Representing and describing words flexibly with the dictionary application TshwaneLex

David Joffe
TshwaneDJe HLT
david.joffe@tshwanedje.com

Gilles-Maurice de Schryver
Ghent University & TshwaneDJe HLT
gillesmaurice.deschryver@{UGent.be, tshwanedje.com}

Abstract

This paper describes the fully customisable and built-in DTD (Document Type Definition) editor of TshwaneLex. This powerful tool, which is based on XML (eXtensible Markup Language) standards, allows lexicographers to tailor the dictionary grammar of any project, and thus to truly represent and describe words flexibly.

The demands placed on modern dictionary databases

A lexicographer typically summarises each word's analysis in the form of a meticulously constructed dictionary article. Thousands of such analyses are then brought together in reference works. Today's reference works are electronic, so are the databases underlying them. With representations and descriptions of words becoming increasingly multifaceted and interlinked, the demands placed on modern dictionary databases grow exponentially. In this paper it is shown how one of the main challenges, namely that of providing a customisable DTD to lexicographers, was met in the dictionary application TshwaneLex.

TshwaneLex is a true hybrid in that it allows for the creation of monolingual, bilingual and semi-bilingual dictionaries, for virtually any language, thanks to full Unicode support, as well as support for the Windows Input Method Editors ("soft keyboards"). At the heart of TshwaneLex lies a fully customisable DTD with which the dictionary grammar may be modified by the lexicographers themselves, and this without the need for an IT expert. Although the mindset that creating and modifying a DTD is "something you give to your IT expert to do for you" is in the dictionary industry already, TshwaneLex offers users the choice of doing this themselves if they want / need to without an IT expert.

Basics of hierarchical dictionary data modelling in TshwaneLex

The basic structure of each dictionary article is hierarchical. One lemma may contain several word senses, and each of those word senses may in turn contain sub-senses. Any of the senses or sub-senses may contain usage examples or MWUs, where the latter may again in turn contain one or more senses, etc. When creating a dictionary article, the lexicographer's task can be seen as consisting of two essentially separate but

closely related sub-tasks: (1) to specify the basic skeletal structure or layout of each dictionary article (represented as a Tree View in TshwaneLex), and (2) to flesh out that basic structure with content (which may be inputted in or selected from sub-windows accessible with the function keys F1 and F2 in TshwaneLex).

Borrowing terminology from XML, the root and branches of the Tree View may be called *elements*, the text content values *attributes*. There are different types of elements, and every element type has its own set of attributes that may be associated with it. For example, a Lemma element could consist of the lemma sign itself, a pronunciation field, and an etymology field. For a usage-example element, say E.G., there may be three attributes, viz. the usage example itself, a translation of that example (if a bilingual dictionary), and optionally the source (citation).

When one clicks on any element in the Tree View, the Attributes (F1) sub-window shows the attributes associated with the type of element one has clicked on, and one may immediately proceed with entering or modifying the relevant contents. These links between (1) internal article structure, (2) Tree View, and (3) boxes / fields to be filled in by the user, constitute one of the crucial design features to ensure smooth and sound compilation. Only the relevant attributes are seen at any given point, so that the screen does not get clogged, and so that the potential for errors is minimised as only attributes allowed by the DTD are editable.

The Document Type Definition in TshwaneLex

A DTD, also known as a 'schema', is a description of the structure of the articles in a particular dictionary, or essentially, a description of the types of elements that a dictionary has, as well as the attributes that each of those elements may have. Every dictionary document needs a DTD, and this is typically mostly set up as completely as possible at the start of a dictionary project, with perhaps minor modifications later on. The DTD is then used while creating the dictionary, and by enforcing conformance to a well-designed DTD, one can ensure that the final dictionary follows a logical and consistent structure throughout.

When work on a new dictionary begins, TshwaneLex creates a basic default DTD. This may then be customised by the user, although a few DTD element and attribute types are fundamental to TshwaneLex and should not be changed, such as the Lemma element's LemmaSign and HomonymNumber attributes (as the program automatically calculates homonym numbers). Three basic aspects are defined by a DTD: (1) valid element types, (2) element attributes, and (3) element child relations.

