From DTD to relational dB. An automatic generation of a lexicographical station out off ISLE guidelines

Marta Villegas, Nuria Bel

Grup d'Investigació en Lingüística Computacional - Universitat de Barcelona (gilcUB)
Adolf Florensa s/n
08028 Barcelona, Spain
{tona,nuria}@gilcub.es

Abstract

This paper describes the Lexicographic Station Development Platform and how it has been used to implement the resulting lexicon guidelines and standards generated by ISLE Computational Lexicon Group in a prototype tool for lexical encoding. The aims of the work described here were to (i) exemplify and disseminate the Multilingual ISLE Lexical Entry (MILE) using an actual model and available monolingual data (ii) make extensive use of already existing PAROLE and SIMPLE lexicons and (iii) to eventually test the goodness of the guidelines by using a real scenario. To cope with these aims, the LSDP was designed as a tool generator which could automatically generate a prototype lexicographic station out of ISLE guidelines when formally expressed in a DTD. Thus, we have tested and exemplified the recommendations expressed in MILE but in addition we have also proved that MILE can be implemented on already existing monolingual resources.

1. Introduction

This paper describes: (i) the lexicographic station development platform used to automatically generate a prototype tool out off ISLE[1] guidelines which has been formally implemented in a DTD. (ii) An actual implementation of the ISLE guidelines expressed in MILE. And (iii) the use of the lexicographic station development platform for generating a prototype lexical tool for MILE/ISLE guidelines.

The aim of ISLE is to develop, disseminate and promote de facto HLT standards and guidelines for language resources, tools and products within an international framework, in the context of the EU-US International Research Cooperation initiative.

In the 'multilingual computational lexicon' area, ISLE has extended EAGLES[2] work on lexical semantics to design standards for multilingual lexicons. The central outcome of ISLE is the definition a general schema for multilingual lexical entry (MILE) which is to be the basis for a standard framework for multilingual computational lexicons. In addition, ISLE is to develop a prototype tool to assist the development of multilingual lexical resources following MILE schema.

The aim of this prototype tool is to (i) exemplify the MILE entry (ii) make extensive use of already existing monolingual resources and (iii) eventually test the guidelines in a real scenario.

Three aspects crucially determined the definition of the lexicographic station development platform we are describing here: (a) MILE is built as an additional layer on top of monolingual descriptions. In most cases, these monolingual layers are already existing resources which must be reused. The possibility to automatically generate a prototype tool out off already existing monolingual lexical resources seemed to be the right approach as this guarantees and facilitates the usability of already existing data and resources. (b) MILE is a general schema liable to be customized according to in-house needs in real scenarios. (c) Both, the definition of the prototype tool and the definition of the MILE itself, were parallel tasks. This meant that the prototype tool had to implement ongoing specifications which were not finished at that time.

This situation led us to define a Lexicographic Station Development Platform that guarantee the portability of the final prototype to the final specifications as well as to existing monolingual resources which will serve as the basic data for MILE. The lexicographical station development platform has been designed as a tool generator which parses any DTD describing an Entity Relationship model to (i) automatically map the DTD into a relational dB and (ii) build up a user-friendly interface able to cover the most common lexicographic requirements –such as means to automatically load - download the database from/into external SGML/XML files.

This article is organised as follows:

Section 2 describes the lexicographical station development platform. Section 3 contains the software specifications for the resulting prototype tool. Section 4 describes the implementation of the MILE (Multilingual ISLE Lexical Entry) module on the top of the prototype tool.

[1] See Atkins, S., Bel, N., Bertagna, F., Bouillon, P., Calzolari, N., Fellbaum, C., Grishman, R., Lenci, A., MacLeod, C., Palmer, M., Thurmair, G,. Villegas, M., and Zampolli, A. From resources to Applications. Designing the Multilingual ISLE Lexical Entry, in these proceedings for further information.

[2] Expert Advisory Group in Lexical Standards. See references for further information.

2. Lexicographic station development platform

The Lexicographical Station Development Platform is a prototype tool generator that reads and parses a DTD and generates a relational data base and a core dB web interface.

