A WEB BASED SYSTEM TO QUERY XML DICTIONARIES
IGNACIO PARRA
COMPUTING ERASMUS
2003/2004

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

13th May 2004
Summary

The aim of this project is to construct a web based tool for the Centre for Translation Studies of the University of Leeds in which the students of this Centre can upload and query the dictionaries they construct for their projects as part of the modules of the Centre.

The dictionaries are XML documents that follow the recommendation of the Text Encode Initiative Consortium (TEI-C) for the encode of electronic documents.

The students will be able to upload, validate, view and query their own dictionaries in a friendly web site.
Acknowledgments

I would like to thank first of all Katja Market and Serge Sharoff, my supervisors for the extensive advice and help that they have provided me with. Without them, this project would be never have been possible.

I thank Brandon Bennet my assessor for the comments and improvements that he gave in the mid project report and in the progress meeting.

I would also like to thank the support staff for providing me with the resources I required and answering all the emails concerning little problems I had with the server.

Finally I wish to extend my thanks to my family for making it possible for me to be here this year and to my ’family in Leeds’ for cheering me up during the hard times.

My friends have supported me while I was working on this project, and have endured my monologues about it. This is priceless.

Many thanks to all those whithout whom this project would not have been possible.
# Table of Contents

## 1 Introduction

1.1 Project introduction ........................................ 1
1.2 Why I have chosen this project ............................... 1
1.3 Project aim .................................................. 2
1.4 Objectives, Minimum requirements and Deliverables ........ 2
1.4.1 Objectives ............................................... 2
1.4.2 Minimum requirements ................................. 2
1.4.3 Deliverables ............................................. 2
1.4.4 Possible further enhancements .......................... 2
1.4.5 Project schedule ......................................... 3

## 2 Background Research

2.1 Dictionaries ................................................ 3
2.1.1 Computers and dictionaries ............................... 5
2.1.2 Corpus .................................................. 5
2.2 Text Encoding Initiative .................................... 6
2.2.1 Purpose ............................................... 6
2.2.2 TEI lite ............................................... 7
2.2.3 Encoding dictionaries following TEI guidelines .......... 7
2.3 XML ....................................................... 8
2.4 XSLT and XPath ............................................ 9
2.5 Usability ................................................... 10
2.6 Questionnaires ............................................. 12
2.6.1 Why ................................................... 13
2.6.2 How ................................................... 13

## 3 Requirements analysis and definition

3.1 Problem owner ............................................. 16
3.2 System users .............................................. 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Project specification</td>
<td>16</td>
</tr>
<tr>
<td>3.3.1</td>
<td>The tool pages</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Choice of development tools</td>
<td>18</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Berkeley DB XML</td>
<td>18</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Perl</td>
<td>19</td>
</tr>
<tr>
<td>3.4.3</td>
<td>CGI.pm</td>
<td>19</td>
</tr>
<tr>
<td>3.4.4</td>
<td>LibXML and LibXSLT</td>
<td>20</td>
</tr>
<tr>
<td>3.5</td>
<td>Methodology</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>System and software design</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Implementation</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Tags used to query a dictionary</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>System testing and evaluation</td>
<td>34</td>
</tr>
<tr>
<td>6.1</td>
<td>System testing</td>
<td>34</td>
</tr>
<tr>
<td>6.2</td>
<td>System evaluation</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>Personal Reflection</td>
<td>38</td>
</tr>
<tr>
<td>A</td>
<td>Encoding a dictionary in XML</td>
<td>40</td>
</tr>
<tr>
<td>A.1</td>
<td>Header</td>
<td>41</td>
</tr>
<tr>
<td>A.2</td>
<td>Text structure</td>
<td>43</td>
</tr>
<tr>
<td>A.3</td>
<td>Entries</td>
<td>44</td>
</tr>
<tr>
<td>A.4</td>
<td>Information on written and spoken forms</td>
<td>45</td>
</tr>
<tr>
<td>A.5</td>
<td>Grammatical information</td>
<td>46</td>
</tr>
<tr>
<td>A.6</td>
<td>Sense information</td>
<td>47</td>
</tr>
<tr>
<td>A.7</td>
<td>Etymological information</td>
<td>47</td>
</tr>
<tr>
<td>A.8</td>
<td>Examples</td>
<td>48</td>
</tr>
<tr>
<td>A.9</td>
<td>Usage information and other labels</td>
<td>48</td>
</tr>
<tr>
<td>B</td>
<td>Dictionary example</td>
<td>51</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Project introduction

In recent years, a new technology has come to revolutionize the way in which programs relate to each other across a network. This new technology is called Extensible Markup Language (XML).

This new “technology” is now used in many other situations, not only for the original idea of facilitating communication. It is also used to store documents in a more readable way than the plain text or the HTML format (you can use as many tags as you want to describe something), to facilitate the web maintenance and creation, to make it easier to publish the same information in different ways, etc.

Before XML existed, everyone created electronic texts as they wanted. Some of them used the plain text, some others used some specific word processor, and many other different types. But some scholarly societies founded a new organization that had to create a standard “to help libraries, museums, publishers, and individual scholars to represent all kinds of literary and linguistic texts for online research and teaching, using an encoding scheme that is maximally expressive and minimally obsolescent.”[1]

This organization is called Text Encoding Initiative Consortium (TEI-C). It was created in 1987 and since then has published 4 major different versions of this guidelines to encode electronic texts.

The guidelines published by the TEI are the guidelines that the students of the Centre for Translation Studies at the University of Leeds are using to encode electronic dictionaries. These dictionaries are the ones that the people can view and query on my final year project website.

1.2 Why I have chosen this project

I have chosen this project for two reasons:

My interest in XML technologies Since I have about XML two years ago, I believe that this “language” is going to revolutionize the way the Internet and the programs works together. I have been working on it since that day, and discovering new things, opening a new world before my eyes.

Never used cgi’s I think that the subject of the final year project must be something in that you are interested in, but you have never worked with so you can learn new things.
1.3 Project aim

The aim of the project is to create a web-based system to query XML dictionaries that the Centre for Translation Studies MA students can use for querying the result of one of their module projects.

1.4 Objectives, Minimum requirements and Deliverables

1.4.1 Objectives

The objective of the project is to create a web site in which the students of the Center for Translation Studies can upload the dictionaries that they have made for their projects and query them.

1.4.2 Minimum requirements

1. The tool has to upload and validate the dictionaries of the student in a XML format

2. Design the full website for the tool

3. Basic queries to the XML document

4. Full and partial display of the dictionary

5. Usability evaluation via user feedback

1.4.3 Deliverables

The deliverables of this project are:

- The tool
- Website
- Project specification (included in the Final year report)

1.4.4 Possible further enhancements

- Advanced queries: The idea of this enhancement is to give the users the possibility to make more difficult queries on the dictionary. That is to say, they can ask for an entry within the dictionary, at the same time that they can ask for some specific words inside examples, or they can look for some usage information.
• Two way queries: When they use a bilingual dictionary, and they search for one entry, the system will look for the word in the entries and inside the examples or the translations of all the entries.

• Dictionary edit: If the system finds an error while validating the dictionary, or when the user wants to add or change something that is in the dictionary that is currently loaded, the system will provide the user with an editor in which he will change whatever he wants and he will be able to save that document to his computer and continue the validation process with the new document.

1.4.5 Project schedule

This is the project schedule. It has been modified since the mid project report because I was very optimistic about the dates and the tasks that I will accomplish along the second term.

2 Background Research

This chapter summarizes the main research done to gain a basic understanding of the issues around this project and how to implement the tool.

2.1 Dictionaries

Because the aim of the project is to create a tool for displaying and querying dictionaries, my research first centered on dictionaries. According to [2], a dictionary is defined as follows:

dictionary
noun
1 a book that contains a list of words in alphabetical order with their meanings explained or written in another language, or a similar product for use on a computer:
a French-English/English-French dictionary
a bilingual/monolingual dictionary

2 a book which gives information about a particular subject, in
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Preparar las tareas</td>
<td>16/10</td>
<td>18/10</td>
<td>15 días</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Lectura de fondo</td>
<td>16/10</td>
<td>18/10</td>
<td>8 días</td>
<td>20/10</td>
<td>7 días</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Especificación del proyecto</td>
<td>20/10</td>
<td>20/10</td>
<td>7 días</td>
<td>06/11</td>
<td>6 días</td>
<td>16/11</td>
<td>3 días</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Primera iteración</td>
<td>06/11</td>
<td>10/11</td>
<td>4 días</td>
<td>18/11</td>
<td>21 días</td>
<td>14/11</td>
<td>5 días</td>
<td>14/11</td>
</tr>
<tr>
<td>5</td>
<td>Diseño del sitio de inicio</td>
<td>21/11</td>
<td>01/12</td>
<td>6 días</td>
<td>01/12</td>
<td>10 días</td>
<td>19/01</td>
<td>29 días</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Prototipo básico de la herramienta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Segunda iteración</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Especificación 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Prototipo básico de la herramienta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Elaboración del Informe del Taller</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Tercera iteración</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Especificación 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Diseño de la interfaz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Prototipo básico de la herramienta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Cuarta iteración</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Especificación 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Diseño de la encuesta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Prototipo básico de la herramienta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Finalización</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Finalización del proyecto</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Revisión de la encuesta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Encuesta de usabilidad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Finalización del sitio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Finalización del sitio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Presentación del informe</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
which the entries are given in alphabetical order:

- a biographical/science dictionary
- a dictionary of quotations

Dictionaries can be classified by many criteria. The following examples are taken from [3].

