Use of Ontology for production of access systems on Legislation, Jurisprudence and Comments

Jean Delahousse[1], Johan De Smedt[2], Luc Six[2], Bernard Vatant[1]

[1] Mondeca - 3 cité Nollez
75018 Paris France
jean.delahousse@mondeca.com
bernard.vatant@mondeca.com
[2] Wolters Kluwer, Waterloo Office Park
Drève Richelle 161 L
BE - 1410 Waterloo Belgium
johan.desmedt@wkb.be
luc.six@wkb.be

Abstract. Wolters Kluwer Belgium publishes about specialized areas related to legislation, jurisprudence and doctrine. The paper reports on an effort to transfer knowledge, scattered over a divers set of classification, coding and index generation systems, into a central thesaurus system, modeled and controlled by an ongtoloy.

1 Wolters Kluwer Objectives

1.1 The Challenge

At WKB, we started with a diverse set of specialized paper and CD publications. For each of these specialized products, end-of-book indexes are maintained and produced by dedicated tools. Typically these tools manage part of the editorial and/or production flow only for a small set of publications in the same specialization area. WKB has decided to bring this content on-line and keep it up to date instantly. The on-line search quality must provide accurate access to the specialized content. Moreover, the cost for editorial index maintenance should not increase. The effort invested in classifying content must be reused in all production flows. The access mechanisms for on-line products have some new challenges. Whereas an end-of-book index exclusively handles a specialized area, on-line content can be packaged and may be subscribed to as requested by the market (end-user needs). The new access mechanisms need to go across specialization areas. A complicating factor in this process was the diverse systems used to mange the content itself. Content was kept and maintained in different system ranging from file systems, over typesetting system to specialized CMS.

1.2 Strategy

The overall strategy had several axes. Those most relevant for this paper are:

- Reducing the number of CMS and introducing the Content Unit as the abstraction of a reusable document;
- Introducing the concept thesaurus model. The system had to be able to support the integration of the existing indexing systems;
- Reorganizing the editorial and production workflow of the publishing process so that content enrichment is done upstream and not bound to production deadlines.

1.3 The action list

Consequences of the above strategy for the access structures were ambitious: Migrate from a indexing process embedded in the production phase to:

a content driven classification embedded in content acquisition and an automated, thesaurus based index generation at production time

Production process changes:

- Automating index generation as much as possible. Corrective procedures have to become rule based and executed by the index generation process.

- With this automated index generation tool, change the editorial workflow procedures. Set up a content driven workflow for classification effort.

Provide classification services on content in multiple Content Management Systems.

- Objectives: classify content wherever it is managed

Service on-line production as well as CD, DVD and paper based productions, all from the same thesauri and classification.

- Objective: reuse the classification information on all channels where the content is being published. The index generation process must be controlled by the delivered content. Classification information of the published CU must feed into the on-line indexing and search engines.

Reduce the number of indexing systems and merge different thesauri when possible.

- Objectives: Replace the different classification systems by one new system Facilitate the merger of thesauri Reduce classification effort by using multilingual thesauri

2 Implemented Solutions

Before detailing the thesaurus management system (TMS) we need to clarify some concepts and their naming as used in the project implementation. Some of these concepts were already touched upon in the previous chapter.

2.1 Introduction: Basic Concepts

The **Content Unit:** A unit of text identified and managed by a CMS. The CU corresponds to an abstraction of a document. The abstraction means a document can be identified independent of its version history. Each document has at least some general meta data like an identifier and a title. Example: a law.

The **Content Unit part**: An identifiable part of the CU. The identifier of the CU part is managed in the scope of the CU. Typically, CU-part identifiers are not changed when its text changes or when it's encapsulating CU is versioned (exception: when the CU-part is removed, the identifier is removed and will not be reused). Each CU has some form of sub-title or label. Example: an article.

