Léacslann: A platform for building dictionary writing systems

Michal Boleslav Měchura

Keywords: *dictionary writing systems, terminology management systems, e-lexicography, databases*.

Abstract

The purpose of this demo is to introduce *Léacslann*, a new platform for building dictionary writing systems (DWS) and terminology management systems (TMS) as well as other lexicographic and reference applications. Léacslann can be used without any knowledge of programming to create a basic lexical database with an arbitrary structure. This will be demonstrated in the first half of the demo, while the second half will show how a software developer can customize Léacslann for more demanding applications.

1. Introduction

For an organization planning to create a new lexical database, there are two approaches to acquiring the necessary software tools. One is to license an existing software package such as TshwaneLex, IDM (for lexicography) or SDL Trados Multiterm (for terminology); the other is to develop a custom system from scratch, either in-house or by outsourcing to a commercial software developer. Both approaches have disadvantages. Ready-made software often imposes structures that do not match the intended application well, while developing one's own system from scratch is expensive. This paper will introduce a software system that offers a compromise between these two extremes.

Even though all lexical databases are different, they all have some features in common. A lexical database typically consists of *entries* where each entry has a strictly defined internal structure defined by elements such as text strings and numbers with rules for combining them together. A lexical database system typically provides features for editing, displaying and searching entries, features for keeping track of an entry's editorial history, features for managing user access rights, and so on.

Our school[1] is engaged in the management of several large lexical databases, including the National Terminology Database for Irish (focal.ie) and the Placenames Database of Ireland (logainm.ie). We have developed a system called *Léacslann*[2] which provides a common infrastructure for all our databases.[3] Léacslann is a platform for working with collections of entries of arbitrary structure. Such collections are called *stocks* in Léacslann. Léacslann provides features shared by all stocks, while allowing for application-specific customization. In this sense, Léacslann is not a dictionary writing system or a terminology database; instead, it is a *platform* for building such systems.

Léacslann can be used to work with various kinds of stocks such as monolingual and bilingual dictionaries, terminology databases or indeed any sort of reference work including a placenames database, a collection of proverbs, a bibliography database or a library catalogue. Léacslann stocks can accommodate any language and any combination of languages, as long as text in those languages can be encoded in Unicode.

In simple cases, a new stock can be set up in Léacslann in just a few minutes, without any knowledge of programming. This will be demonstrated in the next section. In more complicated scenarios, Léacslann can be used as a platform for building a customized application, and this will be discussed in the remainder of the paper.

2. Working with Léacslann

This section will demonstrate how to set up a new stock in Léacslann in a completely self-service way, without any knowledge of programming. To access Léacslann for the purposes of this demonstration, go to `http://lxln.prettydata.eu` and log in with the name `demo` and the password `demo`.