- The number of languages
- The age of users
- The size of the dictionary
- The scope of coverage by subject
- ...

A more discriminating typology can be found in [4].

### 2.1.1 Computers and dictionaries

The appearance of computers in the dictionary industry has changed the development of dictionaries. For example, it is easier for a machine than for a human to get a list of 25,000 words and order them alphabetically. Also, abridging a dictionary is faster if you have it stored on a computer and tagged because the only thing you need to specify are the tags that you need to show.

But not everything is so beautiful. The coding of a dictionary is very hard work, because you have to link every definition to a set of identifiers like source, subject, usage information, etc. For example, when you prepare a medical dictionary, you have to input the lexical unit you want to add (anterior arches of atlas), the singular of the unit if it is plural (anterior arch of atlas), the word under which the term should be listed (anterior arches of atlas has to go under arch, not under anterior), etc. The example was taken from [3].

### 2.1.2 Corpus

However, the most important help that the computer has brought to dictionary development is the corpus. A corpus is the collection of a single writer’s work or of writing about a particular subject, or a large amount of written and sometimes spoken material collected to show the state of a language[2]. This data has always been available, but without the
computers and their fast ordering and searching tools, systematic use of the corpus was impossible.

With a corpus, the creators of dictionaries do not have to rely on their own colleagues reading and listening for a perspective of their languages. New words and meanings are discovered when the dictionary editor is searching the corpus. Because a dictionary has to reflect general language use, with we have attained richer dictionaries with the corpus.

2.2 Text Encoding Initiative

The Text Encoding Initiative (TEI) is a consortium that publishes guidelines for encoding electronic texts. The most recent version of the guidelines is Version four, which was developed and made public in 2001.

The guidelines provide a means of representing those features of a text which need to be identified explicitly in order to facilitate processing of the text by computer programs. They define a set of tags that have to be inserted in the text.

2.2.1 Purpose

The Text Encoding Initiative is an international consortium created in 1987 by some scholarly societies helped economically by some major institutions with the objective to create a “standard that helps libraries, museums, publishers, and individual scholars to represent all kinds of literary and linguistic texts for online research and teaching, using an encoding scheme that is maximally expressive and minimally obsolescent.”

They have created a community-based standard for encoding and interchange of texts that everyone can use for his own purpose. The goals of the consortium are (taken from [7]):

1. To establish and maintain a home for the Text Encoding Initiative (TEI) in the form of a permanent organizational structure.

2. To ensure the continued funding of TEI-C activities, for example: editorial maintenance and development of the TEI guidelines and DTD, training and outreach activities, and services to members.

3. To create and maintain a governance structure for the TEI-C with broad representation of TEI user-communities.
The fundamental principles of the consortium are this (taken from [7]):

1. The TEI guidelines, other documentation, and DTD should be free to users.
2. Participation in TEI-C activities should be open (even to non-members) at all levels.
3. The TEI-C should be internationally and interdisciplinary representative.

2.2.2 TEI lite

The Text Encoding Initiative Guidelines are addressed to anyone who wants to interchange information stored in an electronic form. They emphasize the interchange of textual information, but other forms of information such as images and sound are also addressed [8]. Because of these wide range of applications, the TEI Consortium has defined a more manageable selection of all the SGML tags of the TEI guidelines and it gives it the name of the TEI Lite Guidelines.

These TEI Lite Guidelines try to be a starter set, including all the basic elements that everybody needs to know. With them, you cannot achieve the same result as with the full TEI Guidelines, but for most applications these Lite guidelines are more than sufficient.

2.2.3 Encoding dictionaries following TEI guidelines

Both typographically and structurally, dictionaries are extremely complex. In addition, dictionaries interest many communities with different and sometimes conflicting goals. As a result, many general problems of text encoding are particularly pronounced here, and more compromises and alternatives within the encoding scheme may be required [6].

Because the structure of dictionary entries varies widely the simplest way for an encoding scheme to accommodate the entire range of structures is to allow any element to appear anywhere in a dictionary entry.

Basically, dictionaries have the same structure of front matter, body, and back matter than more general texts. The tag set for front matter and back matter is used here and in the other specific tags sets. Apart from this, it has a big number of specific tags relating to the body of the dictionary. A simple dictionary entry may contain information about the form of the word treated, its grammatical characterization, its definition, synonyms, or translation equivalents, its etymology, cross-references to other entries, usage information, and examples. In addition, dictionary entries often have a complex hierarchical structure.
For example, an entry may consist of two or more sub-parts, each corresponding to information for a different part-of-speech homograph of the headword. The entry (or part-of-speech homographs, if the entry is split this way) may also consist of senses, each of which may in turn be composed of two or more sub-senses, etc. Each sub-part, homograph entry, sense, or sub-sense we call a level; at any level in an entry, any or all of the constituent parts of dictionary entries may appear[6].

For an example of an electronic dictionary encoded following the TEI Guidelines, please refer to appendix B.

2.3 XML

The Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. It is a group of rules that can be used to create new markup languages which can be utilized in a wide range of applications.

It is a standard, simple, self-describing way of encoding both text and data so that content can be processed with relatively little human intervention and exchanged across diverse hardware, operating systems, and applications.

The rules, the recommendations and the tags defined in the TEI guidelines enable the creation of documents that conform to XML.

The power of XML is based on the following three characteristics:

**Its emphasis on descriptive rather than procedural markup** In a descriptive markup system, the markup codes used do little more than categorize parts of a document. In a procedural markup system you will have to describe how these parts of the document have to be displayed. This separation of display from content encoding allows the same information to be displayed in many different ways by only changing the display layer, without touching the data layer.

**Its document type concept** Because we define a document like a tree of tags the computer can validate the document.

**Portability** A basic design goal of XML is to ensure that documents encoded according to its provisions can move from one hardware and software environment to another without loss of information.
The design goals for XML are (Obtained from [9]):

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

2.4 XSLT and XPath

XSL is a family of recommendations for defining XML document transformation and presentation. It consists of three parts:

- **XSL Transformations (XSLT)** a language for transforming XML
- **XML Path Language (XPath)** an expression language used by XSLT to access or refer to parts of an XML document. (XPath is also used by the XML Linking specification)
- **XSL Formatting Objects (XSL-FO)** a XML vocabulary for specifying formatting semantics

XSLT and XPath provide a powerful implementation of a tree-oriented transformation language for transmuting instances of XML using one vocabulary into either simple text, the legacy HTML vocabulary, or XML instances using any other vocabulary imaginable.[10]
XSLT is the language we use to model how a XML document has to appear to the user. Before XSLT was invented, CSS was used to control how an XML-file is viewed. CSS is a very easy way to define this information, but it has one problem: it was not an XML-type file, and there was a need for an XML based Stylesheet Language. Basically, XSLT is a language for transforming XML documents into other XML documents and XPath is a language for defining parts of an XML document.

“In the transformation process, XSLT uses XPath to define parts of the source document that match one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document. The parts of the source document that do not match a template will end up unmodified in the result document.”[11]

XPath is not only the language used by XSLT to select one or more predefined templates. It is also a full querying language like SQL. With it, the user can make all the queries that he wants to a XML document. But unlike SQL, XPath differs more from standard English. Its grammar is more difficult to understand, but is as powerful as SQL.

“The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within Uri’s and XML attribute values. In addition to its use for addressing, XPath is also designed so that it has a natural subset that can be used for matching (testing whether or not a node matches a pattern); this use of XPath is described in XSLT.”[12]

2.5 Usability

One of the forgotten concepts that makes the websites comfortable to stay in is the usability, an idea plenty of times forgotten by the websites designers. It is very important to have an usable website because the people will be more comfortable when it is using it.

One study conducted by Nielsen[13] gives us the ten main usability principles:

1. **Visibility of system status** Always keep the users informed about what is going on, through providing appropriate feedback within reasonable time.

2. **Match between the system and the real world** The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3. **User control and freedom** Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialog. Support undo and redo.

4. **Consistency and standards** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

5. **Help users to recognize, diagnose and recover from errors** Help users recognize, diagnose, and recover from errors. Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

6. **Error prevention** Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

7. **Recognition rather than recall** Make objects, actions, and options visible. The user should not have to remember information from one part of the dialog to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

8. **Flexibility and efficiency of use** Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

9. **Aesthetic and minimalist** Dialogs should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialog competes with the relevant units of information and diminishes their relative visibility.