The Content Unit part **Range**: The range specifies a contiguous text within a CU. Comment may be added to the range to describe it in a human readable form. Example: art. 3 – art. 5

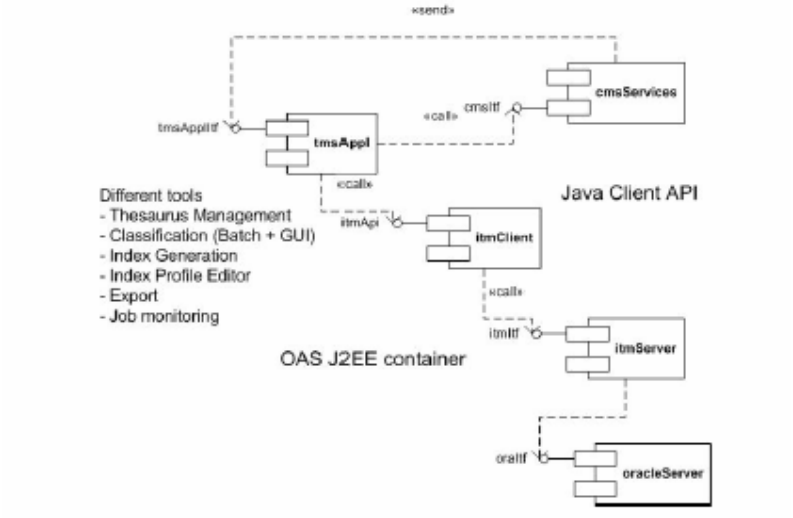
Classification: The link between a text in a CU (the CU, a CU-part or a Range) and a set of thesaurus terms, used to characterize the text. Example: in art 3, points 3, 5-7

Folder: publication specific set of content references. Each content reference identifies the CU (or CU-part) and the location where the CU-part is included in the publication.

Index: In the presentation, refers in particular to back of book indexes.

An **index Profile** is a set of rules applicable for a publication, It is invoked on a folder referencing CUs that are classified by a thesaurus. The result of applying these rules is a back-of-book index.

2.2 System Architecture



The thesaurus system is a set of applications, built around a J2EE service (ITM from Mondeca). The general architecture is depicted in the picture below. Some TMS applications may access other services as well. The CMS is a typical example of such a server. The TMS application defines an interface for its CMS access. Next to the depicted API interface, ITM also offers web interfaces.

2.3 Manage multiple Thesauri

The business provides us with a synergy problem. Occasionally, multiple end-of-book indexes are the basis for thesauri. Some were conceived in well separated and specialized domains. Others however have a great overlap (e.g. French and Dutch indexes on the same content). To handle these two synergy problems, the thesaurus management module has two strategies: **Merging thesaurus terms** to re-organize a thesaurus and **Mapping thesauri terms** to enable integration among thesauri. Both strategies are implemented as API and in the UI.

2.3.1 Merging thesaurus terms

Merging thesaurus terms allows cleaning-up poorly formed thesauri. Typically the poor thesaurus structures result from the migration of index based classifications. Merging terms, deprecates one term for its preferred term. The following sub-actions may be parameterized:

- Classifications of the deprecated term are taken over by the preferred term.
- Associations (RT-RT and BT-NT) may be taken over by the preferred term.
- Names and synonyms of the deprecated term may be taken over by the preferred term.

2.3.2 Mapping thesauri terms

Mapping thesauri terms allows transfer of classification information between terms of different thesauri. The mapping is registered. The classification information is copied (not moved). Names may be copied (becoming synonyms). BT-NT associations may be traversed and copied (deep mapping) or not.

2.4 CMS interfaces

In general, the CMS interface provides functions to:

- signal new or changed CU or CU meta data (CMS to TMS)
- report classification change (TMS to CMS)
- get a read-only copy CU (standard CMS service)
- get extra metadata of a CU (standard CMS service)

The interfaced CMS systems are:

– BRONS: A proprietary WKB CMS handling legislation and jurisprudence
– SigmaLink: A commercial CMS handling specialized documentation and comment. Content in other legacy system are migrated to one of these CMS in order to upgrade their production flow to the on-line product strategy.

2.5 Classification export

Classification export is typical for on-line productions. Thesauri are regularly updated completely. A thesaurus delivery includes synonyms and all associations. The CU classification is delivered per CU. Each CU uploaded to the web portal has its own classification information.