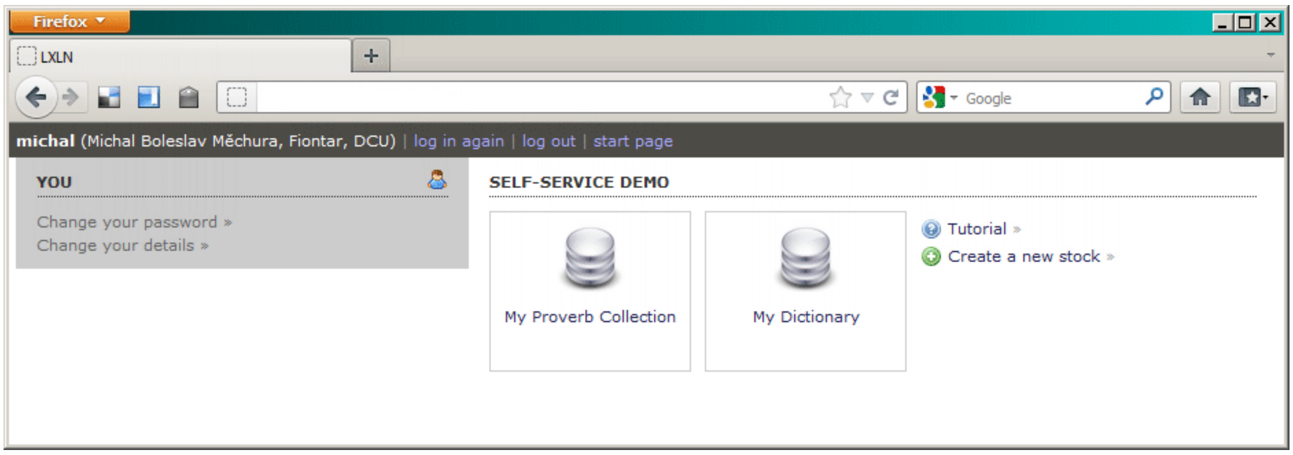
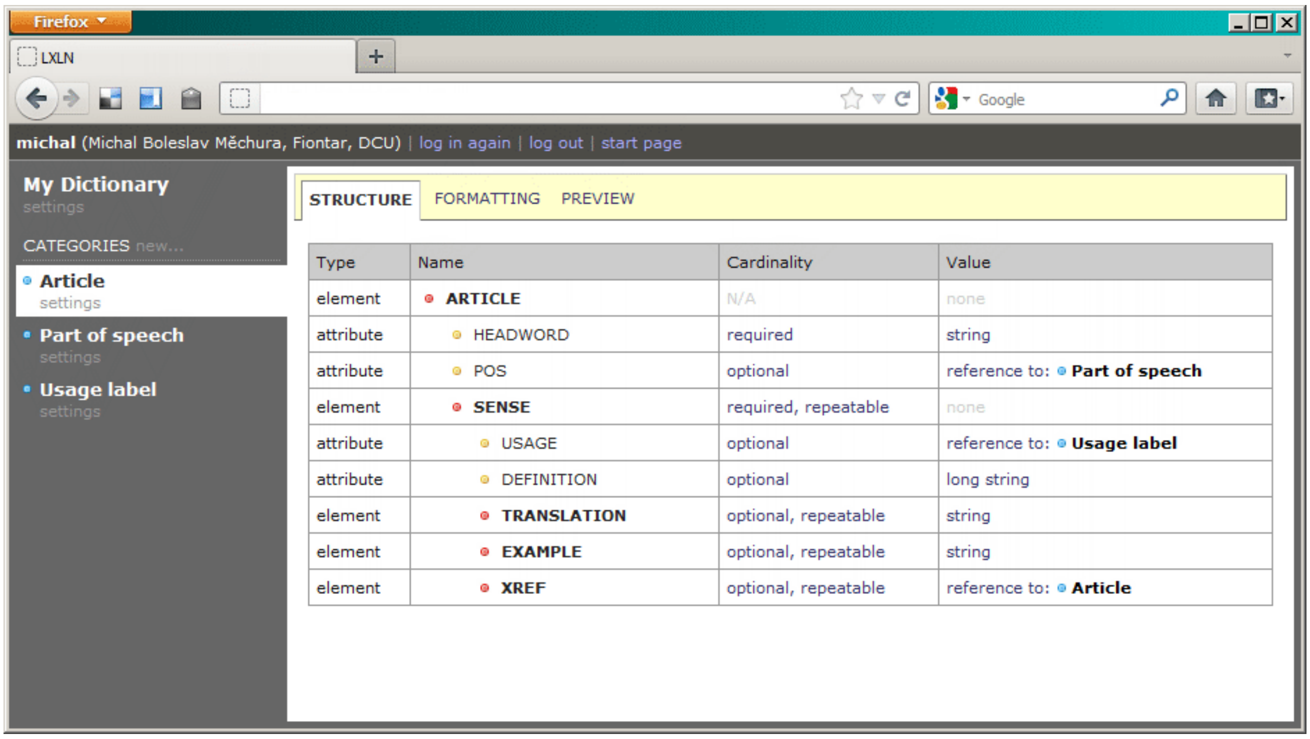


Figure 1. Léacslann start screen after logging in.

Remember that *stock* is a Léacslann term for an arbitrary collection of entries, such as a dictionary or a terminology database. To create a new stock, click the appropriate link on the start page (Figure 1). You will be presented with a user interface where you can specify which *categories* of entries your new stock will contain, and what their internal structure will be (Figure 2).

