Lexicon convertor

Reinout van Rees
Stabu foundation

http://www.stabu.nl
$Id: lexiconconvertor.xml,v 1.14 2002/01/01 20:40:11 reinout Exp $

The lexicon convertor

Introduction

This document documents the program that converts the xml-output from the LexiCon to the xml format needed by the eConstruct project [econstruct].

Transferring LexiCon content to bcTaxonomy

Structure of the LexiCon contents

The LexiCon is an implementation of ISO PAS 12006-3 [isopas] , although it currently deviates slightly from that framework. These deviations should be regarded as issues for discussion, ultimately the LexiCon should be completely compliant with the framework. The LexiCon is developed by STABU [stabu] within the Dutch BAS organisation and is part of two European projects, CONCUR and eConstruct. *LexiCon explorer* (current filename is `lex2cat.exe`) is a browser and editor for the LexiCon format. The xml DTD (Document Type Definition) that defines the export format can be found in Appendix A.

An in-depth discussion of the LexiCon structure can be found in the *D501 - bcTaxonomy* -document available at [econstruct].

Structure of the bcTaxonomy

The bcTaxonomy follows the XTD-format (XML Taxonomy Definition) discussed in *D103 - bcXML* available at [econstruct].

Main mappings between the LexiCon and bcTaxonomy structures

This document will not cover the conversion in detail, but will introduce the most important mappings. It is organised by the resulting bcTaxonomy elements.

Taxonomy — One LexiCon `Lexicon` maps to exactly one bcxml `Taxonomy`. The `Taxonomy` contains info on which language has been used for the `NativeName`s (discussed below).

Object — LexiCon `Subject`s are mapped one-on-one to bcxml `Object`s, this is only a difference in naming. As in the LexiCon, bcxml uses an *inheritance hierarchy* of `Object`s, meaning that an `Object` further down the hierarchy inherits the properties of it's par-

ents. In bcXML this is done by adding a `SupertypeRef` to the `Object`, pointing towards it's parent, the LexiCon does this by nesting the `Subject`s.

Property — The mapping to bcxml's `Property` is less straightforward. The LexiCon uses both `Propcollection` (property sets) and `Localproperty` to connect objects and properties. A `Propcollection` is a simple grouping of properties. A `Localproperty` can be used to use the same property in multiple ways, like using the generic property "height" locally as "inner height" and "outer height".

The different properties from the LexiCon are all mapped to the bcxml `Propery`.

Measure/Unit/Value — `Measure`, `Unit` and `Value` are present in both bcxml and the LexiCon with only small differences in writing them down.

This mapping was made easier by the inclusion of the `Measure` in bcxml late in the project to facilitate the use of multiple measures for one property (now the property "fire rating" can be measured according to a European or a British norm).

NativeName — Bcxml's `NativeName` is attached to `Object`, `Property`, `Measure` and `Unit`. It is a string that is used as an xml-tag name in the bcxml catalogues and messages. So a `<Object><NativeName>Door</NativeName></Object>` will be expressed as `<Door/>` later on. For all elements this `NativeName` is generated from the normal name of that element in the `NativeLanguage` valid for the taxonomy (in eConstruct "English" has been chosen). The generated `NativeName` does not contain spaces and is either a *UpperCamelCase* (for `Object`s) or *lowerCamelCase* (for `Property`s).

Using the LexiCon convertor

Obtaining the software

The LexiCon program is available for free. Apply to Stabu [stabu] for a free license.

The LexiCon convertor is a small program that can be downloaded at [sourceforge].

Exporting XML from the LexiCon explorer

A recent lexicon.xml file is distributed with the convertor program, these are the instructions for exporting your own taxonomy or for exporting a newer LexiCon database.

- start the Lex2Cat.exe program and load the most recent .mdb file
- double-click on one of the "Subject"s in the lefthand pane in the LexiCon program
- select "export all to xml" from the "File" menu
- enter `lexicon.xml` as filename (ignore the fact that the dialogue box is a "load file" dialogue)
- after 1.5 hour (approximately, depends on your machine) the export should be finished (ignore the warning beep and warning window)

Using the convertor

To convert above XML file to the eConstruct format, follow below instructions.