Our lexicographical station development platform guarantees that already existing monolingual resources expressed in SGML/XML can be easily reused by and ported to MILE.

Basically, the lexicographic station development platform includes a generation module, a customisation module and a core web interface module, which can be briefly described as follows.

The generation module automatically generates a relational dB out off a DTD. The project benefits from the fact that a conceptual model expressed in terms of Entity-Relationship model can be easily mapped into a relational dB.

The customisation module allows to modify certain aspects of the dB at the time that overcomes some of the well known shortcomings of DTD' s such as typed references and type declaration.

The core web interface module consists of a series of scripts that allow to manage the dB with a friendly interface. Although user requirements differ from site to site according to in-house needs, the tool comes equipped with a set of basic functionalities. Our experience in past lexicographic projects led us to define an accurate list of requirements which include (i) query and browsing facilities, (ii) import, export and migration of data, (iii) easy encoding of new data, (iv) test and validation of both the data and the model, (v) customisation facilities, and (vi) lexicographic functionalities such as type definition, class extraction and statistical facilities.

As in the case of the generation module, the web interface module acts on the model expressed in the DTD in order to make the necessary calculations to access, manipulate and display data from relevant tables. The web interface module, therefore, only needs a DTD and its corresponding dB to be able to work.

Figure 1 reflects the general architecture underlying the lexicographical station development platform.

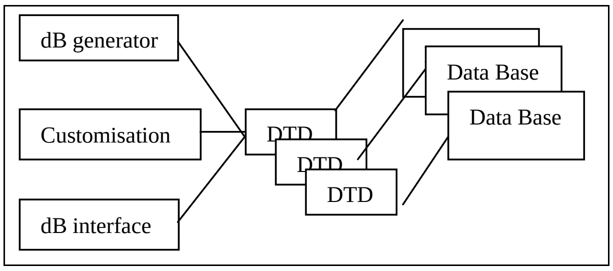


Figure 1. General view of the lexicographical station development platform

2.1. The generation module

The generation of the data base is done by means of a *perl* script that, making extensive use of the perlSGML module, reads the DTD and generates three output scripts:

- CreateDB: is an output file containing the relevant 'CREATE TABLE' instructions. This output file can be edited to make the desired modifications (shorten or length the fields, delete tables,...) and can be executed by MySQL by typing ' mysql> *data_base_name < script.file*'

- TabularDTD: is a *perl* script that reads an SGML data file and distributes the data it contains into a series of tabular files which exactly correspond to the tables in the dB. TabularDTD is sensitive to the hierarchic relations between SGML elements and keeps track of the foreign keys involved in each content element (see section 3.2 for further details).

- LoadDB: is an output file containing the relevant 'INSERT' statements and is responsible of loading the tabular files in the corresponding tables.

2.1.1. Tables definition

BuilDB reads the DTD looking for all elements and classifies them into ***main*** or ***content*** elements. ***Main elements*** are top elements having an ID-type attribute. For each main element, *BuildDB* creates a corresponding ***main table***. Two additional types of tables can be also created. These are ***content tables*** and ***list tables***. Content tables are created whenever an element has a content element. List tables are created whenever an element includes an IDREFS-typed attribute (that is, an attribute valued as a list of IDs).

```
<!ELEMENT Element - - ContentElement>
<!ATTLIST Element
            attribute1   ID        #REQUIRED
            attribute2   IDREFS    ...
            ...
```

```
CREATE TABLE Element (...
CREATE TABLE Element_ContentElement (...
CREATE TABLE Element_List_attribute2 (...
```

Figure 2: Tables definitions

The name of the tables derive from the name of the elements, thus, main tables have the same name as the corresponding main object. *Content* tables names result of the concatenation of the parent and the content element and, finally, *list* table names result from the concatenation of the element and the IDREFS-type attribute name. This can be see in Figure 2 above.

2.1.2. Fields definition

Attributes in the element' s ATTLIST description of elements are directly mapped into fields in the corresponding table definition according to the following criteria: ID and IDREF(s)-typed attributes translate as VARCHAR, NUMBER attributes translate as INT, CDATA and NUMBERS attributes translate as varchar.