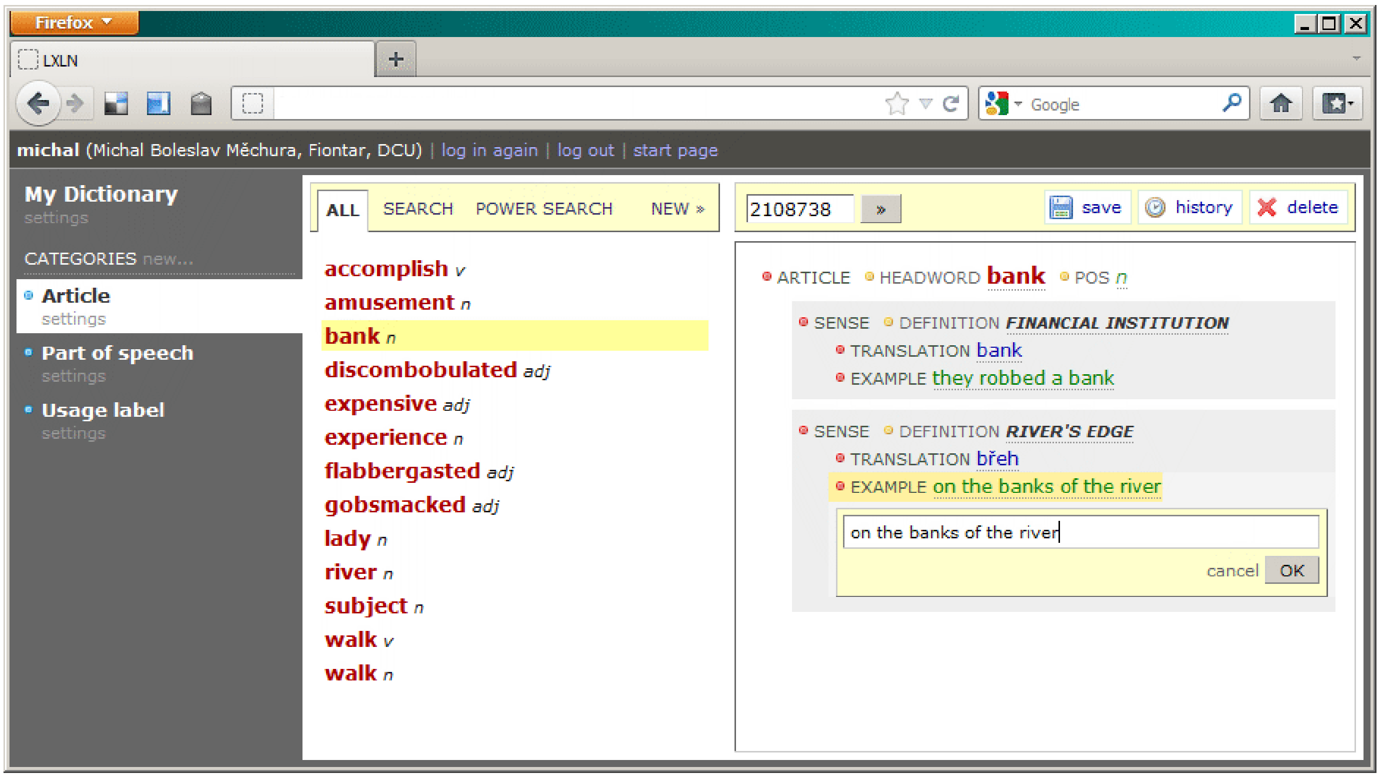


Type	Name	Cardinality	Value
element	ARTICLE	N/A	none
attribute	HEADWORD	required	string
attribute	POS	optional	reference to: Part of speech
element	SENSE	required, repeatable	none
attribute	USAGE	optional	reference to: Usage label
attribute	DEFINITION	optional	long string
element	TRANSLATION	optional, repeatable	string
element	EXAMPLE	optional, repeatable	string
element	XREF	optional, repeatable	reference to: Article

Figure 2. Editing a category’s structure.

A Léacslann entry consists of *elements*. For each category, you must specify what elements the entries can contain and how the elements can be combined. Each element has a name and may also have: (1) a value, such as a string of text or a number, (2) one or more attributes, each of which also has a name and a value, and (3) one or more child elements. Each entry has one *root element* whose name is the same as the name of the category. All other elements are listed as children of this root element, or of one of its children, or its children's children, and so on. You can think of an entry as a multi-level bulleted list of elements.

