One Structure for Both Monolingual and Bilingual Dictionaries
Converting a Large Number of Different Dictionaries to a Single XML Format

Hans de Groot and Pieter Masereeuw
Van Dale Uitgevers

Van Dale converted the source databases of all its dictionaries to a type of XML mainly designed to capture the function of its elements, rather than their formatting. We found that we could apply the same principles consistently to various types of dictionaries (monolingual, bilingual) and capture all content within a single XML structure. The new structure reduces the time needed for our production processes and for database maintenance. This article reports on our findings during the conversion and the principles we applied.

1. Introduction

Van Dale is the leading publisher of dictionaries in the Dutch-speaking world. It carries about 150 titles, both monolingual and multilingual, ranging from comprehensive dictionaries to mini-pocket editions.

During the past year, Van Dale converted the source databases of all its dictionaries to XML. The immediate reason for the conversion was the introduction of the iLEX editor[1], but in addition, we had been looking for a single structural model which we could use for the entire process, from conception of the title to multi-platform publication. The main objective was to develop a more focused, manageable, efficient and inexpensive work process.

We wanted to develop a structural model that takes account of every stage of the process. This model needed to be suitable for content management, content editing and content publication. Before we started, we determined a number of principles. The actual model came into being during the conversion.

The result of the project is that all databases now have a single central structure - irrespective of whether they are monolingual or bilingual. In this article we outline how we achieved this.[2]

2. Background

Until the mid 1990s, a dictionary database was the basis for just one product: a book. Nowadays, many different types of publication can be derived from a single database, each with its own design: books, CDs, online applications, iPhone/PDA applications, etc. The number of applications taken from a single publication has drastically increased. But at the same time, the shelf life of those publications has reduced considerably. When we only had books, a new edition would be published every five to ten years. In the case of CDs, an update frequency of about once a year was usual. Online editions call for even more frequent updates, perhaps even daily: because, after all, doesn't language change daily too? Updating and publishing have changed: instead of occasional activities they have become continuous processes.

[1] iLEX is an XML dictionary writing system, developed by Erlandsen Media Publishing (EMP), Denmark. More information at www.emp.dk.

[2] This article is not based on literature. Instead, it is a report of Van Dale's experiences with converting its databases to XML. Although the structure of Van Dale's databases may not be immediately transferable to those of other publishers, we feel that the way we approached this problem may be a point of departure for others wanting to carry out a similar process.

To satisfy the demand for more and increasingly frequent publications, we need an efficient automated production process. Efficiency can be hugely increased if we use a single model to produce all derivative products. Because a single database can give rise to products in various media, such a model must be subject to strict rules: the structure of the database file must be fully predictable.

We found that the Van Dale databases did not meet this requirement. The main obstacles were:

- The structures used were geared to the design of a particular final product and were not medium-neutral
- The structures differed per product, which meant that specific and non-reusable software was needed for each product, which was expensive and time-consuming
- The structures were too intricate: infrequent patterns with low relevance for the products demanded unnecessarily complex processing software

The above made it clear that intervention in the source files was required. As a result, we would be able to get rid of the most complex and most error-prone production scripts.

3. Database Principles

In order to arrive at a homogeneous collection of possible article structures, our main objective was to reach simplicity and transparency. From this objective we derived the following principles:

- What is the same must be called the same;
- What is not the same must not be called the same;
- Information must be explicit and not be hidden away in external documents or derivation scripts;
- Information must be able to be brought into production without manual intervention;
- Low-frequency patterns may only be applied if they have genuine added value;
- Structures must be provided with a clear, understandable and consistent nomenclature;
- It must be clear what each structural element is and where it belongs;
- The distinction between content and structure must be well-considered. For example, if you indicate word class (noun, verb, adjective) not as text, but by means of an XML element, it is only a small step to making sure that nouns are accompanied by other XML elements than verbs are, for example.
- Design (formatting) plays no role in structure; function is central. Design is only linked to structure during product derivation;
- Gaining space by compressing information only occurs during derivation and only with programmable actions;
- We do not need to reinvent the wheel; we can use insights and techniques that are already available (for example, from the makers of our iLEX editor);
- The process of redefining structure runs most smoothly if carried out by a small, dedicated team. There is no need for a broad-based, democratic process. However, the results will be discussed in detail with all those involved.

If you apply the above principles consistently, the creation of a dictionary turns out not to be so complicated.