Content tables include two additional fields: one corresponds to the table ID and is defined as an auto increment primary key; the other serves to relate the content element with the relevant parent element and acts as foreign key

List tables serve to encode list-typed attributes. They include two fields which are defined as primary keys. One is defined as 'id_parent' and serves to indicate the element containing the list-typed attribute. The other is defined as 'id_attibute' and serves to indicate the attribute itself.

In the following figures we exemplify the mapping of a given element *Element* as described in Figure 3. Thus, in figure 4, we can see attributes mapping into table's fields. Figure 5 describes the mapping of a *content* element. Finally, figure 6 exemplifies the mapping of an IDREFS-typed attribute into a *list* table.

```
<!ELEMENT Element –0- ContentElement>
<!ATTLIST
id              ID              #REQUIRED
attData         CDATA           #IMPLIED
attEnum         (A|B)           A
attIdref        IDREF           #IMPLIED
attIdrefs       IDREFS          #IMPLIED>
```

Figure 3. DTD description for Element

Field	Type	Null	Key	Default	Extra
Id	Varchar		PRI		
AttData	Varchar	YES		NULL	
AttEnum	Enum(A|B)			A	
AttIdRef	Varchar	YES	MUL	NULL	

Figure 4. Main table definition for Element

Field	Type	Null	Key	Default	Extra
Id	Varchar		PRI	0	Auto increment
Id-parent	Varchar		MUL		
….	…	…	…	…	…
…	…	…	…	…	…

Figure 5. Content table definition for ContentElement

Field	Type	Null	Key	Default	Extra
Id	Varchar		PRI		
Id-parent	Varchar		PRI		

Figure 6. List table definition for AttIrefs attribute

2.2. Customisation module

In order to overcome some of the well known shortcomings of DTD' s (typed references, type declaration, inheritance...) the prototype includes a customisation module.

This customisation module serves a double purpose. On the one hand, it allows to express type constraints which cannot be expressed in SGML DTD' s. On the other hand, it becomes crucial to define the 'domain' of a given element. Relations among elements can be established as 'vertical' or 'horizontal' relations. Vertical relations are the standard hierarchical relations between an element and its content elements. Horizontal relations are those established by IDREF or IDREFS typed attributes which serve to relate a given element with any other element of the model. Both, vertical and horizontal relations between elements define the domain or scope of an element. In the following example we describe the domain an imaginary element *Element* 2 containing one IDREF attribute typed as element 5. In this example, the domain for our imaginary Element 2 includes all nodes dominated by Element 2 plus the domain of the Element 5.

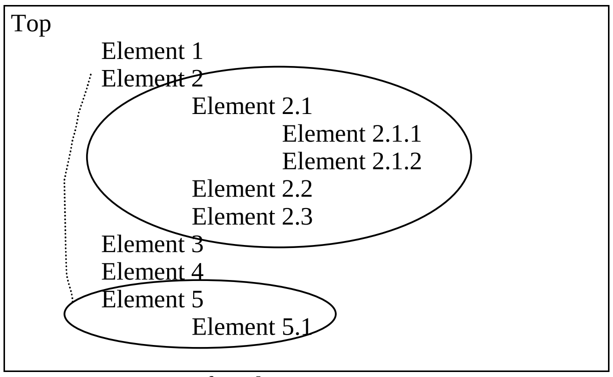


Figure 6 Scope for Element 2

This domain is needed to provide a better functionality to the system. As we will see in next section, the prototype tool comes equipped with some basic functionalities. These functionalities are better tuned if type references are explicitly established in this customisation module.

2.3. Core dB interface

Besides tables definition described in previous section, the system provides with a user interface able to manage the dB in a friendly and explanatory fashion.

The aim of the lexicographical station development platform is to provide the resulting prototype tool with a minimum set of build in functionalities that cover the most common lexicographic requirements. In addition, the explanatory and dissemination purpose of ISLE project, lead us to include a number of functionalities which serve to know and understand the resulting prototype tool and the model this is managing.

All this is to be achieved without facing lexicographers with the technicalities of a dB. Lexicographers are only required to know the model expressed in the DTD and, therefore, they deal with the elements defined in the DTD. It is the system which makes the necessary calculations to access and manipulate data from the relevant tables.