When selecting the data types for element values and attribute values, one data type you can select is *reference*. A reference is a pointer from one entry to another. The term *reference* has a broader meaning in Léacslann than it normally does in lexicography. It encompasses not only classical cross-references from one dictionary entry to another but also references from a terminological concept to the terms that lexicalize it, from a concept to the domain (or domains) it belongs in, from a domain to its parent domain (e.g. *Microbiology* → *Biology*), and so on. Metadata elements such as domain labels or part-of-speech labels can exist as *entries* in your stock, and other entries (such as dictionary articles) can refer to them.



accomplish *v*
amusement *n*
bank *n*
discombobulated *adj*
expensive *adj*
experience *n*
flabbergasted *adj*
gobsmacked *adj*
lady *n*
river *n*
subject *n*
walk *v*
walk *n*

- ARTICLE
  - HEADWORD **bank**
  - POS *n*
  - SENSE
    - DEFINITION ***FINANCIAL INSTITUTION***
    - TRANSLATION bank
    - EXAMPLE they robbed a bank
  - SENSE
    - DEFINITION ***RIVER'S EDGE***
    - TRANSLATION břeh
    - EXAMPLE on the banks of the river

on the banks of the river

Figure 3. Listing and editing entries in Léacslann.

Having specified the categories you want in your stock, you can proceed to creating entries. Your new stock will now be listed on Léacslann's start page and clicking it will take you to a user interface where you can search, display and edit entries (Figure 3). Léacslann provides you with the following user-interface features:

1. An **entry editor** which presents each entry as a multi-level bulleted list of elements. Click on each value to edit it, and click on each element name or attribute name to display additional options for adding attributes and child elements, for moving elements up and down, for removing elements and attributes, and so on. The entry editor uses the category specification you supplied earlier and makes sure that the entry you are composing complies with the structure you specified. While editing an entry, Léacslann keeps a record of all previous versions of the entry and gives you options to restore the entry to a previous state.

2. A **search** feature which allows you to find entries quickly by typing a piece of text. This works by looking for your search term in all elements whose data type is *string* or *long string*.

3. A **power search** feature which allows you to search for entries with an arbitrary combination of criteria. This can range from simple queries ('give me all articles where the headword is a noun') to complex ones ('give me all articles where the headword is a noun, that have more than three senses and where at least one sense has no usage examples'). Your queries can also follow the trail of references, so you can search for entries that refer to entries that fulfil certain criteria.

3. Under Léacslann's hood

The previous section has demonstrated how to set up and use a basic stock in Léacslann. For more complex requirements, Léacslann can be used to build customized applications. Some knowledge of programming is required for this, as well as knowledge of the internals of Léacslann. These will be outlined in this section.

The basic unit of analysis in Léacslann is an *entry*. An entry is a small XML document. Each entry belongs to a *category* which dictates its internal structure (much like a DTD or an XML Schema). Léacslann does not avail of the full expressivity of XML. Instead, it only accepts XML entries that meet certain restrictions, the most important of which is that no text nodes are allowed: all values must be stored as attributes. Due to this simplified XML structure, the nodes inside entries can be mapped to table rows in a relational database; because of this, Léacslann entries lend themselves relatively easily to automated, application-neutral tasks such as indexing and change tracking. Listings 1a and 1b show two sample Léacslann entries. Notice that the entry in Listing 1a contains a reference to the entry in Listing 1b.

Listing 1a. A sample Léacslann entry.
```
<!--ENTRY ID 2145-->
<article headword="bank" pos="3421"/>
      <sense definition="financial institution">
            <translation default="banc"/>
      </sense>
      <sense definition="edge of river">
            <translation default="bruach"/>
      </sense>
</article>
```

Listing 1b. A sample Léacslann entry.
```
<!--ENTRY ID 3421-->
<posLabel abbr="n" title="noun"/>
```

The category an entry belongs to further specifies the kinds of elements that may (or must) be present in the entry, the attributes they may (or must) have, and the values that are allowed in those attributes. The data types available as attribute values include basic ones such as string, number, date, time, Boolean (true/false), values from a list (where the value actually stored may be different from the caption displayed in the user interface) and references to other entries. Like entries, categories are encoded as XML documents. Listing 2a and 2b show two sample Léacslann categories. Notice that the entries in Listings 1a and 1b comply with the category specifications in Listings 2a and 2b. Notice also that the category in Listing 2a explicitly states that an `article` entry may contain a reference to a `posLabel` entry.