4. Results

The base structure that we developed is as follows (figure 1):

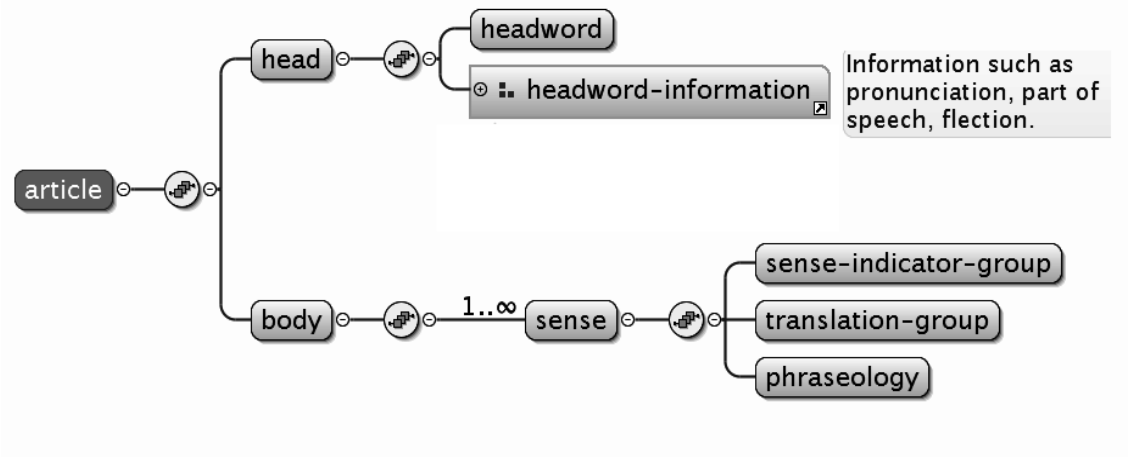

Figure 1. the base structure

Crucial in the base structure is the *sense indicator group*. This group makes it possible to use the same structural model for both monolingual and bilingual dictionaries. In this we deviate from the usual view of seeing the *definition fields* in a monolingual dictionary as the equivalent of the *translation fields* in a bilingual dictionary. In such a view, these fields would have the same position in the structure, meaning that the monolingual dictionary would need a different structural model than the bilingual dictionary.

In our view, bilingual dictionaries also regularly provide information to defining various meanings. Such definition fields are usually much shorter than those found in a monolingual dictionary, but in both cases they represent the same, defining, element.

An example: **mouse** 1. [animal] muis
2. [comp.] muis

In our new structure, we call the labels [animal] and [comp] *sense indicators*. They are comparable to the (much more comprehensive) definitions in a monolingual dictionary. The sense indicator group therefore contains all information that is essential to a meaning (sense) of the word. In monolingual dictionaries, this is what it is all about; in bilingual dictionaries this group is primarily used when there is a need to distinguish between various meanings.

In bilingual dictionaries, the sense-indicator group will be followed by a translation group; obviously, the translation group will not be present in monolingual dictionaries.

The base structure itself contains both groups. We decide per product which group(s) are to be included in the product structure and which rules are to be applied.

5. Reuse of Structures

Another characteristic of the new base structure is that it can be continually reused. The information you give with an example sentence is, in essence, no different to the information you give at headword level: both structures have a head (the word or example sentence that has to be explained or translated) and a body (the explanation or translation). The element

containing the translation of the example sentence does not itself require a name. The fact that the translation is contained in the example group provides sufficient information. The element containing the translation of the example sentence can therefore have the same name as the element containing the translation of a headword. This reduces the number of named XML elements required.

6. Tailoring the New Structure

Reusing the same structure over and over again for all the Van Dale dictionaries sounds great in theory, but does it work in practice? In the XML world, dictionary articles are XML texts, and good XML texts require a formal definition of structure, laid down in a DTD or an XML Schema file. During the project, we found that we could in fact use just a single XML structure for all our dictionaries.

We chose W3C XML Schema to define our structure formally. We augmented the structure with Schematron rules. The difference between Schema and Schematron is that Schema lets you define the contents and order of fields, whereas Schematron can enforce long-distance dependencies. For instance, in Schema it is hard or impossible to specify that the headword of a reflexive verb should be accompanied by a reflexive pronoun, but in Schematron you can – and easily.

Now, after conversion, all Van Dale dictionary structures are contained in just one XML structure. In addition, each dictionary has a derived structure specific to that title. The derived structure is created automatically from underlying structures and Schematron files. One of the benefits of XML Schema and Schematron is that they are XML languages in themselves, which makes it possible to apply XML transformation techniques such as XSLT. This is what enables us to define and maintain one large, abstract structure for all Van Dale dictionaries, and to derive concrete, tailored dictionary-specific Schema and Schematron files automatically from that one structure. We need derived structures for specific dictionaries because we need to be able to switch certain fields on and off, depending on the language and the type of dictionary.

Although the Schema standard has several built-in mechanisms to enable the user to extend or redefine the structure, when working with this program we actually found these mechanisms cumbersome, unnatural and overly complex. We therefore decided to define our own mechanism, which is not ideal but does have the benefit of being simple to understand and maintain.

Our mechanism makes it possible to mark sections of Schema and Schematron rules with so-called processing instructions. With processing instructions, we can specify to which dictionaries a certain section are is relevant. Also, processing instructions can be used to redefine the repeatability of field sequences (optional, zero or more times, one or more times).

7. Conclusion

By avoiding complexity and maintaining transparency, we made the dictionary creation process much more efficient and cost-effective. We could capture the content of many different types of dictionaries, monolingual and bilingual, within just one structure. By doing so, we made our editorial process much less elaborate. The new structure also greatly enhances software reusability.