The prototype tool, therefore, is designed as DTD dependant rather than dB dependant and includes a good number of scripts that, taking the DTD structure as input, make the necessary calculations to act on the relevant tables in the dB. Essentially these facilities include (i) loading and downloading data from and into SGML files, (ii) making forms to manage the dB, (iii) browsing data and (iv) learning about the model.

download data in SGML/XML fashion. The user is given a tree representation of the DTD and selects one element. The system, then, makes the necessary calculations to extract data in SGML/SML format for the desired element.

define forms to extract or load data. The system allows to define online forms to manage the data base. The first step in this process is to define the domain of the form. Here is where the customisation process explained above becomes crucial. The user selects the top most element he wants to include in the form. The system calculates the domain of the selected element by taking into account the horizontal and vertical relations it participates in. Once this is done, the system displays a form with the relevant fields. Fields in the form are defined following attribute's definition in the DTD. Thus, CDATA attributes translate into text fields, ENUM attributes translate into pop-up fields, customised IDREF attributes translate into pop-up menus, and IDREFS attributes translate into multi valued scrolling list fields. Once the user has filled in the form, the system makes the necessary calculations to build up the relevant SQL query.

Browsing the data and the model. The tool contains a good number of facilities to browse both the data and the model and its mapping into the database. The prototype allows the user to see the data in a DTD fashion and benefits from the fact that it knows the relational component of the database since this is formally expressed in the DTD.

Section 5 of this paper includes sample screens of the functionalities described here.

The overall system can be described as in Figure 7.

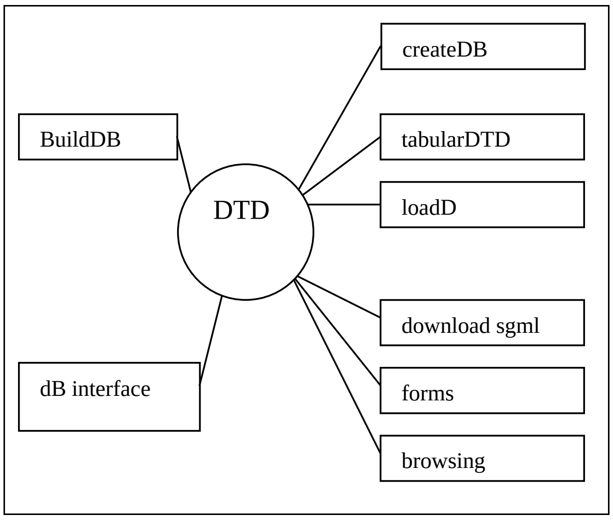


Figure 7. Functionalities

3. Software Specifications

The prototype is implemented using well supported open source resources which can be easily portable. Essentially these include MySQL database server and Apache server:

Database server: 3.23.16-alpha version of MySQL which can be downloaded from www.mysql.com.

Perl support for MySQL: Perl support for MySQL is provided by means of the 'DBI/DBD' client interface. The Perl 'DBI/DBD' client code requires Perl5.004 or later. Perl DBI/DBD modules can be downloaded from www.symbolstone.org/technology/perl/DBI/index.html or www.perl.com/CPAN/modules/by-module/DBIx/i. among others. You should get the Data-Dumper, DBI and Msql-Mysql module distributions and install them in that order.

Web Server: Version 1.3 of the Apache web server which can be downloaded from www.apache.org/httpd.

Perl support for the Apache server: modperl is the Apache/Perl integration project. Modperl can be downloaded from a CPAN site under modules/by-module/Apache

4. MILE module

ISLE defines the multilingual layer as an additional layer on top of the monolingual ones. Thus, whereas monolingual layers collect morphological, syntactic and semantic information needed to describe monolingual lexicons, the multilingual layer defines correspondence objects which describe relations between monolingual representations. This approach guarantees the independence of monolingual descriptions at the time that allows the maximum degree of flexibility and consistency in reusing existing monolingual resources to build new bilingual lexicons.