Listing 2a. A sample Léacslann category.
```
<category rootElement="article">
      <element name="article">
            <attribute name="headword" required="yes">
                  <literal type="string">
            </attribute>
            <attribute name="pos" required="no">
                  <xref category="posLabel"/>
            </attribute>
            <elementRef name="sense" required="yes" repeatable="yes"/>
      </element>
      <element name="sense">
            <attribute name="definition" required="no">
                  <literal type="string"/>
            </attribute>
            <elementRef name="translation" required="yes"
repeatable="yes"/>
      </element>
      <element name="translation">
            <literal type="string"/>
      </element>
</category>
```

Listing 2b. A sample Léacslann category.
```
<category rootElement="posLabel">
      <element name="posLabel">
            <attribute name="abbr" required="yes">
                  <literal type="string"/>
            </attribute>
            <attribute name="title" required="yes">
                  <literal type="string"/>
            </attribute>
      </element>
</category>
```

Léacslann provides a number of user-interface components which the application developer can use to build an application. The most important of these are the entry editor and the search engine demonstrated above.

When entries are displayed in the user interface, the developer can supply XSL style sheets to format them. (In the self-service demo, a default style sheet is generated automatically for each category.) These style sheets operate on a *precomposed* version of the entry, which consists of the entry itself as well as all other entries to which the entry refers or which refer to it, directly or indirectly.

In addition to features related to editing, searching and displaying entries, Léacslann provides a number of features for the management of users' access rights and for tracking entry changes, including the ability to undo changes. These features tend to be identical in all lexicography applications and Léacslann saves the developer time by providing them "out of the box" for all entry types.

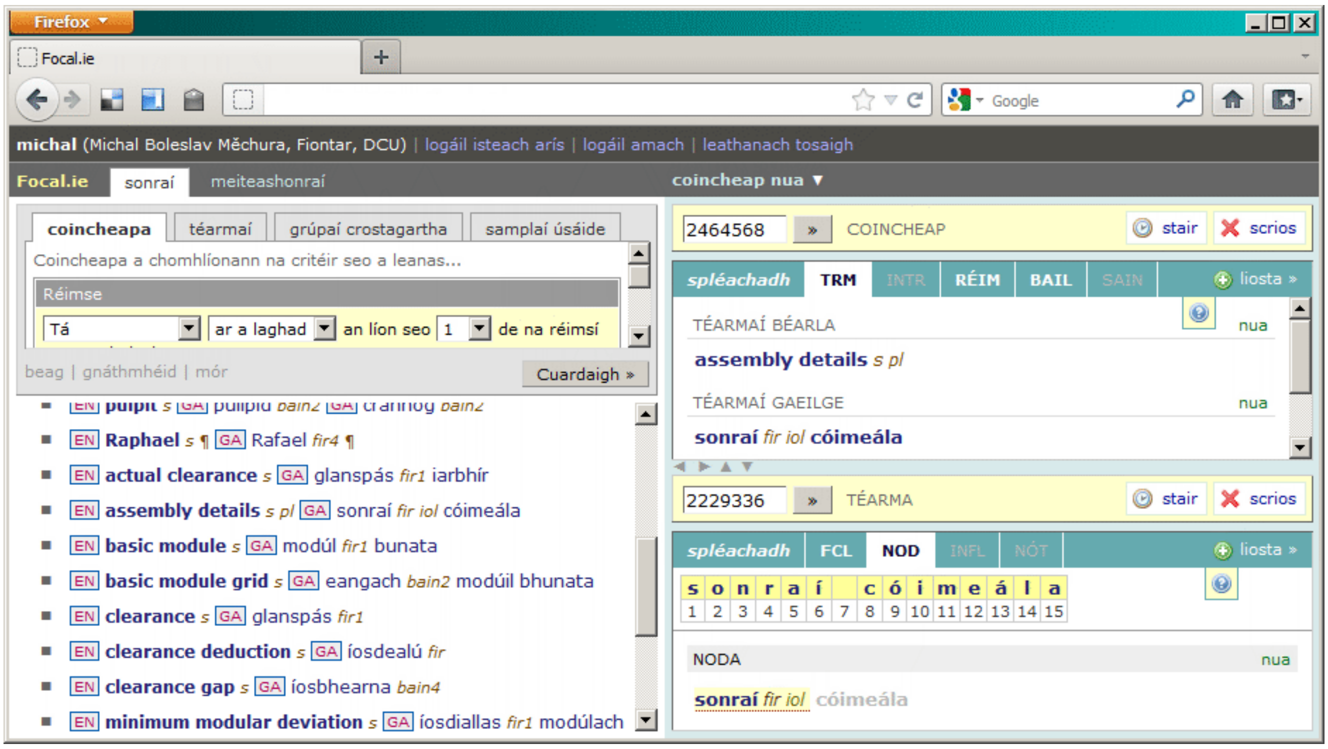
4. Building applications with Léacslann

To produce a customized dictionary writing system or a customized terminology management system, a software developer needs to build a Léacslann *application*. A Léacslann application is a web application embedded inside Léacslann which uses Léacslann as its data storage and re-uses some of Léacslann's user-interface components. Different users may have access to different applications, and a personalized list of available applications is displayed to each user on the start page. The self-service demo environment shown above is also a Léacslann application.