- Copy above `lexicon.xml` file to the directory containing the convertor program.
- Execute the program `convert.bat` (windows) or `convert.sh` (linux) to convert lexicon.xml to bcxml.xml. An existing bcxml.xml will be overwritten.
- The file bcxml.xml now contains the taxonomy in eConstruct format

Future maintenance and adaption

The eConstruct project decided to publish some of the results as open source [opensource] software to better facilitate use and re-use of the project's results. This was also prompted by the increased attention in "Brussels" for open source software and it's advantages. The lexiconconvertor is one of the parts that's distributed as open source. (The LexiCon program itself is closed-source, but available for free. Apply to Stabu [stabu] for a free license.)

The source code can be obtained using CVS from sourceforge [http://sourceforge.net/projects/bcxml].

Using the principle described in [unittest], automated tests have been written to ensure the correct conversion. This makes it more feasible to adapt the software, as adaptions are automatically tested for correctness. The tests are available with the source code.

Lexicon.dtd

Note: this file is slightly out of date because of a number of recent changes. Around the end of January 2002 a final version is expected to be available at http://bcxml.sourceforge.net/

```
<!-- ================================================= -->
<!-- DTD for XML files holding lexicon data -->
<!-- version 0.3 of 2001-05-02 -->
<!-- Changed by Reinout van Rees (R.vanRees@ct.tudelft.nl) to -->
<!-- accommodate some changes in the lexicon xml export, like -->
<!-- nesting of subjects and properties -->
<!-- version 0.2 -->
<!-- Changed by Reinout van Rees (R.vanRees@ct.tudelft.nl) -->
<!-- version 0.1 -->
<!-- Started by Kees Woestenenk (kwoestenenk@stabu.nl) -->
<!-- ================================================= -->

<!-- Let's start with an explanation of the various -->
<!-- declarations below: -->
<!-- <!ELEMENT [ElementName] [AllowedContent]> -->
<!-- <!ENTITY % [EntityName] '[ReplacementText]'> -->
<!-- <!ATTLIST [ElementName]
        [AttributeName] [AttributeType] [Modifier/default]
        [AttributeName] [AttributeType] [Modifier/default]
        ...> -->

<!-- On naming: elements are CamelCase, attributes are -->
```

```
<!-- lowerCamelCase. -->
<!-- An attribute "somethingRefs" contains ID's of -->
<!-- "Something"s with which the element has some sort of -->
<!-- relation. <Subject propertyRefs="P1 P34"> means that -->
<!-- <Property id="P1"> and <Property id="P34"> are -->
<!-- properties of this subject. -->

<!-- "Lexicon" is the top-level element with as it's -->
<!-- children a number of elements containing the actual -->
<!-- information. All elements directly below "Lexicon" are the -->
<!-- top level elements of their tree structure. The elements -->
<!-- below it are interconnected -->
<!-- using ID's and IDREF's like the properties that are -->
<!-- attached to subjects. -->

<!ELEMENT Lexicon (Subject, Activity, Propcollection,
                        Property, Measure, Unit*, Reference ) >

<!-- A combination of subelements that's quite common is -->
<!-- that of one or more "Longname"s, optional -->
<!-- "Description"s and optional "Filename"s. In some cases -->
<!-- this is extended by optional "Shortname"s. -->
<!-- This calls for two reusable templates. -->
<!-- Apart from the initial "Longname", the elements may be -->
<!-- freely mixed. -->
<!-- LDF = "Longname, Description, Filename" -->
<!-- LSDF = "Longname, Shortname, Description, Filename"-->

<!ENTITY % LDF         '(Longname, (Longname| Description|
                 Filename)* )' >

<!ENTITY % LSDF '(Longname, (Longname| Shortname|
                Description| Filename)* )' >

<!-- Something that is also used in various places is a -->
<!-- "language" attribute. It is good to specify this in a -->
<!-- central place to be able to add languages more -->
<!-- easily. The default language is "en" (english). In version -->