10. **Help and documentation** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.

One of the most frustrating moments for the user when he is using some software is when an error appears without any explanation. Apart from the ideas outlined in the item five of the principles, we have to get this other things clearly in the mind:

- Do not use jargon.
- Do not write the error, write how to fix the problem.
• Explain the cause of the problem.

• Rather than condemn the user, the message should be courteous.

• Do not use fatal, error, invalid, bad or illegal.

• Avoid long code numbers and uppercase.

• Be precise.

• Provide help about the error in the same message.

At last, we have to get two things always in the mind. The first idea is that we have to understand the user and design the software with him in mind. The second one is when the user looks at the web page, he can always answer the three basic questions:

Where am I?

What’s here?

Where can I go?

2.6 Questionnaires

I keep six honest serving-men
(They taught me all I knew):
Their names are What and Why and
When and How and Where and Who.

Kipling[14]

When you are designing an application, it is basic to bear in mind the users opinion. This opinion is normally restricted only to the Graphical User Interface(GUI), but in some cases, it can be extended to some other relevant issues of the development of a program. In this case, the users can give their opinions about the appearance of the tool, about the stylesheet used to display the results of a query and about the queries that the tool must have.

We can get the information from the users from a variety of ways(personal interview, questionnaire, workgroups, . . . ), but because the tool is constructed as a website, the selected method to mine the data from the users is going to be an online questionnaire.
2.6.1 Why

The principal objective of making a questionnaire is to assess the effects and effectiveness of something. Our questionnaire will address the usability of the dictionary querying tool.

Other possible uses of a questionnaire are:

- Gather user requirements for a software product
- Evaluate a software product
- Find people’s opinion

With this questionnaire, we try to make a ‘usable’ tool. Something is usable if it is easy to learn, it use is easy to remember, if it is effective to use, efficient, safe and provides an enjoyable user experience. Also, the answers that the user give us help to optimize the user interaction with the system, to understand what the user needs and what he is expecting from it.\[15\]

2.6.2 How

To create a usable questionnaire first of all we will have to define what our objectives are. With objectives well established, we have an overview of what we want, and we can start thinking how we are going to achieve the objectives.

The questionnaire construction is not very easy because we have to ask the right question that gives us valid and reliable information for making a decision. It is not only to ask the right question to the right person, also we have to ask it in the right way. Improper questions or questions asked improperly will most likely result in invalid and unreliable information.

The task of designing a questionnaire(according to [14]) must pass over the following seven steps:

- Review the information requirements
- Develop and prioritize a list of potential questions
- Assess each potential question carefully
- Determine the type of the questions to be asked
• **Decide on the specific wording of each question**

• **Determine the structure of the questionnaire**

• **Evaluate the questionnaire**

• **Review the information requirements**

This is the first task. Unless we understand the information requirements (what information is needed and how that information will be used), no attempt should be made to construct a questionnaire[14].

• **Develop questions**

After we have set the objectives, we must create a list of possible questions that will give us the information we are looking for. This list has to be as specific as possible, because it will be easier for us later to apply the results to the project.

Each potential question should be screened with respect to how the answers to it will be analyzed, the anticipated information it will provide and how the ensuring information will be used.[16]

• **Evaluate potential questions**

In this step, we have to look very carefully at the questions because now we start thinking about the group of people at whom the questionnaire is aimed. Are they going to understand the questions? Will they answer them? How can they react? This and other questions are the ones that the survivors from the second point must pass.

• **Types of questions**

The type of a question is quite important. It is different whether you ask someone with an open-ended question, a closed-ended one or a rating question. Depending on what you need and what you expect, you can choose one or another.

Open-ended questions leave you space to answer a question. Here, the “user” can say anything he wants, because he is not restricted. This type is very versatile. An easy-to-answer open-ended question is normally used in the beginning of the questionnaire to gain confidence, and often it is not analyzed.

In the early stages of the design, they are used because we don’t know how they will answer, so we can use this information to generate new ideas or to restrict the categories so
in the next stages we can use closed-end ones. Also we can use them when we don’t want to influence the answer.

With the closed-end ones, we define the question and the answers, so the user only has to choose. The answers that we put there can be acquired in many different ways.

Inside the closed-ends questions, we have two subtypes. One is the dichotomous question and the other one is the rating question. With the dichotomous question, we have only two possible answers or categories. Normally they are use to obtain demographic and behavioral information with a limited number of answers.

On the other hand, we have the rating scales. It is a closed-end question whose answer alternatives are graduated or organized to measure a continuous construct, such an attitude, opinion or preference[16]. Here we have to think about how many scales we are going to use, how we are going to label them and how the scale might look like.

- Constructing and wording questions

This is the trickiest part of the questionnaire making, because some words may influence some people and make the answers invalid. We have to choose words that means the same to everybody. An example used in [14, p. 45] shows us that you will have different answers depending if you ask if a movie has been long or has been short, or if you use a definite or indefinite article inside the question.

We have to think the questions with common sense, with knowledge and with experience. The questions must be brief, relevant, unambiguous, specific and objective.

- Questionnaire structure and evaluation

After we have defined the individual questions, they must be combined into the questionnaire. For this, we have to follow three basic guidelines:

1. Make it easy to administer.

2. Structure it to facilitate the transfer to the analysis.

3. Avoid biasing question answers by the order in which they are asked.

Every questionnaire is divided into some basic sections. These are an introductory section, a substantive question section, a classification section and the questionnaire instructions. In the introductory we communicate with the questioned. Sometimes it is defined inside the same document as the questionnaire, and other times it is in other document than the questionnaire.
The substantive question section contains the critical questions to the project. They are very specific ones and they are preceded with a series of questions that change the point of view from a generic one to a specific one. This is preferable because we know that the answers of the user are going to be more realistic. The classification section is used to determine the characteristics of the participants and the instruction section is to instruct the users how they have to answer it.

After structuring a questionnaire, we have to evaluate it. We have to review all the selections that we have made during the process to check that this is the questionnaire that we are looking for.

Before the questionnaire is O.K., we have to pretest it. This can be achieved by a panel of experts, the use of a convenience sample (the method we are going to use) or some other method.

3 Requirements analysis and definition

3.1 Problem owner

The problem owner is the Centre for Translation Studies at the University of Leeds. It requires a web based tool with which the students can upload and query the dictionaries that they had developed as a project for some of their modules.

3.2 System users

The system users are going to be the students that are attending the modules described in the previous section and they want to check and upload their own projects.

3.3 Project specification

The aim of this project is to design and create a tool for querying and displaying electronic dictionaries based on the TEI lite specifications. This tool has to be designed as an interactive website, using for this purpose cgi-bin’s written in Perl and the Berkeley DB XML database management system.

This tool is designated to help the Center of Translation Studies students by creating a way in which they can view the result of their own dictionary projects. Because of the difficulties with these technologies, the website have to be very simple, with a very easy way of navigating around it, and with a very simple interface to deal with. Also the errors
that the tool must provide when something is wrong have to be translated to plain English to enable normal people to understand the possible errors and solutions to them.

The main page of the website will be the first page that the users see. This has to be very friendly and very easy to work with. This page is divided in two parts. In the first part, there will be a welcome message and a short introduction to the website. In the short introduction, I will introduce the different parts of the website and a basic description of each one. In the second part of the website, there will be the links to the different parts in which the website is divided. Some of these sections are:

- Tool page.
- General information links.
- Help pages.

Each of these links will go to the proper part of the website.

The help pages will be the pages where the users go when they need to search for more information about the website. Here there will be specific information about how to make a search, how to upload a document, and other useful information.

The general information links link will go to a page in which there will be many links for making dictionaries and other stuff. The links will include some about the TEI, XML, XSLT and other things that be used for something.

3.3.1 The tool pages

This is the central part of the website. Here there will be the different cgi’s that constitute the tool.

The first tool page is the Document upload page. Here the user has to choose between uploading some dictionary that he owns or trying the tool with one of the test dictionaries that the Centre for Translation Studies has supplied us. These dictionaries have to comply with the TEI Lite DTD. The appearance will be simple, with a browser dependant file chooser field and three or four buttons to upload the user’s dictionary or the test dictionaries.

Once we have chosen the dictionary, the system will pass it to the next screen. Here, the system will display some basic information about it and show us the two possibilities that we can pick.

One of the options is to display the dictionary and the other is to query it directly.
If the user selects the display option, the system will transform the dictionary with a XSLT stylesheet and display it on the browser screen. It will have some of passing it to the query screen.

The query screen will be also very simple. Here you can query the dictionary using XPath queries. It will be divided into different parts, depending on if you want to make an easy query or a more complex one.

The easy queries will only need to specify the entry that you want to search for, and the advanced ones let you limit the result of that search. The advanced queries include a search for usage, for some word inside the examples and other queries than the users will point out with the questionnaire.