In Figure 1, a screenshot of the DTD structure tab of the TshwaneLex DTD editor dialog is shown. The full list of all the valid element types that may appear in this particular dictionary, appears on the left under 'Element types'.

Each element in the DTD has its own element attributes or set of attribute types. When an element is selected in the list of all the element types, the 'Attributes of this field' section displays a list of the attributes for that element. For example, if the Lemma element is selected, the basic Lemma attributes are shown, such as LemmaSign, Pronunciation, Etymology, etc. (These are of course also the attributes that are shown under Attributes (F1) when a node of this element type is selected in the Tree View while editing the dictionary. The names and order for these attributes are also configured here.)

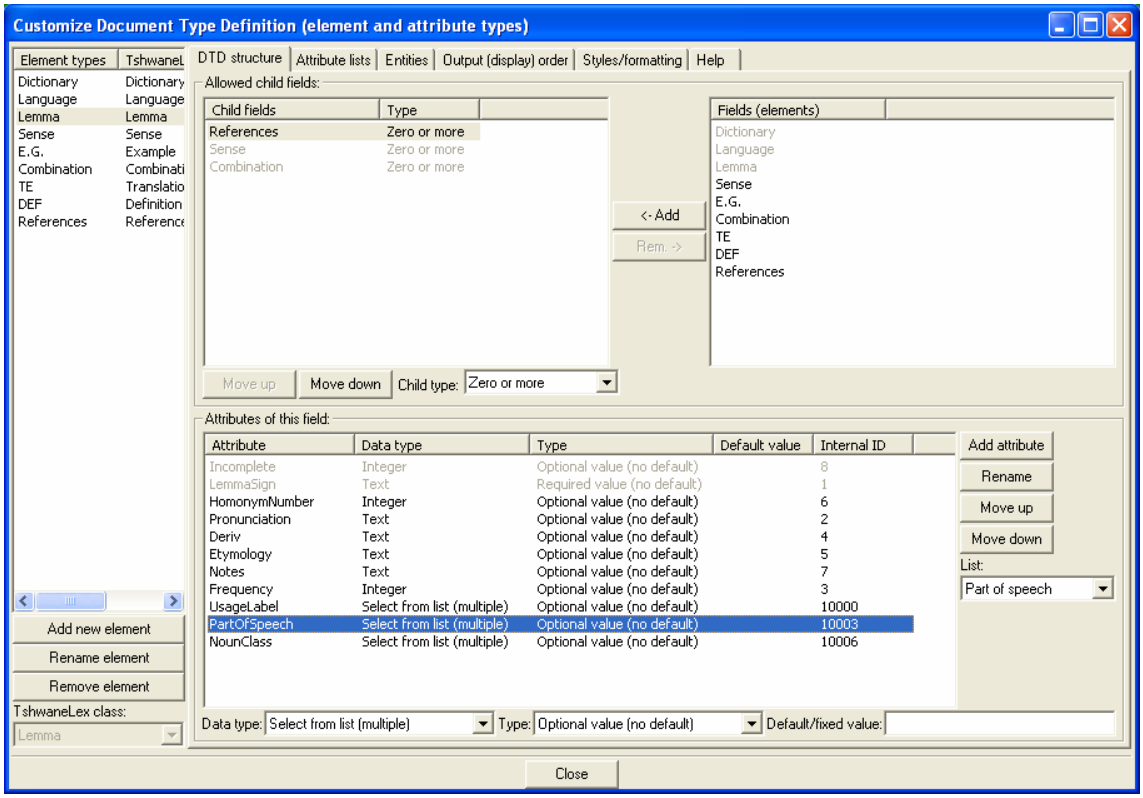

Figure 1. TshwaneLex 'DTD structure' editor dialog

The list of elements at the left of the DTD structure editor dialog is a flat list, and does not describe which element types are allowed to be added under which other element types when adding elements in the Tree View. Clearly only certain elements may be added to others. For example, one cannot add a Lemma element beneath a usage example. One therefore needs to define element child relations, which describe the allowed child element types of each element. This is done in the 'Allowed child fields' section in the DTD structure editor dialog. This section contains two lists. The left one, with header 'Child fields', is a list of the currently allowed child elements for the selected element type. The list on the right is simply a list of all element types, from which to choose to add new child types.