<!-- 0.3 the allowed element content was changed from a list of -->
<!-- possible languages to just plain NMTOKEN, thereby ditching -->
<!-- the need to adapt this file everytime a language gets -->
<!-- added. -->

<!ENTITY % language '
        language NMTOKEN "en"' >

<!-- There is a standard set of attributes attached to most -->
<!-- elements. -->
<!-- id = an unique id. In xml it cannot start with a -->
<!-- number, so a prefix has to be added. As the ID's aren't -->
<!-- globally unique in the lexicon, the prefix indicates the -->
<!-- kind of element like 'S' for Subject. -->
<!-- version = "1.0" or something like that. -->
<!-- date = date of the last change in 2001-02-14 format. -->
<!-- status = signifies the status of the element (see -->
<!-- possible values for a clue). -->
<!-- update = kind of a -->
<!-- status-after-the-last-big-version. -->

<!ENTITY % standard '
        id         ID #REQUIRED
        version         CDATA "1"
        date          CDATA ""
        status          (-|Draft|Release|Deleted)
                              "-"
        update          (-|New|Updated|Expired)
                              "-" '>
```

```
<!-- Now follow the elements that are allowable directly -->
<!-- underneath the "lexicon" top-level element. -->

<!-- A subject is an object or a service, like door or -->
<!-- paintjob. -->
<!-- The nested subjects should be placed last, behind the -->
<!-- "longnames" etc. -->
<!-- WARNING: The attribute "componentRefs" points towards -->
<!-- multiple Components which themselves are "Subject"s, so -->
<!-- implementors should keep in mind that it points towards -->
<!-- "Subject"s (despite the name "componentRefs") but that -->
<!-- the "Subject"s pointed towards are components. -->
<!ELEMENT Subject (%LDF;,Localproperty*,Subject*)>
<!ATTLIST Subject
        %standard;
        componentRefs IDREFS #IMPLIED
        propcollectionRefs
                        IDREFS #IMPLIED
        referenceRefs IDREFS #IMPLIED >

<!ELEMENT Activity (%LDF;,Activity*)>
<!ATTLIST Activity
        %standard; >

<!-- A Localproperty is a mechanism to attach multiple -->
<!-- properties of the same kind to a subject, for instance to -->
<!-- define multiple heights h1, h2, h3. Those extra -->
<!-- identicators (h1, h(w), etc.) can be put into this -->
<!-- element's content. -->
<!ELEMENT Localproperty (#PCDATA)>
<!ATTLIST Localproperty
        propertyRef IDREF #IMPLIED
        extnr         NMTOKEN "0"
        extlanguage NMTOKEN "en" >

<!-- A propcollection is a collection of properties, grouped -->
<!-- together for some purpose. -->
<!-- The nested propcollections should be placed last, behind -->
<!-- the "longnames" etc. -->
<!ELEMENT Propcollection (%LDF;,Localproperty*,Propcollection*)>
<!ATTLIST Propcollection
        %standard;
        referenceRefs IDREFS #IMPLIED >

<!-- A property is something like "height" "length" -->
<!-- "fire-resistance". -->
<!-- The nested properties should be placed last, behind the -->
<!-- "longnames" etc. -->
<!-- WARNING: The attribute "measureRefs" is required, so -->
<!-- you ought to connect at least one Measure to a -->
<!-- property. -->
<!ELEMENT Property (%LSDF;,Property*)>
<!ATTLIST Property
        %standard;
        measureRefs IDREFS #IMPLIED
        referenceRefs IDREFS #IMPLIED >

<!-- A measure is a quantification of a property. This means -->
<!-- that a property derives it's meaning from the attached -->
<!-- measures. -->
<!-- The nested measures should be placed last, behind the -->
<!-- "longnames" etc. -->
<!-- WARNING: The attribute "unitRef" references a *SINGLE* -->
<!-- unit, not multiple! -->
<!ELEMENT Measure (%LDF;,Value*,Measure*)>
<!ATTLIST Measure
```

```
        %standard;
        unitRefs IDREF #IMPLIED
        valueRefs IDREFS #IMPLIED
        referenceRefs IDREFS #IMPLIED
        type         (Single|Enumeration|Bounded|Range)
                                "Single">

<!-- A unit is something like "meter" "inch" "kelvin" -->
<!-- The nested units should be placed last, behind the -->
<!-- "longnames" etc. -->