Also, there will always be the possibility of seeing the entire dictionary with some link to the query page.

3.4 Choice of development tools

3.4.1 Berkeley DB XML

The election of the Berkeley DB XML as the database management system was made because of some reasons:

- Native XML DBMS.
- Open source license.
- The same DBMS has been used with other projects in the same field with successful results.
- It has a Perl API.

Also, after reading the key features of the DBMS (taken from [17]), I was totally convinced that this DBMS will work well:

- Stores and retrieves native XML data- no conversion to relational or object-oriented models required
- Supports XPath 1.0
- Enables flexible indexing
- Supports updates to document fragments
• Supports both XML and non-XML data
• Built on Berkeley DB - proved, mature, flexible and fully featured (e.g., transactions, recovery, replication for high availability)
• Runs as a library that links directly into the application-supports multiple processes per application and multiple threads per process
• Provides C++, Java, Perl, PHP and other APIs
• Supports Windows, Linux and Solaris
• Includes complete source code

3.4.2 Perl

The election of using Perl as the programming language for the tool was based in this ideas:

• It has the API for connecting to the DBMS.
• Has been used with the DBMS before.
• It is straightforward to learn it.
• It can do powerful things with not to much code.
• ...

3.4.3 CGI.pm

The CGI.pm is a Perl5 module (by Lincoln Stein) used to write CGI scripts. It is the library that I use inside the tool. With it, you can create HTML documents on the fly, fill-out forms and also receive this forms to make something with the data sended by the user at the other side of the connection.

The syntax of the CGI.pm methods is very similar to the syntax of the HTML, so it very straightforward to use it, if you know HTML. You can create cookies to store and retrieve some information from the user, use frames to design the webpage, has support for javascript, can test it on the command line before uploading the CGI to the server and many other interesting features.
3.4.4 LibXML and LibXSLT

I have chosen these two libraries for working with XML and XSLT documents for a couple of reasons:

**Speed** The high speed of the LibXML on checking and validating XML documents and of the LibXSLT applying the transformations on a XML document were the main reasons for my choice of these libraries. I have also tried libraries built on top of the James Clark Expat parser to check and validate the XML documents, and they were unbelievably slow when they were parsing.

**Correct check and validation process** They also check and validate the data correctly. It sounds strange to say that a library did not check or validate well, but some of them do not do it. Theoretically, in the same document, you can not have two elements with the same id tag, but some of the libraries permit it without any problem.

3.5 Methodology

The model that I have used for making my final year project is the Waterfall Model. This was described by W.W. Royce in 1970 in [18]. The model is basically divided into five very well defined stages:

1. Requirements analysis and definition.
2. System and software design.
3. Implementation.
4. System testing and evaluation.
5. Operation and maintenance.

You have to pass through all of the five stages to get a complete solution. When you are in the stage \( n \), you can always go to the \( n+1 \) stage to continue the process or to the stage \( n-1 \) to review something.

In the first stage, you have to specify the problem in “concrete” terms and specify the requirements. When you are specifying the problem you are identifying it, and modeling it. This helps a lot because one of the basic things that you always have to do is to clearly define the problem.
After defining the problem, you have to limit the project. This is done with the project specifications.

The second stage is the system and software design. Here you stop thinking about ideas, and you start to get down to work. You have to take the specifications, and design something with it. Normally you will design a workflow diagram, a class diagram and some other important stuff.

In the implementation phase, you have to merge all the diagrams of the previous stage and get something that answers the problem following the specifications defined in the first step. Here you make it work.

The system testing and evaluation phase is just that. Once you have finished some module, you have to test it until you are sure that it does not break down. When the system is in some way “usable”, in your opinion, you show it to some representative group of people and they evaluate it.

The best thing with this model, and that is why I have chosen it, is that it is very straightforward, very simple, and you get a very fast development cycle. When I started this project, I had never worked with Perl or with Berkeley DB XML, so I know that I will
need to start with some simple tasks, and getting them evolving to the difficult ones through some iterations. This model suits my thoughts of how this project has to be carried.

4 System and software design

Following the Waterfall model, the system and software design will change along the time as the requirements are changing. Because this was the first time I worked with Perl and LibXML in Perl, the first few designs were very simple models because I have to do an incremental construction of the system, until I reached the model explained in the project specifications.

The first model was the simplest one. It only consists of a Data input screen and a Document validation screen. In the Data input screen, the user has to upload a dictionary from his computer. This dictionary is the one that was validated in the second part of the system. The Document validation screen was also the place in which the system will display the user’s dictionary.

After I did some tests, the design was not good, because one of the project specifications was that the tool had to be simple, and with this model, the Document validation screen was overloaded and did not look like a straightforward tool. So I made a new model. The only thing that I have changed is that the view of the document is going to be on other screen. I have separated the view from the validation.
After the appropriate trials, the model looked good, so I extended the system and I included the search capabilities. The first model had the Document search screen after the Document view screen, and it was OK, but in the implementation, a problem arised when I tried the system. It was not exactly a problem, it was a recurrence. If all the times you make a little change to the dictionary you have to reload and view it, you will be tired of
viewing again and again the same dictionary. So, I decided to change the model and make
the view and the query two parallel thread interconnected, instead of a serial layout.

5 Implementation

The system implementation has followed the rhythm defined in the System and software
design (chapter 4). Because this project has been my first approach to Perl, I have imple-
mented the models in a incremental way. This is, I have started creating the very basics,
and with that working, I have added more functionality to it.

The first model implemented was a very basic one. It was only the Data input screen
and a Document check screen. The Data input screen I have designed here is, with some
slight modifications, the same that is still in use. In the Document check Screen, the system
only writes a short notice and dumps the DOM tree of the checking process. This model
was so simple that it does not take any information from the file, or show it on the screen
in a tidy way.

After making this model work, I have added functionalities to it. The first and basic
functionality was the validation process.

When you have a XML document, there are two different operations to assure that it meets the XML standards.

**Check** This is the slightest one. It only assures us that the document follows the XML rules on how to create a XML document, that it is well formed.

A “Well Formed” XML document is a document that conforms to these XML syntax
Figure 8: Sixth model of the system

rules:

- All XML elements must have a closing tag
- XML tags are case sensitive
- All XML elements must be properly nested
- All XML documents must have a root element
- Attribute values must always be quoted
- With XML, CR / LF is converted to LF

Validation This other process is more powerful. First it checks the document, and after that, it checks that the document conforms to the rules of a DTD. A DTD is another document, normally a external one but can go inside the own XML document, that defines the document structure with a list of legal elements.

To assure that the XML dictionaries are correct, it is not enough to check it. We have to validate it with the TEI DTD of the Centre for Translation Studies. Until now, the Document check page has been working with one of the XML-checking library of Perl, XML::Parser. This library is constructed on top of the John Clark’s Expat library, and
Figure 9: The Data input screen

does a good job in checking the documents, but it cannot validate them. So, I have started searching the web, and I found two libraries that can validate the documents:

- **XML::LibXML**

- **XML::Checker::Parser**

After testing the two libraries, I have finally used the XML::LibXML library, as is explained in 3.4.4.

Once I got the validation library, I modified the Document check screen CGI and made it the Document validate screen. After this change, the next step was to query the document once it has been validated and extracted some information from it. The query of a XML document has to be made with the XPath language. This is the other use of this querying language. At the same time that it is used by the XSLT technology, you can query a XML document and obtain all the information that you want from it.
But when the system printed the information inside the file and in the same page dumped the contents of the file, the screen was overloaded with information. So, I took the decision to move the view of the file to another screen, to improve the overall impression of the tool.

The overall impression of the tool was not so good because the documents where dumped instead of treated with a XSLT engine and a XSLT stylesheet, but I thought that it was more important to start with the Document query screen and later on I would deal with these improvements.

The first query interface was also very simple. It consisted of one text box were you have to write the XPath query(only accepted XPath queries until some iterations after)and one button to send the query to the server to be executed.

The first prototypes of the tool were not implemented with the Berkeley DB XML database management system because it was not available yet in the School of Computing servers. Until that day, the project had been working with some libraries that gave the

Figure 10: The Document validation screen
possibility to make XPath queries to XML documents validated by the LibXML. They were slower than the Berkeley DB XML DBMS, but they produced the same results as the DBMS.

When this prototype was working, I started to create the XSLT stylesheet to improve the view of the document or the results of the queries. I have tried first to use one of the stylesheets that the TEI Consortium had on his web page, but they did not work well because they did not have a stylesheet for the dictionaries. To prepare a stylesheet, you have to know which ones are the tags that you are going to find inside the document. To know which were the tags that I would find inside the dictionaries, I intensely used the document “Basic rules for encoding dictionaries in XML”[19], in which the most common tags that the students use to encode dictionaries in XML are described.