2.6 Index productions

This is typical for end-of-book publications used in books, loose-leaf and on CD or DVD. Index production is initiated by the publication build system (typically, a sub-system of the CMS, occasionally an independent system). The publishing system establishes a folder with CU-references (see above). Index generation starts from this folder. It generates the default thesaurus view for the CU in the Folder, excluding all terms that are not leading to a classified CU. Subsequently rules are applied to transform this default view. Consequences of the applied rules are:
– Name selection of the thesaurus term (commonly used names, domain specific names);
– Synonym selection for the thesaurus term;
– Term exclusions from the index;
– Index depth selection (indexes typically have a maximum depth);
– Copying and moving of thesaurus branches in the index;
– Finally, when all terms have been decided upon, only RT-RT associations connecting terms in the remaining thesaurus are retained.

2.7 Productivity Tools

Thesaurus editor: Manages thesauri (merge and mapping functions), descriptors (names, synonyms) and associations

Classification editor: Interactive tool for research and corrective action on classified content.

Index profile editor: a tool to edit a generated index. Corrections are saved as rules, applicable in an automated way on subsequent index productions.

Batch job import and export: Interactive and command-line interface to an application to launch batch jobs, parameterized by publication constraints.

Batch job monitoring: the operator tool to follow-up and manage batch job productions. The framework also monitors interactive users on the TMS.

3 Ontology based information modeling

An Ontology Model to meet WKB needs

We are now presenting a quick non-formal review of the ontology used to meet the above presented business needs. A formal presentation is available in separate OWL/RDF documents, developed using the Protégé-OWL ontology editor.

3.1 Thesaurus Ontology model

3.1.1 Context and approach used

The thesaurus ontology model is a formalization of the meta-model implicitly defined by commonly used thesaurus standards, such as ISO 2788:1986[1] and ISO 5963:1985[2]. Although developed in the framework of WKB project, this ontology is not domain specific, and can be applied to any monolingual or multilingual thesaurus built upon the above standards. It has been used since in a variety of other Mondeca projects, involving multilingual resources published by international organisms, such as the Thesaurus on Tourism and Leisure activities (World Tourism Organization), the UNESCO Thesaurus, or the European Environment Thesaurus (GEMET), as well as several domain-specific thesauri for various industrial customers. A preliminary important remark is to stress that the approach used is not to convert the thesaurus structure of concepts into an ontology, where descriptors would be re-factored as classes, and generic-specific relationships as sub classing relationships. Such an approach has been sometimes used in the past, and brought about confusion and misunderstanding between the ontology community and thesaurus community. Fortunately, those communities are now on a more constructive track, as proven by the current W3C SKOS[3] project. It brings about the ontology viewpoint as a meta-level, describing the generic structure of a thesaurus, and independent of the domain and content nature (descriptors). Actually the thesaurus ontology model was built in 2004, when SKOS was still only a by-product of the SWAD-Europe[4] project and not yet on the mainstream W3C Semantic Web Activity track where it is now. Anyway, at that time, SKOS was already closely monitored by Mondeca and its approach qualified as relevant. Hence, the core structure of the thesaurus ontology model is very similar, and can easily be mapped into SKOS core vocabulary.

3.1.2 Core structure

The core structure elements map quite exactly the standard thesaurus constructions and their SKOS representation. They allow a formal representation of any Thesaurus

[1] http://www.collectionscanada.ca/iso/tc46sc9/standard/2788e.htm
[2] http://www.collectionscanada.ca/iso/tc46sc9/standard/5963e.htm
[3] http://www.w3.org/2004/02/skos/
[4] http://www.w3.org/2001/sw/Europe/reports/thes/

in a semantic format, independently of any specific application. They form the declarative part of this module. The core class is “Descriptor”, which maps to the SKOS class “Concept”. The various terms, or lexical forms, used to denote a descriptor, are attached as attributes, either preferred terms or synonyms, with a specification of the language. Abbreviations, definitions, and scope notes are defined the same way. Hierarchy of descriptors is described using a generic “broader-narrower” relationship. This hierarchy has no added semantics regarding the original thesaurus, given its main if not only main purpose to help navigation and search of resources indexed by descriptors, and to be reproduced in index hierarchies. Technical attributes, such as “Top Term”, “Leaf Term” and “Level” can be computed from this structure. For the same purpose, a loose associative relationship type is defined (related terms). A “Thesaurus” class is defined, allowing integration of several thesauri, each declaring its own language, and linked to its various application profiles (see below). The “Thesaurus” class maps quite neatly to the SKOS class “Concept Scheme”.