Bilingual correspondences between source and target unit elements can be rather complex and may involve different transfer conditions. In these cases, the bilingual layer allows to establish *tests* and/or *actions* upon monolingual descriptions in source and target lexicons respectively. Tests and actions are constraints or enrichments on monolingual descriptions needed only when moving from one language to another. More exactly: tests specify a condition in source language under which a given translation is valid; and, actions specify a condition in the target language under which a given translation is valid.

These transfer conditions include semantic transfer conditions and syntactic transfer conditions which can be briefly summarized as follows:

Semantic transfer conditions:

- ☐ Argument correspondences between source and target predicates.
- ☐ Addition of semantic feature(s) to source or target SemUs.
- ☐ Addition of semantic feature(s) to an argument of source or target predicate.

Syntactic transfer conditions:

- ☐ Constrain the head of the syntactic description by adding syntactic or semantic features.
- ☐ Link source and target positions (i.e. syntactic arguments)
- ☐ Adding a syntactic position to source or target syntactic descriptions.

- ☐ Changing the optionality status of a given syntactic position.
- ☐ Prohibit the realization of a given syntagma in a given syntactic position.
- ☐ Adding semantic or syntactic features to syntagmas filling a given syntactic position.
- ☐ Lexicalizing the syntagma filling a given syntactic position.

The kind of tests and actions involved in each correspondence depends on the words involved and on the kind of information included in both source and target lexicons. More crucially, the set of transfer conditions involved in a given bilingual correspondence acts on descriptive elements which, in most cases, vary from unit to unit.

This scenario makes it impossible to define a static fixed form (or template) for encoding bilingual correspondences. Notice that the number and kind of transfer conditions, and the number and kind of objects these transfer conditions apply on will change from correspondence to correspondence depending on the kind of monolingual descriptions we are trying to link.

The complex nature of bilingual correspondences led us to define the MILE module as an object and the list of admissible transfer conditions as a set of methods that further enrich the initial MILE object in order to collect the desired information.

Formally, the ISLE MILE object has been implemented following the bilingual correspondence elements defined in the Genelex bilingual DTD. This allows us to (i) exemplify the MILE entry in an actual model (ii) to make extensive use of already existing PAROLE and SIMPLE lexicons which, in fact, are instances of the EAGLES / Genelex monolingual model (iii) to eventually test the model in a real scenario and (iv) to include MILE component on the top of the prototype tool.

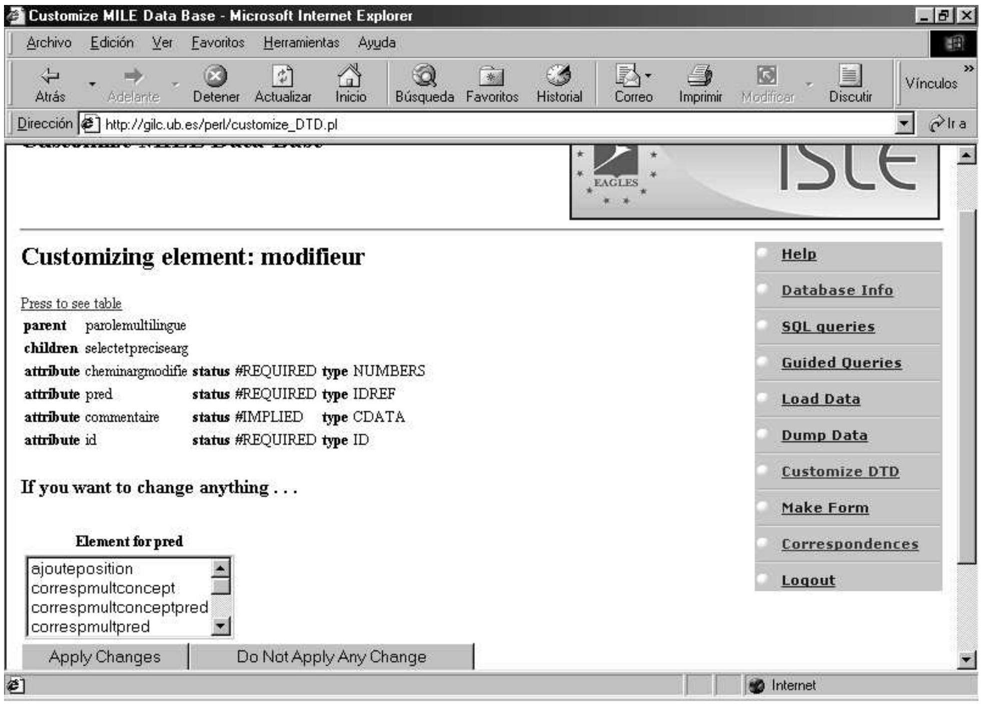
Essentially, the methods that our implementation of MILE includes are those listed above. Each method takes as input a relevant monolingual descriptive element and the constraints or enrichments (i.e., its transfer conditions) the user wants to apply on them. A simplified version of some syntactic transfer conditions is listed below:

```
add_sem_feature_to_head(head, list_of_features)
add_synt_feature_to_head(head, list_of_features)
add_sem_feature_to_position(position, features)
add_synt_feature_to_position(position, features)
lexicalize_position(position, lexical_unit)
change_position_status(position, optional_status)
add_synt_position(synt_descripton, position)
```

