Call for Expression of Interest: "AMI" OJS N° 2005/S 195-191899
Field : N. 2: Multilingual thesauri
Tender Ref. N.: OPOCE TENDER N°10118
Tender Full Title: EUROVOC Studies - LOT2

Deliverable D2.3

Report on execution and results of the interoperability tests

E. Francesconi, S. Faro, E. Marinai, V. Sandrucci and F. Bargellini, M. Biasiotti, G. Peruginelli
Institute of Legal Theory and Techniques
Italian National Research Council
Florence, Italy

Phase: **PH2 - Interoperability**

Abstract
This document contains the description of the thesaurus interoperability assessment, as regards the tools used to implement the interoperability test as well as the results of the execution of such tests on a meaningful subset of the thesauri of interest.

Version 1.0 , December 7^{th}, 2007

Eurovoc Studies - LOT2
Institute of Legal Theory and Techniques
Italian National Research Council
via de' Barucci 20
50127 Firenze
Italy

tel: +39 055 43995
fax: +39 055 4399605
http://www.ittig.cnr.it

Corresponding author:
Enrico Francesconi
tel: +39-055-4399665
francesconi@ittig.cnr.it
http://www.ittig.cnr.it

Contents

1 Introduction

The methodologies proposed with this project for the interoperability between thesauri (described in Deliverable D1.5) have been specifically assessed on a data sample selected from the thesauri of interest. The environment where to test the proposed methodologies, as well as the steps followed to execute the interoperability assessment, have been described in Deliverable D1.5 and detailed in Deliverable D2.2.

This report contains the results of such tests as well as tools and methodologies used for interoperability assessment.

In particular in Section 2 an overview of the proposed methodologies and the workflow to test them is reported. Then the details of the workflow implementation is described.

Section 3 illustrates a comparative analysis of thesauri native formats with respect to the standards to be used for introperability assessment, as well as the methodologies allowing such an implementation, while in Section 4 the details of the thesaurus mapping methodologies implementation are described. Finally in Section 5 the first results of the interoperability approach are reported.

2 Interoperability workflow

In the second phase of the project (PH2: Interoperbility) the approach for the thesaurus mapping problem, described in D1.5, and the environment to assess such an approach, shown in D2.2, have been implemented.

According to the project specifications, a mapping between EUROVOC and the other thesauri of interest are expected. Other different mappings in fact might not be meaningful since some of them pertains to different domains (ex: GEMET focused on environmental issues, while ETT on learning issues).

Therefore in the proposed mapping strategy, EUROVOC will be acting as a reference, and the mapping strategies will be tested to and from EUROVOC terms.

Nevertheless, this approach allows also to achieve interoperability among the other thesauri having EUROVOC as a mapping pivot. This technique reduces the computational complexity of the problem of multi-thesaurus interoperability (N-to-N mapping), since it works in a framework similar to the one considered for the problem of multilingual translation, where a pivot language is used for mapping. The use of a "pivot" language in a N-language environment allows the reduction of the number of bilingual thesauri from a factor N^2 to a factor N.

The basic mapping methodologies will be applied to *descriptors* within corresponding microthesauri in their *English version* as a pivot language.

The system workflow, sketched also in Fig. 1, consists therefore in the following steps:

1. **SKOS Core transformation of each thesaurus**:
   it will be performed using XSLT techniques able to trasform specific thesauri native formats into RDF SKOS Core;

2. **Thesaurus term pre-processing**:
   thesaurus terms will be normalized so that digit characters and non-alphabetic