3.1.3 Functional elements

The functional elements are defined as extensions of the declarative core, to meet specific process requirements. They are not exhaustively described here, some of them are presented to make clear the distinction between declarative and functional ontology. Functional elements linked to the “Descriptor”. “Classifying Term”. This boolean attribute indicates whether or not indexing of resources is allowed directly on the current descriptor. Typically it will be set to “false” for higher levels of the hierarchy. The value of this boolean is used to control the indexing interface. Deprecation mechanism. A “Descriptor” can be changed through the Thesaurus Management interface into a “Deprecated Term”, with attributes “Replaced By” (pointing to a valid descriptor) and “Valid Until”. The resources which used to be indexed on the deprecated term are redirected to the replacing one, along with its lexical forms, which are kept as synonyms. More functional elements are linked to the “Thesaurus” class itself. They are mainly linked to the publication process, and singularly the index generation, which is likely to use the same thesaurus structure or elements in different ways. Those elements are defined in the “Index Generation” module, they can be customized at will, and extended to meet more specific business requirements. They leverage the declarative part of the thesaurus ontology model, which is standard and stable, but are independent of the content of the thesaurus itself. For example, the “Index Profile” class is bearing attributes asserting rules under which thesaurus elements are used and published in a given application context. Thesaurus (one or more) used, and language used (for multilingual thesaurus) Specific subclass of descriptors used (for example if the thesaurus is organized along semantic fields) Levels of the thesaurus which will be used, if one wants to exclude too generic, or too specific descriptors. Indexing attributes used (different types of attributes can be used to index content units against the same thesaurus) Exclude specific descriptors, or at the opposite promote them as top entries. Include or not synonym entries, and/or use specific synonym types (such as abbreviations), or customized synonyms defined in the profile. Include or not associative relations Sort and display options (use the default alphabetical order, or specific sort keys)

3.1.4 Remark on OWL species used

The index profile elements are somehow providing an extra layer of description of the thesaurus structure. Note that this description will make assertions about thesaurus ontology elements, such as “use this or that attribute”, or “use this or that class”. Such declarations have an impact of the characteristics on the ontology, and the species of OWL used: using properties or classes as values of other properties make them to be considered as individuals. For example an “index profile” for the “authors’ table” will declare “dc:creator” as value of the property “indexed attribute”. In the OWL/RDF description, in this situation, “dc:creator” is defined both as property and individual, which makes the ontology OWL-Full. On the other hand, if the ontology is defined by modules, “dc:creator” is an individual only in the index generation module, whereas it is a property in the content unit module. The two descriptions of “dc:creator” correspond to different aspects of the same RDF resource in different application perspectives.

3.2 Reusable CU ontology model

For a publisher a Reusable Content Unit is: a valuable asset, stored in a CMS and qualified with administrative metadata (source, editor, date..) the basic component to build a product which implies to have all the useful attributes describing the Content Unit to build semi automatic process able to select the relevant Content Units for a product publication (validity of the Content Unit, subject, Content Unit language…) the basic content component an end user will be able to access using knowledge based information tools such as book index, taxonomies or faceted search. This implies to manage trough metadata classification links to the thesaurus level. The Content Units management ontology is focus on delivering the proper services to the publisher when building “intelligent” end users products with the best productivity and reactivity. The ontology model for Content Unit reflects those needs: Manage relationships between the Content Unit ontology and the Content Management system: Content Unit Identifier enables to manage the relation between the ontology repository and the CMS where the files are stored. A specific connector is created for each Content Management System to solve the Content Unit Identifier into a file address in the CMS. Enable Content Unit classification on thesaurus using metadata. In our case, depending on the Content Units origin a single French-Dutch legal thesaurus or two different thesaurus, one in French the other one in Dutch are used. Assure Content Unit Ontology independence with Publication ontology. Content Units are used in several publications at the same time. It was decided to keep a strict independence with Publication ontology. Publication ontology is able to relate to the Content Units trough Content Reference, when Content Units are not aware of their use and position in a publication.