Each parent-child element relation may also have a constraint associated with it. There are four possibilities, which may be selected using the 'Child type' drop-down list in the 'Allowed child fields' section of the dialog: (1) One child of this type only, (2) Zero or more children of this type, (3) One or more children of this type (i.e. 'at least one'), and (4) Zero or one child of this type. When compilation proper proceeds in the Tree View, TshwaneLex will of course not allow these constraints to be broken and will 'grey out' certain options when needed, preventing lexicographers from creating invalid articles and thus ensuring consistency throughout the dictionary.

The styles system in TshwaneLex

The styles system in TshwaneLex is used to 'transform' the structured data of each dictionary article into the corresponding output in the preview area, and is also used when exporting to formats such as RTF or HTML.

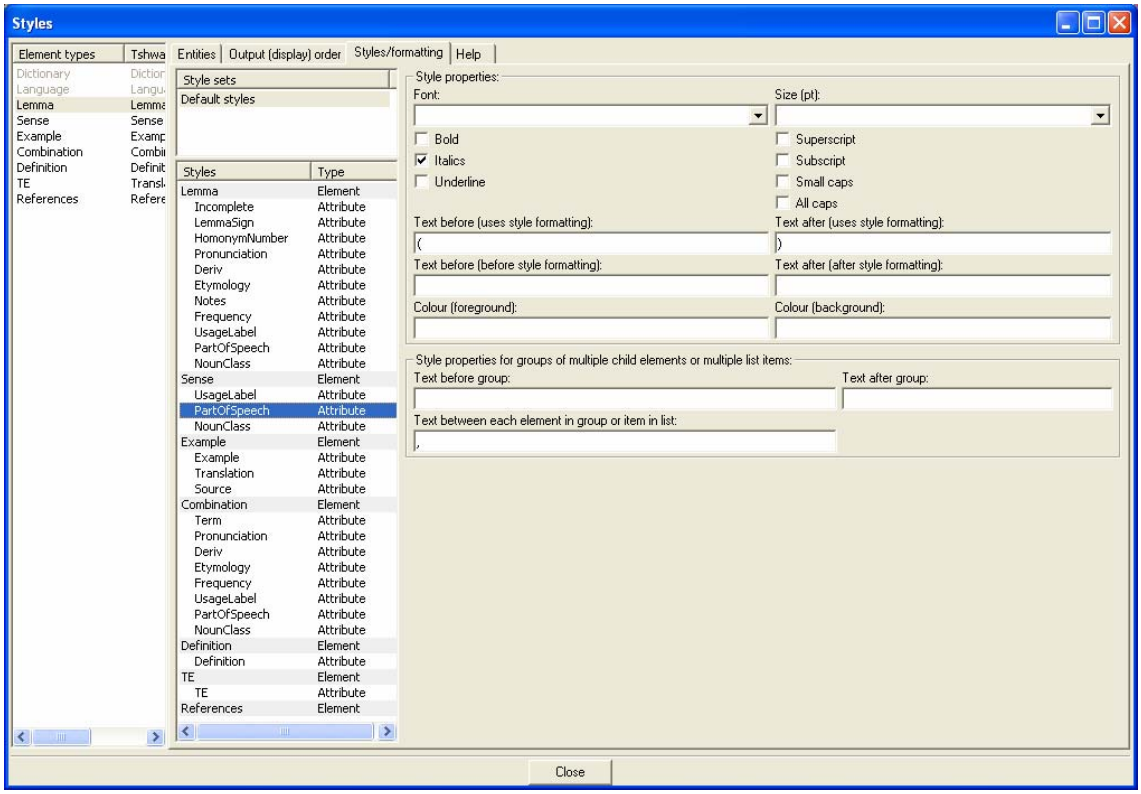

Figure 2. TshwaneLex 'styles/formatting' editor dialog

Every element and every attribute in TshwaneLex may have its own style. A 'style' allows various formatting options (such as font face, font size, text colour, bold, italics, underline, superscript, subscript, etc.) to be used for that element or attribute. The styles in TshwaneLex can be modified in the styles editor dialog, as seen in Figure 2.