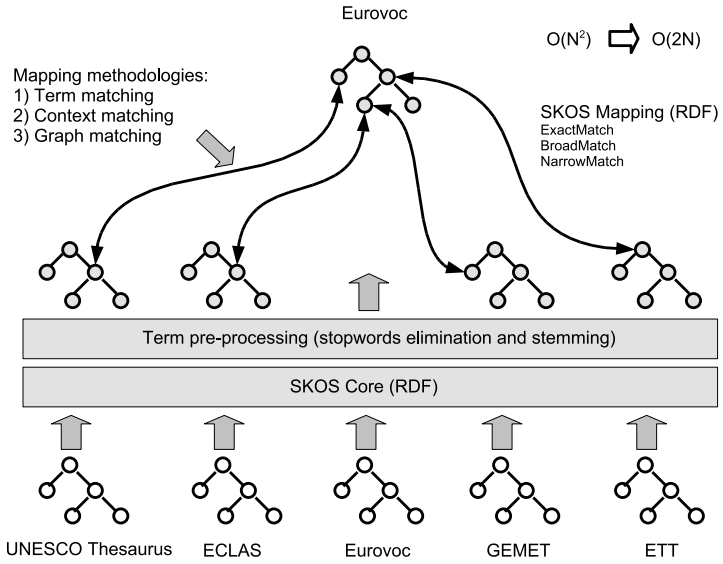

Figure 1: Thesaurus mapping workflow

characters (if any) will be represented by a special character, then other operation as *stemming* or the use of word stoplists (*stopwords elimination*) will be performed;

3. **Thesaurus term representation**:
the three representations of a thesaurus term introduced in D1.5 - Section 5.2.2 will be implemented:

   - *Lexical Manifestation* (string of characters);
   - *Lexical Context* (a vector of weighted terms,composed by the term itself, relevant terms in its definition and linked terms);
   - *Lexical Network* (a graph where nodes are terms along with related ones, and the labeled edges are semantically characterized relations between terms).

   The choice of the term connection degree to be used for a *Lexical Context* or a *Lexical Network* description (descriptors, non-descriptors, micro-thesaurus terms, descriptors definition, etc.) will be one of the aim of the test mapping.

   Some criteria can be given in advance:

   - in case of mapping implemention at the level of fields and microthesauri or corresponding sub-structures (such as domains, chapters or groups), terms representing the related sub-structured will not be inserted in Lexical Contexts or Lexical Networks representation, since this information would be redundant;

- in case of mapping implemention at the level of the whole thesauri, terms representing sub-structures (ex. fields or microthesauri) will provide relevant information and they will be considered for Lexical Contexts or Lexical Networks representation.

4. **Thesaurus term candidate selection for mapping**:
the ranking functions described in D1.5 - Section 5.2.4 will be evaluated; according to the related term representation they are:

   - normalized Levenshtein Distance (for the *Lexical Manifestation*)

     being s_i and s_j two strings to be compared, the *normalized Levenshtein Distance* $dist_{lev}(s_i, s_j)$ is defined as the minimum number of operations (insertion N_{ins}, deletion N_{del}, or substitution N_{sub} of a single character) needed to transform one string into another, normalized with respect to the maximum length of the two strings.

     $$dist_{lev}(s_i, s_j) = \frac{N_{ins} + N_{del} + N_{sub}}{max(|s_i|, |s_j|)}, \quad dist_{lev} \in [0, 1]$$

     The similarity measure in terms of Levenshtein distance can be defined as:

     $$sim_{lev}(s_i, s_j) = 1 - dist_{lev}(s_i, s_j), \quad sim_{lev} \in [0, 1]$$

   - Cosine Distance (for the *Lexical Context*)

     given $\vec{q}$ and $\vec{d_j}$ two *Lexical Contexts* of binary/weighted terms $[w_1, \ldots, w_{|T|}]$ in source and target thesaurus respectively, the *Cosine Distance* measure the similarity sim_{cos} between $\vec{q}$ and $\vec{d_j}$ as:

     $$sim_{cos}(\vec{d_j}, q) = \frac{\vec{d_j} \times \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|}, \quad sim_{cos} \in [0, 1]$$

     where $|\vec{d_j}|$ and $|\vec{q}|$ are the norms of the vectors representing terms in target and source thesauri.

   - Graph Edit Distance (or variants, for the *Lexical Network*)

     being a *Lexical Network* a direct graph $g = (V, E, \alpha, \beta, L_V, L_E)$ the similarity between Lexical Networks is reduced to a problem of *graph isomorphism* to be measured by the *Graph Edit Distance*, namely the minumum number of node deletions/insertions/substitutions and edge deletions/insertions/substitutions to transform a graph g_1 into a graph g_2
     In particular for problems of computational tractability, in the experiments we have used two alternatives to the Graph Edit Distance: *conceptual similarity* and the *relational similarity*.