3.2.1 Return of experience

The Content Units Ontology is a technical tool for the publisher to manage content units and content unit classification, it should be independent of the product they will be used for and of end user needs for content navigation into the final product. Content Unit Ontology model is the most business dependant part of the ontology model as the attributes (metadata) of Content Units depend on the publisher needs but also on the existing metadata schema. This is not too much of a constraint as the other part of the ontology model: Thesaurus ontology model and Publication ontology model don't depend on Content Units ontology model.

3.3 From Machine centered ontology (Thesaurus and Content Units ontologies) to human centered ontology for book publication

Ontology model for publication needs to describe all the components of a product issue by the publisher: table of content, selected Content Units to include in the product, product indexes enabling reader to access content from an organized list of subjects.

The requirement for the "Publication ontology" was to be generic for any type of product, to be able to produce products in continuity with actual processes, to make the process change invisible for clients and last to support both French and Dutch publications. "Product" class enables to manage descriptive attributes on a product, independently of the periodical publications. Attributes describe the publication media, the product language and links to the set of rules used for index automatic generation "Publication" class describes the container that will link all the publication components for a specific publication. "Publication" has attributes such as publication date, publication manager… and is the binding point for the three components of the publication: Table of Content, selected Content Units and Publication index. Reusable Content Units can be selected for numerous publications, for each publication there is a need to manage specific attributes for the Content Unit, such as a specific name depending on the context in which the CU is included. The answer was to create a class of object "Content Reference" that will be used in the "Publication" and have a link to the Content Unit. The Content Reference will support all specific attributes related to the Content Unit in the publication. Table of Content is the backbone of the publication; it enables to organize the selected Content References into a meaningful product. Ontology model reflects the hierarchical organization of Content References. The need to respect an order between Content references explain the attribute "order" which is recalculated each time a new Content Reference is inserted in a table of content. Index of a book can be considered as a human centered view of a subset of the thesaurus. It is a subset of the thesaurus as it must only list the terms of the thesaurus relevant to the selected Content Units. It inherits from the thesaurus the hierarchical organization of terms (Broader / Narrower terms) but also the non hierarchical relationships (Related terms) and the synonyms. Index is customized for each product. Humans use index searching with their own term in first level index entries. Humans appreciate to have first level index entries organized in alphabetical order and to find the word they are looking for even if the index entries redirect them

to the preferred term. This mean relevant terms placed in branches of the thesaurus should be put at top level in the index, synonyms of first level index entries should be listed in the index first level also, all first level index entries should be listed alphabetically. Also the highest hierarchical levels of the thesaurus may not be relevant to the end user, this means a need to suppress highest level terms or lower level terms, or even to crunch some intermediate level of the thesaurus to make a sound full index for the publication.

Index can than be seen as a reengineered view of the thesaurus for a selected set of Content units. On the ontology model level this implies to build specific classes of objects to model the index: “index entries” instances related both Terms in the thesaurus and the classified Content Units, index entries hierarchical relationships to describe index entries hierarchy. Another requirement for index generation is to build a reusable application able to adapt to Content Unit metadata schemas, to adapt to the publication language and to adapt to the publication media (paper, CD Rom).

Index generation is an ongoing process, done for each publication of the same product: the productivity requirement is to capitalize on a set of general rules for the index generation and on a set of local rules that can be enriched at each index generation process. The ontology models the general rules as attributes of the class “Index” (language choice, number of level of the index, create index entries for synonyms). Local rules are described in the class “Profiles” enabling to memorize specific rules to apply to a term of the thesaurus if it is used in the index of a specific product. The automatic index generation process is a two steps process, first run enable to build a simple index based on the thesaurus structure. This simple index structure is used to visualize and edit local rules to apply during index generation such as: exclude terms, moved the term to the top level of the index, use an other word for the index entry than the one used in the thesaurus…), the second step will generate the final index applying general and local rules. The resulting index is stored in the ontology before being exported using API or XML serialization.

There are several lessons to retain from the Index generation process:

First is this need to build separate ontology models for a machine centered application which role is to manage the complete collection of Content Units and their classification terminology independently of any usage for a human centered usage into a publication made of a selected set of the Content Units, a Table of Content and an Index as knowledge based navigation tool.

Second lesson is the possibility to automatically generate the Human centered ontology from a machine centered ontology using transposition rules and get a result as good as if it was edited manually.