![Image of XSLT stylesheet applied to dictionary data](image)

**Figure 11:** How the data is viewed with the XSLT stylesheet

With the help of this document, I created the XSLT stylesheet for the tool. After I created this stylesheet, I had to integrate it inside the tool. To present a XML document with a XSLT stylesheet, we can do it in two ways:
• Adding the stylesheet information to the XML document and leaving the browser to do the “dirty work”. This way is better for someone that does not have the tools to make the transformation in the server, because you leave all the work to the browser, but it is very imprecise, because you do not know how a browser might react to one document like this.

• Make the transformation in the server and send to the browser the HTML of the transformed document. This server-side programming is worst for the programmer because we will have to work more, but is better because in all the browser the appearance of the results will be the same, or more or less the same.

After weigh up the pros and the cons of the two ways, I decided to merge the XML and the XSLT server-side to have control of the situation, and assure that the results would be displayed in a very similar way in all the browsers.

However, the tool was very limited because it only searches by entry. So, I extended the searching capabilities and provide two complementary queries that restricted the number of results.

The first of these two new queries is the example search. Here, you have to specify some word that has to appear inside a example tag. That is, if you want to search to word big but only referring to a big city, you search for big in the entry and for city inside a example. If you do this, the tool will give you as a result of your query all the senses of the word big that contains the word city inside any of the examples.

On the other hand we have the usage search. Inside the senses, the document creator can encode some usage information about that specific meaning with the usg tag. This tag has an attribute that specifies the type of usage information that it is going to give us. The two more common usage tags are the hint and the collocation. The hint usage tag give us some clue about when and where use that translation. Whereas the collocation usage did not give us some clues, it gives us the words with that word goes with.

These two extended queries can also be combined. The Document query screen accepts also basic boolean operations with the two query types. At this moment, it only accepts the AND and the OR clauses.

5.1 Tags used to query a dictionary

To query a dictionary, we use the information about the dictionaries used by the TEI Consortium explained in the appendix B and the XPath querying language. XPath is the query-
The most important tags used to query inside a dictionary are the \textit{entry} and the \textit{sense}. The first of them defines a dictionary entry, inside one can define examples, definitions, different senses, etc. It is the most general tags in which the information about a word can be encoded.

complex hierarchical structure

But normally you did not encode the information directly inside this tag. The TEI-C also provided also some tags that have to go inside this one and structure the information

Figure 12: The Document query screen
in a better way. Inside one entry you can find these tags (normally called top-level tags):

- `<form>`
- `<gramGrp>`
- `<def>`
- `<trans>`
- `<etym>`
- `<eg>`
- `<usg>`
- `<xr>`
- `<note>`
- `<re>`

But apart from these tags, you also can find the `sense` tag. This one divides the entry into different sections, each of them can have any of the tags previously defined. Normally, the `<form>` and the `<gramGrp>` tags appears below the `<entry>` because they are generic to all the senses, and all the other tags go inside the `<sense>` because they are specific to that sense.

When we have to search for some word in the dictionary, we are going to search the entries for that specific word. Inside the entry tag, there is a compulsory attribute, `key`, in which we have to specify the word that goes inside that entry. For example (taken from [20]), if we want to encode the word `big`, the entry will be like this:

```xml
<entry id="big" key="big" lang="en">
<!-- here it goes all the information of the word "big" -->
</entry>
```

Inside the entry, we can find normally more than one sense. For example, if we continue with the same example:
<entry id="big" key="big" lang="en">
<form> form information </form>
<gramGrp> grammar information </gramGrp>
<note> note </note>
  <sense n="1">
    <def> A big person or thing is large in physical size. </def>
    <trans>
      <tr> grande </tr>
    </trans>
  </sense>
  <sense n="2">
    <def> Something that is big consists of many people or things. </def>
    <usg type="hint"> large number of elements </usg>
    <trans>
      <tr> gran </tr>
    </trans>
  </sense>
</entry>

Here we can notice some of the top-level tags that appear inside the sense information. This is because there are used to specify the information about that particular sense of the word.
When we want to constrain the results using the advanced search, we can search for words inside the examples or for some specific word usage. When this occurs, in the example search we look inside the `<eg>` tag for that specific word and inside the `<usg>` tag when we are looking for usage information. When some of these advanced searches finds something, the tool take the sense in which the match was done and shows it in the screen.

6 System testing and evaluation

6.1 System testing

The system testing is a very important phase in the design of any tool. It is normally here where the program arises all the errors that the designer was trying to avoid. These are the moments when you are more frustrated looking for the error that caused the fault.

To test the tool, I have done different types of verifications to check the most of errors before the evaluation by the users. This test have included:

- **Cross-browser and cross-platform testing:** Because different browsers have different implementations of the standards, I have to prove it with the most common of them. it has been proved with Internet Explorer (versions 5, 5.5 and 6), Mozilla for Windows and Mozilla for Linux.

- **W3 compliant:** But you can not test your website with all the actual browsers or the future ones, it is a very good idea to make your website W3 compliant, so the future browsers that implement the HTML “recomendation” will not have many problems with your code.

- **Search for GUI errors:** Sometimes, when you are writing the GUI of a program, because your mind is focused in the algorithm, you make some basic errors with it. You did not write the instructions correctly, you make orthographical and grammatical errors, etc. These are the details that you have to test here, to check them.

- **Boundary testing:** It is the systematic testing of error handling. You have to test the forms and data inputs, starting from known good values, and progressing through reasonable but invalid inputs all the way to known extreme and invalid values.

- **Performance:** This evaluation tests the server in which the website is deployed and the code that you have written. The server must be able to respond to a sufficient
number of simultaneous petitions, so the code must be optimized for a faster answer of the server.

6.2 System evaluation

The system evaluation has been done via a questionnaire. This questionnaire has been accessible from the query page of the tool, and 8 people have answered it. In it, they have answered questions about the clarity of the instructions, if it helped them to solve any problem, the color scheme and the presentation of the results and the queries that now you can make to the dictionary.

After studying the results of the questionnaire, the conclusions to it are these:

- **Instructions** The majority of the people think that the instructions were clearly defined. However, because there were people that thought that they were not well defined, I will write a more extensive set of instructions inside the help pages for those people.

![Figure 13: Are the instructions clearly defined?](image)

- **Help** The help section of the tool has not been widely accessed, but the ones that have gone inside have a good impression of it. They have searched information about the TEI and the XML, and also for a solution for some specific problems when querying the dictionary. One point that I have forgotten to implement is to create a link from all the pages to the help one.
- **Web site style** The web site style and the output presentation has been high rated by the users, but more than one has suggested that a change in the color scheme could be neccesary. I will change that detail when the tool has been deployed on the Centre for Translation Studies web server.

![](image)

Figure 14: How the visual style of the web site?

- **Queries** The queries have not been so successful like I thought. The users said that they where useful, but a more complex query system will be useful. They also suggested to me (the users from the School of Computing) that it would be good if there were a text box in which the user can directly write his own XPath query. All this propositions will be directly forwarded to the Centre for Translation Studies supervisor to assess if it is acceptable to add them to the query page.

The overall feeling of the people was good. They have given the tool a high mark and they think that it will be a good tool that will help them.

7 Conclusion

The aim of this project was to create a web-based system for the Centre for Translation Studies at the University of Leeds for querying and displaying electronic dictionaries. This system has been designed as a web site in which the tool is the main aim and with some
help pages that the users can consult to get some information. The tool was constructed with Perl as the programming language and with the Berkeley DB XML as the DBMS to store XML documents in a native form.

The objectives of the project were:

1. The tool has to upload and validate the dictionaries of the student in a XML format.

2. Design the full website for the tool.

3. Basic queries to the XML document.

4. Full and partial display of the dictionary.

5. Usability evaluation via user feedback.

The objectives have all been accomplished and exceeded in most cases. The only one that has not been exceeded is objective number 5, the Usability evaluation via feedback, because of my bad schedule (I did not bear in mind all the coursework that I had to do during the second semester, and I published the questionnaire two weeks after it was scheduled) and also because of the poor response from the people of the Centre for Translation Studies.

However, the late publishing of the questionnaire was not the worst difficulty that I have encountered while working on the project. My lacking in experience of Perl, the difficulties to get the wwwdev server running accurately and the myriad of courseworks have been my worst enemies in the achievement of the basic objectives of the project.
In spite of all this problems, the project is a success because I have extended the basic objectives of the project, I have learned quite a lot of about the technologies involved in the realization, and I have created the tool that the Centre for Translation Studies have asked for.

8 Personal Reflection

I feel that my project has been a success. I have met all the minimum requirements, and I gone beyond some of them, I have learned a lot of things while reading the project background research and I have acquired a lot of knowledge about how to make a project alone.

The project management has not been as successful as I had planned at the first stages of the project, and this has taught me not to be so optimistic when it comes the task assignment.

The information research has showed me how to search information in a much more professional way than I have been doing so until now. Also, it has helped me to learn how to discard the worst information at the first sight when you have to check vast amounts of data, as the Edward Boyle Library has on some of the subjects.