Figure 2 shows a list of all available styles in a particular dictionary, in the column headed by 'Styles'. Element styles are highlighted in grey, and the attribute styles for each element are shown beneath it. The style of a selected element or attribute from this list may be modified in the 'Style properties' section on the right. Style changes are applied immediately in the preview area, thus one may instantaneously see the effects of style changes while changing the styles. These changes are moreover effected throughout the entire dictionary.

In addition to formatting options such as bold and italics, common text and punctuation that should appear before or after specific elements or attributes can be configured with the style. This can be used, for example, to place the POS between round brackets, the etymology between square brackets, etc. By using styles, this sort of punctuation may be changed throughout the entire dictionary at once, simply by changing the style. The lexicographer is furthermore also freed from entering such text or punctuation manually.

Note that there are, for each style, two 'Text before' and 'Text after' options. The first of these, labelled 'uses style formatting', will be output with the other formatting options of the style (e.g. bold) applied. The other options, labelled 'before style formatting' and 'after style formatting' will be output before / after the formatting options of the style are applied.

Element styles have additional properties that attribute styles do not have. These are shown in the bottom right area of the styles editor dialog under 'Style properties for groups of multiple child elements or multiple list items'. These are options that are used for groups of elements. One is dealing with a group of elements when there are multiple child nodes of that element type under another node in the tree (e.g. if the Lemma element has multiple word senses then there is a group of Sense elements, if a Sense element has multiple Definition elements then there is a group of Definition elements). The element group style options allow specified text to appear before or after the entire group of child nodes of the same type, as well as between each element, allowing, for instance, a comma or semi-colon to be used to separate multiple definitions or usage examples.

Element and attribute output order in TshwaneLex

The output order of elements and attributes in the preview area (and in formatted output such as RTF or HTML) is configured separately to the DTD. This allows some separation of the underlying data structure and the final visual output. The output order of elements is configured using the 'Output (display) order' tab in the DTD editor dialog. For each element type, one can configure the output order of both child elements of that element type, and attributes of that type – all relative to one another. Compare in this regard Figure 3.