In order to lead the user through the whole process of encoding complex transfer conditions, the system first parses monolingual lexicons and collects the morphological, syntactic and semantic descriptions of the words to be linked. All this data is displayed in SGML/XML format with browsing facilities. This allows the user to have an exact idea of monolingual descriptions and to select the relevant elements of the initial MILE object (that is, relevant semantic units and relevant syntactic descriptions in case transfer conditions are needed). Once the basic MILE object is constructed, the system parses the data it contains looking for the monolingual descriptive elements it includes in order to list the relevant methods that can be applied and the elements these methods apply on (that is, the first argument in the examples listed above). The user selects the methods and fills in the data required (i.e., the second argument in the examples listed above). Finally, the system builds up the complex MILE object by applying the set of selected methods.

All this process is possible because the system knows the relations established among elements in the DTD. The Genelex Bilingual DTD results from the adding of two monolingual lexicons plus a bilingual layer that includes correspondence elements which, essentially, are in charge of relating monolingual elements. The type declaration expressed in the customisation process explained above allows the system to know the horizontal relations that hold between bilingual correspondence elements and monolingual descriptions. The way Genelex bilingual layer is defined together with type declaration process opens the possibility to include different monolingual descriptions in the MILE implementation. In other words, MILE object reflects MILE schema and object methods reflect MILE transfer conditions. Object methods require monolingual information in order to provide with the final MILE complex object (that is, morphosyntactic information of source and target words, the syntactic environments these units occur in, etc.). In our case, this monolingual information is expressed in the PAROLE SIMPLE lexicons following the Genelex DTD but it could be expressed otherwise if using different monolingual resources.

5. Examples


Customize MILE Data Base - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Adelante Detener Actualizar Inicio Búsqueda Favoritos Historial Correo Imprimir Modificar Discutir Vínculos
Dirección http://gilc.ub.es/perl/customize_DTD.pl
Ir a
EAGLES
ISLE
Help
Database Info
SQL queries
Guided Queries
Load Data
Dump Data
Customize DTD
Make Form
Correspondences
Logout
Apply Changes
Do Not Apply Any Change
Internet