The requirement analysis is a very important part of every project, and this project has given me the opportunity to make a real requirements capture as opposed to the fake ones that I have realized in other modules in Spain.

Also, it has given me more experience in the Waterfall Model. I know that the most extended model for software development is the Rational model, using the UML and tools like Together or Rational Rose, but not for such a big project, in which there are not too many people working on it. I have worked on other much bigger projects, working with 20 more people, and the Rational model is perfect for these types of projects.

When developing the tool, I have also started working with Perl and Berkely DB XML DMS, and I have liked working with these tools. This was my first time developing a web based system, and with the good feeling that it has given me, I think I will work with these two tools again in the future.

However, despite the “failure” in the project management, my overall impression of the project is very good. It has been hard to finish it, but I think that it has been very worthwhile. Also, the success has not been only on a “professional” level, it has been also on a personal level. This Erasmus year at the University of Leeds has made me grow as a person more than anything in my life has done before and this project has helped me to discover my errors while working on it.
One more time, I would like to thank everyone that has made this project possible.
A Encoding a dictionary in XML

The TEI Consortium specifies a specific set of tags among their guidelines to encode dictionaries in XML. Apart of that set of tags, a dictionary has a plenty of other tags that are used inside other types of electronic texts.

All the information of this point is taken from [6] and from [19].

Before we start talking about the tags, the TEI Guidelines, in the chapter about the "Print dictionaries", got this affirmation(taken from [6, Chapter 12]):

“Both typographically and structurally, dictionaries are extremely complex. In addition, dictionaries interest many communities with different and sometimes conflicting goals. As a result, many general problems of text encoding are particularly pronounced here, and more compromises and alternatives within the encoding scheme may be required.99 Two problems are particularly prominent.

First, because the structure of dictionary entries varies widely both among and within dictionaries, the simplest way for an encoding scheme to accommodate the entire range of structures actually encountered is to allow virtually any element to appear virtually anywhere in a dictionary entry. It is clear, however, that strong and consistent structural principles do govern the vast majority of conventional dictionaries, as well as many or most entries even in more ‘exotic’ dictionaries; ideally, a set of encoding guidelines should capture these structural principles. We therefore define two distinct elements for dictionary entries, one (<entry>) which captures the regularities of most conventional dictionary entries, and a second (<entryFree>) which uses the same elements, but allows them to combine much more freely. It is recommended that <entry> be used in preference to <entryFree> wherever the structure of the entry allows it. These elements and their contents are described in sections 12.2 The Structure of Dictionary Entries, 12.6 Unstructured Entries, and 12.4 Headword and Pronunciation References.

Second, since so much of the information in printed dictionaries is implicit or highly compressed, their encoding requires clear thought about whether it is to capture the precise typographic form of the source text or the underlying structure of the information it presents. Since both of these views of the dictionary may be of interest, it proves necessary to develop methods of recording both, and of recording the interrelationship between them as well. Users interested mainly in the printed format of the dictionary will require an encoding to be faithful to an original printed version. However, other users will be interested primarily in capturing the lexical information in a dictionary in a form suitable for further processing, which may demand the expansion or rearrangement of the information...
contained in the printed form. Further, some users wish to encode both of these views of the data, and retain the links between related elements of the two encodings. Problems of recording these two different views of dictionary data are discussed in section 12.5 Typographic and Lexical Information in Dictionary Data, together with mechanisms for retaining both views when this is desired.

A.1 Header

The basic structure of an electronic document encoded following the TEI guidelines includes a header and the text. The header of a document must carry this set of descriptions:

File description Contains a full bibliographical description of the computer file. It is tagged with `<fileDesc>`.

Encoding description Describes the relationship between an electronic text and its source or sources. It is tagged with `<encodingDesc>`.

Text profile Contains classificatory and contextual information about the text, such as its subject matter, the situation in which it was produced, the individuals described by or participating in producing it, and so forth. It is tagged with `<profileDesc>`.

Revision history Allows the encoder to provide a history of changes made during the development of the electronic text. It is tagged with `<revisionDesc>`.

Of these, only the `<fileDesc>` element is required in all TEI headers; the others are optional.

Inside the `<fileDesc>` header, we can find these tags:

`<titleStmt>` groups information about the title of a work and those responsible for its intellectual content.

`<editionStmt>` groups information relating to one edition of a text.

`<extent>` describes the approximate size of the electronic text as stored on some carrier medium, specified in any convenient units.

`<publicationStmt>` groups information concerning the publication or distribution of an electronic or other text.
<seriesStmt> groups information about the series, if any, to which a publication belongs.

<notesStmt> collects together any notes providing information about a text additional to that recorded in other parts of the bibliographic description.

<sourceDesc> supplies a bibliographic description of the copy text(s) from which an electronic text was derived or generated.

The <titleStmt> element is the first component of the <fileDesc> element, and is mandatory. It contains the title given to the electronic work, together with one or more optional statements of responsibility which identify the encoder, author, compiler, or other parties responsible for it:

<title> contains the title of a work, whether article, book, journal, or series, including any alternative titles or subtitles. level (bibliographic level (or class) of title) indicates whether this is the title of an article, book, journal, series, or unpublished material.

<author> in a bibliographic reference, contains the name of the author(s), personal or corporate, of a work; the primary statement of responsibility for any bibliographic item.

<sponsor> specifies the name of a sponsoring organization or institution.

<funder> specifies the name of an individual, institution, or organization responsible for the funding of a project or text.

<principal> supplies the name of the principal researcher responsible for the creation of an electronic text.

<respStmt> supplies a statement of responsibility for someone responsible for the intellectual content of a text, edition, recording, or series, where the specialized elements for authors, editors, etc. do not suffice or do not apply.

<resp> contains a phrase describing the nature of a person’s intellectual responsibility.

<name> contains a proper noun or noun phrase. type indicates the type of the object which is being named by the phrase.
A.2 Text structure

After encoding the header, the following tags should be used to mark the gross structure of a printed dictionary:

- `<text>` contains a single text of any kind, whether unitary or composite, for example a poem or drama, a collection of essays, a novel, a dictionary, or a corpus sample.

- `<front>` contains any prefatory matter (headers, title page, prefaces, dedications, etc.) found at the start of a document, before the main body.

- `<body>` contains the whole body of a single unitary text, excluding any front or back matter.

- `<back>` contains any appendixes, etc. following the main part of a text.

- `<div>` contains a subdivision of the front, body, or back of a text.

- `<div0>` contains the largest possible subdivision of the body of a text.

- `<div1>` contains a first-level subdivision of the front, body, or back of a text (the largest, if `div0` is not used, the second largest if it is).

- `<entry>` contains a reasonably well-structured dictionary entry.

- `<entryFree>` contains a dictionary entry which does not necessarily conform to the constraints imposed by the entry element.

- `<superEntry>` groups successive entries for a set of homographs.

The front and back matter of a dictionary may well contain specialized material like lists of common and proper nouns, grammatical tables, gazetteers, a 'guide to the use of the dictionary', etc. These may be tagged as elements defined in the core tag set or as specialized dictionary elements as defined in this chapter.

The `<body>` element consists of a set of entries, optionally grouped into one or several `<div>`, `<div0>`, or `<div1>` elements. These text divisions might correspond, for example, to sections for different languages in bilingual dictionaries, sections for different letters of the alphabet, etc.

A sort key, given in the key attribute, is often required for superentries and entries. This is the attribute that the tool uses to search inside the dictionaries.
A.3 Entries

A simple dictionary entry may contain information about the form of the word treated, its grammatical characterization, its definition, synonyms, or translation equivalents, its etymology, cross-references to other entries, usage information, and examples.

Also, they often have a complex hierarchical structure. For example, an entry may consist of two or more sub-parts, each corresponding to information for a different part-of-speech homograph of the headword. The entry (or part-of-speech homographs, if the entry is split this way) may also consist of senses, each of which may in turn be composed of two or more sub-senses, etc.

Dictionary entries, and subordinate levels within dictionary entries, may comprise several constituent parts, each providing a different type of information about the word treated. The top-level constituents of dictionary entries are:

- Information about the form of the word treated (orthography, pronunciation, hyphenation, etc.). Tagged with `<form>`.
- Grammatical information (part of speech, grammatical sub-categorization, etc.). Tagged with `<gramGrp>`.
- Definitions. Tagged with `<def>`.
- Translations into another language. Tagged with `<trans>`.
- Etymology. Tagged with `<etym>`.
- Examples. Tagged with `<eg>`.
- Usage information. Tagged with `<usg>`.
- Cross-references to other entries. Tagged with `<xr>`.
- Notes. Tagged with `<note>`.
- Entries (often of reduced form) for related words, typically called related entries. Tagged with `<re>`.
A.4 Information on written and spoken forms

Any of the hierarchical levels (<entry>, <entryFree>, <hom>, <sense>) may contain any of these top-level constituents, since information about word form, particular grammatical information, special pronunciation, usage information, etc., may apply to an entire entry, or to only one homograph, or only to a particular sense.

