates or

Example 5 A field label and the respective entity in the DTD file.

```
<label value="&entry.headword;"/>

<!ENTITY entry.headword "headword">
```

modify the existing ones for different data presentation. When adding a new template, the template name must be added to the dictionary description in the database management interface. The modified templates are again uploaded to the server using the administration interface.

5 CONCLUSION

We have presented a new common administration module in the DEB development platform that is shared by all kinds of DEB client applications. With this interface, the DEB platform provides a invaluable basis for new dictionary writing applications for all purposes and all types of dictionaries

Even though the DEB platform is developed as open source and free platform, we believe that already in this stage of development it offers interesting features for DWSs and that it can speed up the development in this area. The applicability of the platform is best justified with the implemented clients described in this article and with the 200 registered users that currently access those applications (counting only the accesses to the Masaryk University DEB server).

ACKNOWLEDGEMENTS

This work has been partly supported by the Academy of Sciences of Czech Republic under the projects T100300414 and T100300419, by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009.

REFERENCES

Balkanet (2002). Balkanet project website, `http://www.ceid.upatras.gr/Balkanet/`.

Chaudhri, A. B., Rashid, A., and Zicari, R., editors (2003). *XML Data Management: Native XML and XML-Enabled Database Systems*. Addison Wesley Professional.

EuroWordNet (1999). EuroWordNet project website, `http://www.illc.uva.nl/EuroWordNet/`.

Feldt, K. (2007). *Programming Firefox: Building Rich Internet Applications with Xul*. O'Reilly.

Graff, D. (2003). English Gigaword. Technical Report LDC2003T05, Philadelphia, PA USA.

Hanks, P. (2004). Corpus pattern analysis. In *Proceedings of the Eleventh EURALEX International Congress*, Lorient, France. Universite de Bretagne-Sud.

Horák, A. and Pala, K. (2006). DEB tools for merging linguistic resources. In *Proceedings of the Workshop on Layering Linguistic Information, LREC 2006*, pages 55–61, Italy. ELRA.

Horák, A., Pala, K., Rambousek, A., and Rychlý, P. (2006). New clients for dictionary writing on the DEB platform. In *DWS 2006: Proceedings of the Fourth International Workshop on Dictionary Writings Systems*, pages 17–23, Italy. Lexical Computing Ltd., U.K.

Joffe, D. and de Schryver, G.-M. (2004). TshwaneLex – professional off-the-shelf lexicography software. In *Third International Workshop on Dictionary Writing Systems: Program and List of Accepted Abstracts*, Brno, Czech Republic. Masaryk University, Faculty of Informatics.

Kilgarriff, A., Rychlý, P., Smrž, P., and Tugwell, D. (2004). The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pages 105–116, Lorient, France. Universite de Bretagne-Sud.

Klímová, J., Oliva, K., and Pala, K. (April 2005). Czech lexical database – first stage. In *Short Proceedings of Complex Conference 2005*, Budapest, Hungary.

McNamara, M. (2003). Dictionaries for all: XML to final product. In *XML Conference 2003*, Philadelphia, USA.

Santoso, Y. (2004). Gnome's Guide to WEBrick. (`http://microjet.ath.cx/WebWiki/WEBrick.html`).

van der Vlist, E. (2003). *RELAX NG*. O'Reilly Media.