Customizing element: modifieur

Press to see table

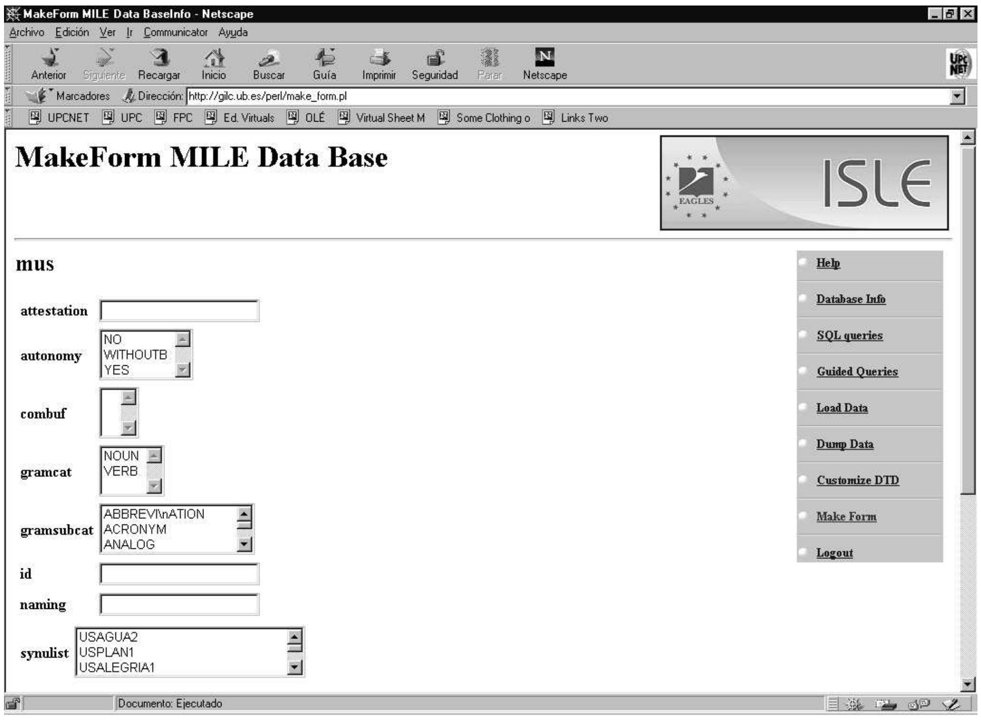
parent	parolemultilingue				
children	selectetpreciseearg				
attribute	cheminargmodifie	**status**	#REQUIRED	**type**	NUMBERS
attribute	pred	**status**	#REQUIRED	**type**	IDREF
attribute	commentaire	**status**	#IMPLIED	**type**	CDATA
attribute	id	**status**	#REQUIRED	**type**	ID

If you want to change anything . . .