The elements used to encode the form information of the word are this: as shown in the examples.

<form> groups all the information on the written and spoken forms of one headword.

The <form> element groups one or more occurrences of any of the others; it can also be recursively nested to reflect more complex sub-grouping of information about word form(s).

<orth> gives the orthographic form of a dictionary headword.

<pron> contains the pronunciation(s) of the word.

<hyph> contains a hyphenated form of a dictionary headword, or hyphenation information in some other form.

<syll> contains the syllabification of the headword.

<stress> contains the stress pattern for a dictionary headword, if given separately.

<label> in dictionaries, contains a label for a form, example, translation, or other piece of information, e.g. abbreviation for, contraction of, literally, approximately, synonyms, etc.

In addition to those listed above, the following elements, which encode morphological details of the form, may also occur within <form> elements:

<gram> within an entry in a dictionary or a terminological data file, contains grammatical information relating to a term, word, or form.

<gen> identifies the morphological gender of a lexical item, as given in the dictionary.

<number> indicates grammatical number associated with a form, as given in a dictionary.

<case> contains grammatical case information given by a dictionary for a given form.
<per> contains an indication of the grammatical person (1st, 2nd, 3rd, etc.) associated with a given inflected form in a dictionary.

<tens> indicates the grammatical tense associated with a given inflected form in a dictionary.

<mood> contains information about the grammatical mood of verbs (e.g. indicative, subjunctive, imperative).

<itype> indicates the inflectional class associated with a lexical item.

Of these, the <gram> element is most general, and all of the others are synonymous with <gram> elements with appropriate values (gen, number, case, etc.) for the type attribute.

### A.5 Grammatical information

The <gramGrp> element groups grammatical information, such as part of speech, subcategorization information (e.g., syntactic patterns for verbs, count/mass distinctions for nouns), etc. It can contain any of the following elements:

<pos> indicates the part of speech assigned to a dictionary headword (noun, verb, adjective, etc.).

<subc> Contains subcategorization information (transitive/intransitive, countable/non-countable, etc.).

<colloc> contains a collocate of the headword.

In addition, <gramGrp> can contain any of the morphological elements defined for <form>:

<gram> within an entry in a dictionary or a terminological data file, contains grammatical information relating to a term, word, or form.

<itype> indicates the inflectional class associated with a lexical item.

<gen> identifies the morphological gender of a lexical item, as given in the dictionary.

<number> indicates grammatical number associated with a form, as given in a dictionary.
<case> contains grammatical case information given by a dictionary for a given form.

<per> contains an indication of the grammatical person (1st, 2nd, 3rd, etc.) associated with a given inflected form in a dictionary.

<tns> indicates the grammatical tense associated with a given inflected form in a dictionary.

<mood> contains information about the grammatical mood of verbs (e.g. indicative, subjunctive, imperative).

A.6 Sense information

Dictionary definitions are those pieces of prose in a dictionary entry that describe the meaning of some lexical item. Most often, definitions describe the headword of the entry; in some cases, they describe translated texts, examples, etc. Normally is used the tag <def>, but it is used <tr> also.

Multilingual dictionaries contain information about translations of a given word in some source language for one or more target languages. Minimally, the dictionary provides the corresponding translation in the target language; other information, such as morphological information (gender, case), various kinds of usage restrictions, etc., may also be given. If translation equivalents are to be distinguished from other kinds of sense information, they may be encoded using the <tr> element.

A.7 Etymological information

The element <etym> marks a block of etymological information. Etymologies may contain highly structured lists of words in an order indicating their descent from each other, but often also include related words and forms outside the direct line of descent, for comparison.

Of particular relevance for the markup of etymologies are:

<etym> encloses the etymological information in a dictionary entry.

<lang> name of a language mentioned in etymological or other linguistic discussion.

<date> contains a date in any format.

<mentioned> marks words or phrases mentioned, not used.
<gloss> identifies a phrase or word used to provide a gloss or definition for some other word or phrase.

<p>pron> contains the pronunciation(s) of the word.

<p>usg> contains usage information in a dictionary entry.

<p>lbl> in dictionaries, contains a label for a form, example, translation, or other piece of information, e.g. abbreviation for, contraction of, literally, approximately, synonyms:, etc.

A.8 Examples

Dictionaries typically include examples of word use, usually accompanying definitions or translations. In some cases, the examples are quotations from another source, and are occasionally followed by a citation to the author. The <eg> element contains usage examples and associated information; the example text itself should be enclosed in a <cit> element, if attributed, or a <q> or <quote> element otherwise. The <cit> element associates a quotation with a bibliographic reference to its source.

<eg> (in a dictionary) contains an example text containing at least one occurrence of the word form, used in the sense being described; examples may be quoted from (named) authors or contrived.

<q> contains a quotation or apparent quotation - a representation of speech or thought marked as being quoted from someone else (whether in fact quoted or not); in narrative, the words are usually those of of a character or speaker; in dictionaries, q may be used to mark real or contrived examples of usage.

<quote> contains a phrase or passage attributed by the narrator or author to some agency external to the text.

<cit> A quotation from some other document, together with a bibliographic reference to its source.

A.9 Usage information and other labels

Most dictionaries provide restrictive labels and phrases indicating the usage of given words or particular senses. Other labels, not necessarily related to usage, may be attached to
forms, translations, cross references, and examples. Usage and other labels should be marked with the following elements:

<usg> contains usage information in a dictionary entry.

<lbl> in dictionaries, contains a label for a form, example, translation, or other piece of information, e.g. abbreviation for, contraction of, literally, approximately, synonyms, etc.

Typical usage labels mark

- temporal use (archaic, obsolete, etc.)
- register (slang, formal, taboo, ironic, facetious, etc.)
- style (literal, figurative, etc.)
- connotative effect (e.g. derogatory, offensive)
- subject field (Astronomy, Philosophy, etc.)
- national or regional use (Australian, U.S., Midland dialect, etc.)

Many dictionaries provide an explanation and/or a list of such usage labels in a preface or appendix. The type of the usage information may be indicated in the type attribute on the usg element. Some typical values are:

geo geographic area

time temporal, historical era ('archaic', 'old', etc.)

dom domain

reg register

style style (figurative, literal, etc.)

plev preference level ('chiefly', 'usually', etc.)

acc acceptability

lang language for foreign words, spellings pronunciations, etc.

gram grammatical usage

49
In addition to this kind of information, multilingual dictionaries often provide ‘semantic cues’ to help the user determine the right sense of a word in the source language (and hence the correct translation). These include synonyms, concept subdivisions, typical subjects and objects, typical verb complements, etc. These labels are also marked with the <usg> element; sample values for the type attribute in these cases include:

- **syn**: synonym given to show use
- **hyper**: hypernym given to show usage
- **colloc**: collocation given to show usage
- **comp**: typical complement
- **obj**: typical object
- **subj**: typical subject
- **verb**: typical verb
- **hint**: unclassifiable piece of information to guide sense choice
B  Dictionary example

<?xml version="1.0" encoding="utf-8" standalone="no"?>

<!DOCTYPE TEI.2 PUBLIC "DTD for the lexicographical database"
 "http://corpus.leeds.ac.uk/tei-lexicon.dtd">

<TEI.2>
 <teiHeader>
   <fileDesc>
     <titleStmt>
       <title>The database of the bilingual English-Spanish
dictionary of size adjectives</title>
       <author>Serge Sharoff</author>
     </titleStmt>
     <publicationStmt>
       <p>A fragment from the Oxford Spanish dictionary</p>
     </publicationStmt>
     <sourceDesc>
       <p>Created from the electronic copy by Serge Sharoff, s.sharoff@leeds.ac.uk</p>
     </sourceDesc>
   </fileDesc>
   <profileDesc>
     <langUsage>
       <language id="en"></language>
       <language id="es"></language>
     </langUsage>
   </profileDesc>
 </teiHeader>
 <text>
   <body>
     <div type="EN" lang="en">
       <entry id="big" key="big">
         <sense n="1">