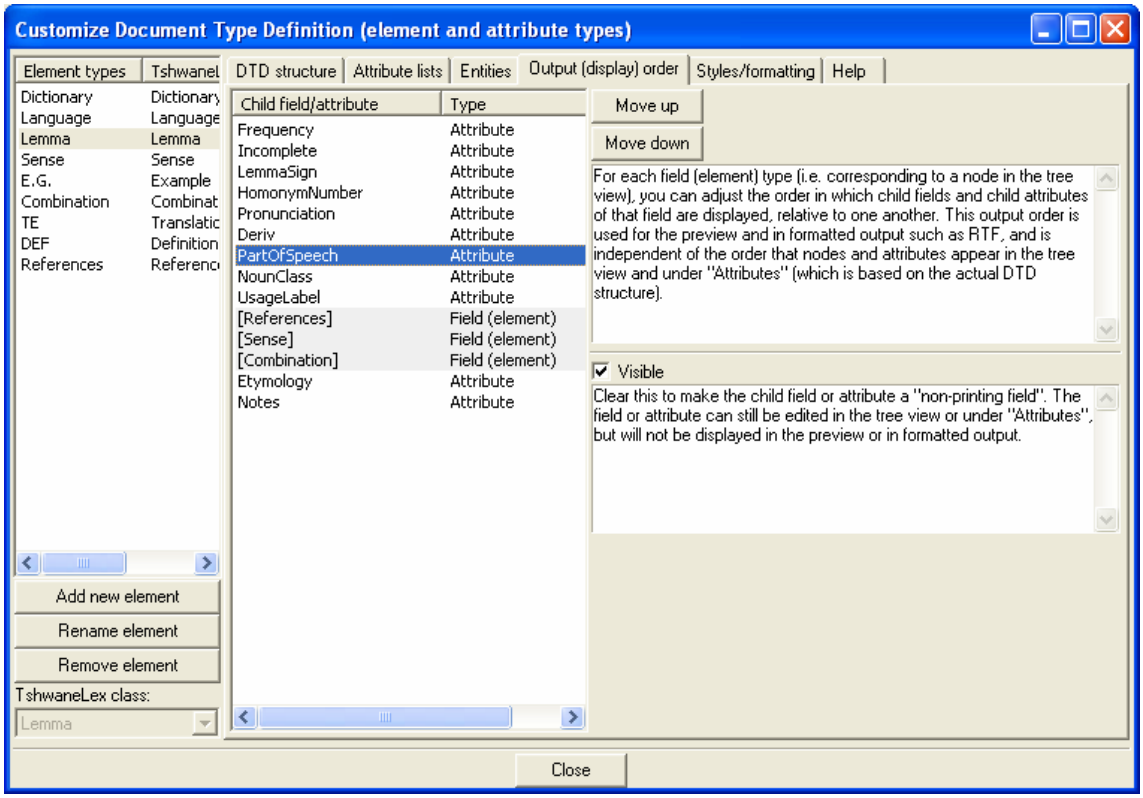

Figure 3. TshwaneLex 'output (display) order' editor dialog

In Figure 3, the output (display) order for the Lemma element type is shown. This element has a number of attributes such as LemmaSign, Pronunciation and Etymology, as well as, among others, a number of child element types, including child senses ('Sense'). One sees that some of the Lemma attributes (such as LemmaSign and Pronunciation) appear before those child senses, while other Lemma attributes (such as Etymology) appear after the child senses. The output order of this list of attributes and child fields for Lemma can now easily be modified by using the 'Move up' and 'Move down' buttons. As another example, whether the HomonymNumber attribute should be placed before or after the LemmaSign attribute for all articles throughout the central text for the production of a certain dictionary, may simply be decided at the output stage.

Each child element or attribute in the output order configuration may furthermore also be selected to be hidden in the preview area or output. This can be done by toggling the state of the 'Visible' checkbox. This allows any fields to be marked as non-printing fields, thus fields that should not appear in the output of a certain dictionary.

Customisable DTDs in TshwaneLex: Rather formidable and intimidating?

No doubt, to new users the DTD editor dialog as a whole will appear to be rather formidable and intimidating. Yet it is worth investing some time in studying its mechanics. With this customisable DTD users now have an entire printing house at their fingertips. Compare in this regard the following advice by Zgusta in the recent anthology of lexicography:

> A matter of great practical importance is the decision concerning the font types in which the dictionary will be printed; this problem should be given attention from the very beginning of the work and should be finally decided as soon as the lexicographer begins to write entries. It would seem that there is no problem in this decision, that the headword is simply printed in bold type, the rest of the lemma in smaller italics, the definitions in italics, the examples in roman, etc. In reality, it is not an easy task to make all these decisions; they should be made after a thorough consultation with the prospective printer. (Zgusta 2003: I 77; being a revision of Zgusta 1971: 353)

This might indeed have been the case until a few decades / years ago, and might actually still be the case at various dictionary publishing houses where the bonnet cannot be taken off their typesetting programs. However, as should be clear from the discussion of the styles system in TshwaneLex, the layout aspects Zgusta mentions can trivially be changed at any point in time, and can also be adapted for the production of any type of dictionary. Moreover, TshwaneLex offers a true database in the literal sense of this word, in that various products may be derived from it that contain selections of the 'base of data', and these data may be presented in whatever order is appropriate for the production of a particular dictionary. On top of all this, the customisable DTD safeguards the dictionary grammar at all times.

References

TshwaneLex (2002-2005). Online info, http://tshwanedje.com/tshwanelex/ (accessed: 31 March 2005).

Zgusta, Ladislav (1971), *Manual of lexicography* (Prague: Academia).

Zgusta, Ladislav (2003), 'Planning and organization of lexicographic work', in Reinhard R.K. Hartmann (ed.), *Lexicography: critical concepts* (London: Routledge), pp. I 70–82 (Ch. 3).