Element for pred

ajouteposition
correspmultconcept
correspmultconceptpred
correspmultpred

Table 2: Customisation procedure

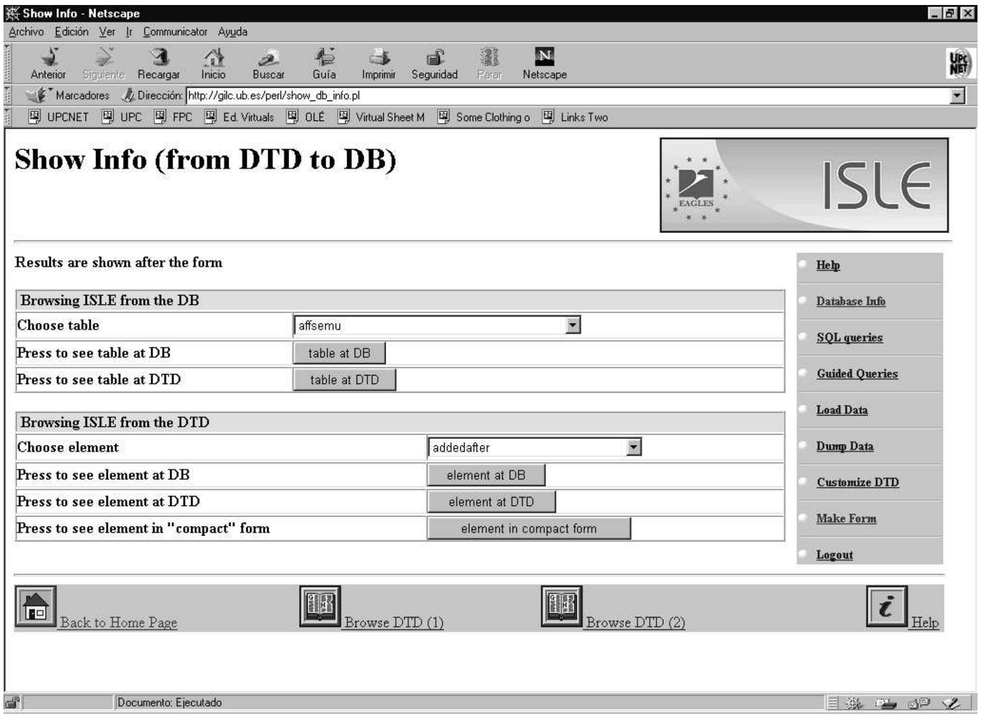

MakeForm MILE Data BaseInfo - Netscape
Archivo Edición Ver Ir Communicator Ayuda
Anterior Siguiente Recargar Inicio Buscar Guía Imprimir Seguridad Parar Netscape
Marcadores Dirección: http://gilc.ub.es/perl/make_form.pl
UPCNET UPC FPC Ed. Virtuals OLÉ Virtual Sheet M Some Clothing o Links Two
EAGLES
ISLE
Help
Database Info
SQL queries
Guided Queries
Load Data
Dump Data
Customize DTD
Make Form
Logout
Documento: Ejecutado

MakeForm MILE Data Base

mus

attestation	
autonomy	NO WITHOUTB YES
combuf	
gramcat	NOUN VERB
gramsubcat	ABBREVI\nATION ACRONYM ANALOG
id	
naming	
synulist	USAGUA2 USPLAN1 USALEGRIA1

Table 2: Making a form


Show Info - Netscape
Archivo Edición Ver Ir Communicator Ayuda
Anterior Siguiente Recargar Inicio Buscar Guía Imprimir Seguridad Parar Netscape
Marcadores Dirección: http://gilc.ub.es/perl/show_db_info.pl
UPCNET UPC FPC Ed. Virtuals OLÉ Virtual Sheet M Some Clothing o Links Two
ISLE
EAGLES
Help
Database Info
SQL queries
Guided Queries
Load Data
Dump Data
Customize DTD
Make Form
Logout
Back to Home Page
Browse DTD (1)
Browse DTD (2)
Help
Documento: Ejecutado

Show Info (from DTD to DB)

Results are shown after the form

Browsing ISLE from the DB	
Choose table	affsemu
Press to see table at DB	table at DB
Press to see table at DTD	table at DTD

Browsing ISLE from the DTD	
Choose element	addedafter
Press to see element at DB	element at DB
Press to see element at DTD	element at DTD
Press to see element in "compact" form	element in compact form

Table 2: Browsing the model

6. References

Bel, N., Busa, F., Calzolari, N., Ogonowski, A., Peters, W., Ruimy, N., Villegas, M., Zampolli, A. 2000. SIMPLE: a general Framework for the Development of Multilingual Lexicons. In *LREC Proceedings*, Athens.

Busa, F., Calzolari, N., lenci, A., and Pustejovsky, J. 1999, Building a Semantic lexicon: structuring and Generating Concepts. In *The Third International Workshop on Computational Semantics*, Tilburg, The Netherlands.

Calzolari, N. 1991. *Acquiring and Representing Information in a Lexical Knowledge Base*, ILC-CNR, Pisa ESPRIT BRA-3030/ACQUILEX – WP No. 16.

Calzolari, Nicoletta, Lenci, Alessandro, Zampolli, Antonio, Bel, Nuria, Villegas, Marta, Thurmaier, Gregor., 2001. The ISLE in the Ocean Transatlantic Standards for Multilingual Lexicons (with an Eye to Machine Translation). In *Proceedings of MTSummiy VIII*, Santiago de Compostela, Spain.

Calzolari, N., Mc Naught, J., Zampolli, A. 1996. EAGLES Final Report. Pisa.

Fellbaum, C. 8ed.). 1998. *Wordnet: An Electronic Lexical database*, Cambridge, MA: The MIT Press.

Fontenelle, T. 1997. *Turning a bilingual dictionary into a lexical-semantic database.* Tübingen: Max Niemeyer, (Lexicographica. Series maior; 79).

GENELEX Consortium. 1994. Report on the Semantic Layer, Project EUREKA GENELEX, Version 2.2.

Underwood, N., Navarretta, C. 1997. A Draft Manual for the Validation of Lexica. *Final ELRA Report*, Copenhagen.