<!ELEMENT Unit         (%LSDF;,Unit*)>
<!ATTLIST Unit
        %standard;
        referenceRefs IDREFS #IMPLIED>

<!-- References are the mechanisms to construct a different -->
<!-- view on the lexicon's contents. You can use it to -->
<!-- attach for instance classification system's names to -->
<!-- subjects in the lexicon. -->
<!-- References can be nested. -->
<!-- The actual data is in the "description" -->
<!ELEMENT Reference (Reference*)>
<!ATTLIST Reference
        id          ID #REQUIRED
        %language;
        edition         CDATA ""
        publdate CDATA ""
        shortname CDATA ""
        title         CDATA ""
        subtitle CDATA ""
        description CDATA "" >

<!-- Now follow a few elements that are not allowed directly -->
<!-- under the top-level "Lexicon" element. They also do not -->
<!-- have an "id" attribute and therefore are not pointed -->
<!-- towards, they are simply contained within other -->
<!-- elements. -->

<!-- A value's content is in the NominalValue element -->
<!-- seqNr = This is used to distinguish between possible values -->
<!-- in an enumeration. -->
<!-- valueType = What sort of a value is included in the -->
<!-- content. -->
<!-- upperTolerance = The optional maximum value (only -->
<!-- applicable for reals, integers or numbers). -->
<!-- lowerTolerance = The optional maximum value (only -->
<!-- applicable for reals, integers or numbers). -->
<!ELEMENT Value         (NominalValue)>
<!ATTLIST Value
        measureRef IDREF #IMPLIED
        propertyRef IDREF #IMPLIED
        %language;
        id          ID #REQUIRED
        seqNr         CDATA "0"
        valuetype         (-|Real| Integer| Enumeration|
                        String| Boolean| Number)
                                "-"
        upperTolerance CDATA #IMPLIED
        lowerTolerance CDATA #IMPLIED>

<!-- A NominalValue's content is the actual value, like "2", -->
<!-- "red", "red, blue, green, yellow". -->
<!ELEMENT NominalValue (#PCDATA)>


<!-- A longname is the normal name for something, like -->
<!-- "door", "fire-resistance", "kelvin" -->
```

```
<!-- language = Indicates the language of the Longname. -->
<!-- preference = A value of "1" indicates that this name is -->
<!-- preferred for this 'something' (provided that there are -->
<!-- more "Longname"s), whereas a value of "0" indicates -->
<!-- that it's best not to use this name. -->
<!ELEMENT Longname (#PCDATA)>
<!ATTLIST Longname
        %language;
        preference (0|1|2) "1">

<!-- The shorthand name for something, used for properties -->
<!-- and units. So: "l" for property "length" and "K" for -->
<!-- unit "Kelvin". -->
<!-- language = Indicates the language of the Longname. -->
<!-- preference = A value of "1" indicates that this name is -->
<!-- preferred for this 'something' (provided that there are -->
<!-- more "Longname"s), whereas a value of "0" indicates -->
<!-- that it's best not to use this name. -->
<!ELEMENT Shortname (#PCDATA)>
<!ATTLIST Shortname
        %language;
        preference (0|1|2) "1">

<!-- A description is a textual explanation of something. -->
<!-- language = Indicates the language of the description. -->
<!ELEMENT Description (#PCDATA)>
<!ATTLIST Description
        %language; >

<!-- A filename is either a local filename or an internet -->
<!-- hyperlink containing more information like a VRML file -->
<!-- or a picture. -->
<!ELEMENT Filename (#PCDATA)>
```

Bibliography

[econstruct] Michel Bohms. *econstruct web-site* . On-line at econstruct.org [http://econstruct.org].

[isopas] *ISO PAS 12006-3* . On-line at icis.org [http://www.icis.org/tc59sc13wg6/] .

[opensource] *open source web-site* . On-line at opensource.org [http://www.opensource.org].

[sourceforge] *sourceforge download site*. On-line at sourceforge.net [http://sourceforge.net/project/showfiles.php?group_id=20109].

[stabu] *Stabu foundation web-site* . On-line at stabu.nl [http://www.stabu.nl].

[unittest] *unit tests*. J. Donovan Wells. On-line at extremeprogramming.org [http://www.extremeprogramming.org/rules/testfirst.html] .