Third lesson is that Human centered knowledge navigation on content depends on the final media: Index is very adapted to books, but navigation based on multiple taxonomies, faceted search and full text searched are very rich tools for Web/Intranet access to contents. As book index, taxonomy navigation, faceted search and rich index can all be issued from the Machine centered ontology: Navigation Taxonomy can be issued from the thesaurus structure, faceted search is based on content unit metadata, and full text search needs semantic extension using terms relationships, terms synonyms, and terms translation from the thesaurus.

3.4 A reusable Ontology Architecture

Aspects of the solution stressed in previous section show the general interest of the approach used, and can be summed up by the following points. These aspects can qualify the system presented as a ontology-driven architecture[5], meaning that architecture of the system and architecture of the ontology are fully integrated.

The ontology model is modular. Each module answers a set of business requirements, generally meeting a consistent set of tasks and possibly users (either humans or systems). Examples of such modules are Thesaurus Management (building, edition, and update), Indexing (using the thesaurus to index resources), and Index Generation (extracting relevant descriptors to build an index). Each module is implemented in specific parts of the information system architecture, and for that purpose every module has declarative elements, and functional elements. The declarative elements describe the business objects independently of their use, in other words what is generally called the domain ontology, whereas the functional elements describe parameters and rules on how the information system handles those objects. Each module is independently re-usable. For example the Thesaurus Management module can be used independently of the Index Generation module.

The same ontology element (RDF resource) can be used in different modules with different semantics, either declarative or functional, which can appear at first sight as inconsistent, but are in reality orthogonal or complementary, as the above quoted "dc:creator" example has shown. Only when merging all the modules does the resulting "OWL-Full-ness" of the ontology appear.

4 Project Status / Results

4.1 Status

The implementation and technical deployment is finished. 8 indexing system have been migrated, including the 5 mayor systems

- 2 mono lingual indexing systems have been migrated into 1 bilingual thesaurus
- 2 bilingual indexing systems have been migrated into 1 bilingual thesaurus
- 4 monolingual indexing systems (two nl, two fr) have been migrated. Cleaning and merging are ongoing.

The migrated indexing systems have resulted in thesauri used to support:

- 2 operational bilingual production sites
- 80 productions on CD end end-of-book

Remaining indexing systems are expected to be fully integrated in one of the above thesauri, either as a specialized sub-thesaurus, or by mapping them into one of the above thesauri.

[5] http://www.w3.org/2001/sw/BestPractices/SE/ODA/060103/

4.2 Costs

Item	ICT Days	ICT Cost (€)	Editorial Days	Editorial Cost (€)	
From book to online	525	348,750	210	92,400	
Software package, Analysis, Implementation, Testing	450	306,000	60	26,400	
Documentation and user guidance	75	42,750	150	66,000	
From indexing systems to one thesaurus	270	183,600	1,000	440,000	
Data conversion	110	74,800			
Initial data-cleaning of thesaurus	60	40,800	500	220,000	
Migration to one thesaurus	100	68,000	500	220,000	
Optimization of the index process performance issues	120	81,600	-200	-88,000	per/year for all indexes
Total	915	613,950	1,010	444,400	

4.2.1 Final remarks

The analysis and testing is done by the ICT department of Wolters Kluwer Belgium. The same department has implemented the ITM software package, fully integrated with two content management systems (one legacy system and Sigmalink of Empolis). The software package can be re-used within Wolters Kluwer world wide with minor changes if they use Sigmalink.

The cost for interfacing with the European internet Platform of Wolters-Kluwer has been included (from book to online). This platform was not yet reliable at the beginning of our TMS project. Setting up a production ready system for fully automated flow took us several man months for two main sites (Human resources and Safety & Environment). Data conversion from the old indexing systems and data-cleaning to one multilingual thesaurus (per country) is always needed. We expect the same cost for every Wolters Kluwer sister. Companies which have already investigated in maintaining one thesaurus will have no editorial cost. Maintenance cost of the old indexing software systems is not included in the above mentioned table.

During the production of huge indexes a performance issue appeared. It took us (Wolters Kluwer and Mondeca) several months to redesign the architecture. Query optimization and multi threading solved the problem.