51
<eg><q>a big garden</q><tr>un jardín grande</tr></eg>
<eg><q>I need a bigger size</q><tr>necesito una talla más grande</tr></eg>
<eg><q>these shoes are too big for me</q><tr>estos zapatos me quedan grandes</tr></eg>
<eg><q>her big, blue eyes</q><tr>sus grandes ojos azules</tr></eg>
<eg><q>how big is the table?</q><tr>cómo es de grande</tr></eg>
</sense>
<sense n="2">
<eg><q>a big explosion/flood</q><tr>una gran explosión/inundación</tr></eg>
<eg><q>a big hug/kiss</q><tr>un abrazote/besote</tr></eg>
<eg><q>a big success/effort</q><tr>un gran éxito/esfuerzo</tr></eg>
</sense>
</entry>
<entry id="brief" key="brief">
<sense n="1">
<eg><q>his report was brief and to the point</q><tr>su informe era breve e iba al grano</tr></eg>
<eg><q>be brief</q><tr>sé breve</tr></eg>
<eg><q>in brief</q><tr>en suma</tr></eg>
</sense>
<sense n="2">
<eg><q>it’s a very complicated brief</q><tr>es un caso muy complicado</tr></eg>
</sense>
</entry>
<entry id="broad" key="broad">
<sense n="1">
<eg><q>he had broad shoulders</q><tr>era ancho de hombros</tr></eg>
</sense>
</entry>
<eg><q>she has broad hips</q>
<tr>es ancha de caderas</tr></eg>
</sense>

<sense n="2">
<eg><q>a broad range of courses</q>
<tr>una amplia gama de cursos</tr></eg>
<eg><q>this has broad implications</q>
<tr>esto tiene consecuencias en muy diversos planos</tr></eg>
<eg><q>in its broadest sense</q>
<tr>en su sentido más amplio</tr></eg>
</sense>

<entry id="deep" key="deep">
<sense n="1">
<eg><q>the ditch is 6 ft deep</q>
<tr>la zanja tiene 6 pies de profundidad</tr></eg>
<eg><q>the deep waters of the river</q>
<tr>las profundas aguas del río</tr></eg>
<eg><q>a deep pile carpet</q>
<tr>una alfombra de pelo largo</tr></eg>
<eg><q>the deep end of the pool</q>
<tr>la parte (más) profunda</tr></eg>
<eg><q>ankle-/knee-deep</q>
<tr>hasta los tobillos/la rodilla</tr></eg>
<eg><q>the water’s only ankle-deep</q>
<tr>el agua llega sólo hasta los tobillos</tr></eg>
<eg><q>we’re waist-deep in work</q>
<tr>estamos con muchísimo trabajo</tr></eg>
</sense>

<sense n="2">
<eg><q>the soldiers were standing 12 deep</q>
<tr>los soldados formaban columnas de 12 en fondo</tr></eg>
<eg><q>the site is 100ft wide by 50ft deep</q>
<tr></tr></eg>
</sense>
<tr>el terreno tiene 100 pies de ancho por 50 de largo</tr></eg>
</entry>

<entry id="fine" key="fine">
  <sense n="1">
    <eg><q>goods of the finest quality</q>
    <tr>artículos de la mejor calidad</tr></eg>
    <eg><q>the country’s finest minds</q>
    <tr>los cerebros más brillantes del país</tr></eg>
    <eg><q>fine words, but will they do it?</q>
    <tr/todo eso suena muy bien pero lo harán?</tr></eg>
    <eg><q>a fine-looking man</q>
    <tr>un hombre bien parecido</tr></eg>
    <eg><q>it’s a fine thing you’re doing</q>
    <tr>es algo admirable lo que estás haciendo</tr></eg>
  </sense>
  <sense n="2">
    <eg><q>we had a fine time (of it)</q>
    <tr>lo pasamos de bien ...</tr></eg>
    <eg><q>a fine friend you are!</q>
    <tr>menudo</tr></eg>
    <eg><q>you’ve picked a fine time to tell me!</q>
    <tr>en buen momento me lo dices!</tr></eg>
    <eg><q>you’re a fine one to talk!</q>
    <tr>mira quién habla!</tr></eg>
  </sense>
</entry>

<entry id="great" key="great">
  <sense n="1">
    <eg><q>the Great Lakes</q>
    <tr>los Grandes Lagos</tr></eg>
  </sense>
  <sense n="2">
    <eg><q>a great many people</q>
  </sense>
</entry>
<tr>muchísima gente</tr>
<tr>a great deal of criticism</tr>
<tr>muchas críticas</tr>
<tr>the great majority</tr>
<tr>la gran mayoría</tr>
<tr>to fall from a great height</tr>
<tr>caer* de muy alto</tr>
<tr>we discussed it in great detail</tr>
<tr>lo discutimos muy minuciosamente</tr>
<tr>she lived to a great age</tr>
<tr>vivió hasta una edad muy avanzada</tr>
<tr>on an even greater scale</tr>
<tr>incluso a mayor escala</tr>
<tr>how high is it?</tr>
<tr>qué altura tiene?</tr>
<tr>the tower is 40 m high</tr>
<tr>la torre tiene 40 m de alto</tr>
<tr>a 12 ft high wall</tr>
<tr>un muro de 12 pies de alto</tr>
<tr>I’ve known him since he was this high</tr>
<tr>lo conozco desde que era así
(de pequeño)</tr>
<tr>at a high altitude</tr>
<tr>a gran altitud</tr>
<tr>to take a high dive</tr>
<tr>zambullirse*</tr>
<tr>the river is very high</tr>
<tr>el río está muy alto</tr>
<tr>this is the highest the river has been
since last spring</q>
es la máxima altura que ha alcanzado el río desde la primavera pasada</eg>
a high forehead</q>una frente amplia</eg>
high cheekbones</q>pómulos</eg></entry>
<entry id="huge" key="huge">
  <sense n="1">
    it was a huge success</q>
    fue un exitazo</eg>
  </sense>
</entry>
<entry id="large" key="large">
  <sense n="1">
    a large garden</q>un jardín grande</eg>
    he’s a large man</q>
    es un hombre corpulento</eg>
    she has a large nose</q>
    tiene la nariz grande</eg>
    try on a larger size</q>
    pruébate una talla</eg>
    large print</q>letra</eg>
  </sense>
  <sense n="2">
    a large proportion of my income</q>gran parte</eg>
    he drew a large audience</q>
    atrajo (a) una gran cantidad de público</eg>
    the largest collection of stamps in the world</q>
    la mayor colección de sellos del mundo</eg>
<entry id="little" key="little">
  <sense n="1">
  <eg><q>I don’t like little dogs</q>
    <tr>no me gustan los perros pequeños </tr>
  <eg><q>a lovely little dog</q>
    <tr>un perrito precioso </tr>
  <eg><q>what’s in the little box?</q>
    <tr>qué hay en la caja pequeña </tr>
  <eg><q>what’s in that little box?</q>
    <tr>qué hay en esa cajita? </tr>
  <eg><q>would you like some more? -- just a little piece</q>
    <tr>quieres más? -- bueno, un pedacito </tr>
  </sense>
  <sense n="2">
  <eg><q>when I was little</q>
    <tr>cuando era pequeña </tr>
  <eg><q>what’s your little sister/brother called?</q>
    <tr>cómo se llama tu hermanita/hermanito? </tr>
  <eg><q>I didn’t work while the children were little</q>
    <tr>yo no trabajaba cuando los niños eran pequeños </tr>
  </sense>
</entry>

<entry id="long" key="long">
  <sense n="1">
  <eg><q>how long do you want the skirt?</q>
    <tr>cómo quieres la falda de larga? </tr>
  <eg><q>the wall is 200 m long</q>
    <tr>el muro mide 200 m de largo </tr>
  <eg><q>it was a long three miles for the runners</q>
  </sense>
</entry>
<tr>las tres millas se les hicieron muy largas a los corredores</tr></eg>
<eg><q>it’s a long way to Tulsa from here</q>
<tr>Tulsa queda bastante lejos de aquí</tr></eg>
<eg><q>the grass is getting very long</q>
<tr>el pasto</tr></eg>
<eg><q>a long drink</q><tr>un trago largo</tr></eg>
</sense>
<sense n="2">
<eg><q>the book is over 300 pages long</q>
<tr>el libro tiene más de 300 páginas</tr></eg>
</sense>
</entry>
<entry id="long-term" key="long-term">
<sense n="1">
<eg><q>what are your long-term prospects?</q>
<tr>¿qué perspectivas tienes a largo plazo?</tr></eg>
</sense>
<sense n="2">
<eg><q>this measure will do long-term damage</q>
<tr>los efectos perjudiciales de esta medida se dejarán sentir durante largo tiempo</tr></eg>
<eg><q>long-term memory</q><tr>memoria</tr></eg>
</sense>
</entry>
<entry id="low" key="low">
<sense n="1">
<eg><q>to fly at low altitude</q><tr>volar* bajo</tr></eg>
<eg><q>the dress had a very low back</q>
<tr>el vestido era muy escotado por la espalda</tr></eg>
<eg><q>he gave a low bow</q>
<tr>hizo una profunda reverencia</tr></eg>
<eg><q>a low point in his career</q>
<tr>un momento bajo en su carrera</tr></eg>
</sense>
<sense n="2">
  <eg><q>turn the radio down low</q>
      <tr>bájale al radio</tr></eg>
  <eg><q>the TV’s on too low</q>
      <tr>la tele está demasiado baja</tr></eg>
</sense>
</entry>
</div>
</body>
</text>
</TEI.2>
References