An application can provide custom editing and search features that override the built-in ones when needed, while reusing built-in components where appropriate. An application author can even decide not to re-use any of Léacslann components but still use Léacslann as back-end storage, availing of shared functionality such as editorial history and user rights management.

The effort needed to build an application in Léacslann will depend on the level of customization required. In the extreme case, when only the built-in editor and the built-in search engine are needed, it is possible to construct a basic application within a few hours. In any case, the development effort will be smaller than developing a new software tool from scratch.

For many of our terminology stocks, our school uses a highly customized Léacslann application with a custom entry editor and several custom search engines (see Figure 4). The effort required to develop this application was approximately four weeks for one full-time software developer.


michal (Michal Boleslav Měchura, Fiontar, DCU) | logáil isteach arís | logáil amach | leathanach tosaigh

Focal.ie | sonraí | meiteashonraí | coincheap nua ▼

coincheapa | téarmaí | grúpaí crostagartha | samplaí úsáide

Coincheapa a chomhlíonann na critéir seo a leanas...

Réimse

Tá | ar a laghad | an líon seo | 1 | de na réimsí

beag | gnáthmhéid | mór | Cuardaigh »

- EN **pulpit** *s* GA puilpid *bain2* GA crannóg *bain2*
- EN **Raphael** *s* ¶ GA Rafael *fir4* ¶
- EN **actual clearance** *s* GA glanspás *fir1* iarbhír
- EN **assembly details** *s pl* GA sonraí *fir iol* cóimeála
- EN **basic module** *s* GA modúl *fir1* bunata
- EN **basic module grid** *s* GA eangach *bain2* modúil bhunata
- EN **clearance** *s* GA glanspás *fir1*
- EN **clearance deduction** *s* GA íosdealú *fir*
- EN **clearance gap** *s* GA íosbhearna *bain4*
- EN **minimum modular deviation** *s* GA íosdiallas *fir1* modúlach

2464568 » COINCHEAP | stair | scrios

spléachadh | TRM | INTR | RÉIM | BAIL | SAIN | liosta »

TÉARMAÍ BÉARLA nua

assembly details *s pl*

TÉARMAÍ GAEILGE nua

sonraí *fir iol* **cóimeála**

2229336 » TÉARMA | stair | scrios

spléachadh | FCL | NOD | INFL | NÓT | liosta »

s	o	n	r	a	í		c	ó	i	m	e	á	l	a
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

NODA nua

sonraí *fir iol* cóimeála

Figure 4. Fiontar's customized terminology application in Léacslann.

The skills needed to build applications with Léacslann include Microsoft SQL Server, ASP.NET and C#, as Léacslann's back-end was built using these. Knowledge of XML technologies and web front-end technologies (such as HTML and JavaScript) is also necessary.

5. Availability

Everybody is welcome to use the self-service demo application shown above. However, it is important to remember that this is a demonstration only; the data you enter may be changed or deleted by us or other demo users at any time. Potential users wishing to use Léacslann for real-world purposes should contact us to discuss the options.

Notes

[1] *Fiontar,* Irish-language teaching and research school in Dublin City University, http://www.dcu.ie/fiontar.
[2] The name can be roughly translated from Irish as 'lex-centre' or 'lex-place'. The suffix *-lann* means 'place for' or 'centre for', compare *leabharlann* 'library' (from *leabhar* 'book'), *pictiúrlann* 'cinema' from *pictiúr* 'picture'.
[3] The Placenames Database of Ireland has not been migrated into Léacslann yet but it is a move we are considering.

References

A. Dictionary writing and terminology management software
TshwaneLex. 2003-2012. http://www.tshwanedje.com/tshwanelex/.
IDM. **2011.** http://www.idm.fr/products/dictionary_writing_system_dps/.
SDL Trados MultiTerm. **2008.** http://www.sdl.com/en/language-technology/products/terminology-management/.

B. Lexical databases
Irish National Terminology Database. **2006-2012.** http://www.focal.ie/.
Placenames Database of Ireland. **2008-2012.** http://www.logainm.ie/.