The *Conceptual similarity* s_c expresses how many concepts the two graphs g_1 and g_2 have in common:

$$s_c = \frac{2n(g_c)}{n(g_1) + n(g_2)}$$

The *Relational similarity* s_r indicates how similar the relations between the same concepts in both graphs are:

$$s_r = \frac{2m(g_c)}{m_{g_c}(g_1) + m_{g_c}(g_2)}$$

Considering a graph g to be matched with a graph g_T as reference, a possible similarity measure can be [1]:

$$sim_{graph} = \frac{N_c(g, g_T) + E_c(g, g_T)}{N(g_T) + E(g_T)}$$

where

- $N_c(g, g_T)$ is the number of nodes shared by graph g and g_T
- $E_c(g, g_T)$ is the number of edges common to g and g_T
- $N(g_T)$ is the number of nodes in graph g_T
- $E(g_T)$ the number is of edges in g_T

5. **Ranking among candidate terms and mapping implementation**: terms of the target thesaurus, represented according to one of the foreseen models, will be matched with the chosen term in a source thesaurus, represented with the same model, using a proper similarity measure.

   Candidates terms of the target thesaurus will be ranked according to the similarity values $sim \in [0, 1]$ and the semantics to the mapping relation will be assigned using proper heuristic threshold values (T_1, $T_2 \in [0, 1]$)

if	$sim < T_1$	$\Rightarrow$	No Match
if	$T_1 < sim < T_2$	$\Rightarrow$	partial match (broadMatch or narrowMatch)
if	$sim > T_2$	$\Rightarrow$	exactMatch

   The problem of predicting the type of partial match in terms of broadMatch or narrowMatch has been be solved in different fashions, depending on the model used for term semantics representation:

   - *Lexical Manifestation*: the longest string between the two ones taken into account for comparison has been assumed as the more specific one;
   - *Lexical Context*: the term represented by a vector containing the highest number of non-zero entries (terms of the vocabulary used to describe the term context) is assumed as the more specific between the two taken into account for comparison;
   - *Lexical Network*: the term represented by a graph containing the highest number of nodes and edges is assumed as the more specific between the two taken into account for comparison.

Alternatives for the mapping strategies are: 1) mapping at the level of the whole thesauri; 2) mapping at the level of sub-structures (corresponding fields, microthesauri, etc.). The basic assumption is that the methods for mapping may give better results on semantically correlated sub-domains.

6. **Representation of the semantics of mapping in SKOS Mapping**: the established relations between thesaurus terms will be described and stored using RDF SKOS Mapping standard.

In the follwing paragraphs the details of the workflow steps are described.

3 Thesaurus SKOS standard trasformation

3.1 Analysis of SKOS standard for thesauri trasformation and mapping

In order to implement tools able to transform thesauri native formats into RDF-SKOS Core, as well as to describe thesauri mapping relations an analysis has been done on the SKOS Core and SKOS Mapping specifications. UML diagrams of the SKOS standard specifications have been sketched. They have been used as a design support to implement the XSL stylesheets able to convert thesauri property XML formats into the corresponding SKOS Core representation, as well as to implement mapping relations in the phase of "gold standard" creation and automatic mapping execution.

In the following figures (Figs. 2 to 8) such UML diagram analysis is reported.

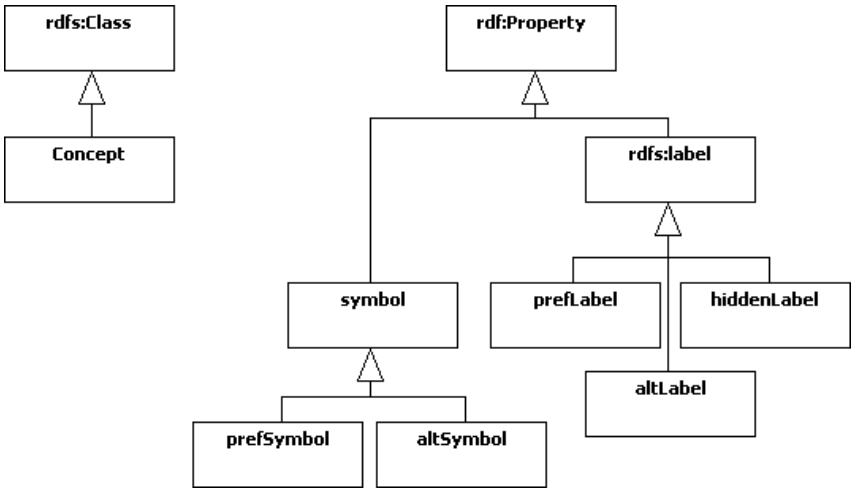

Figure 2: SKOS Core

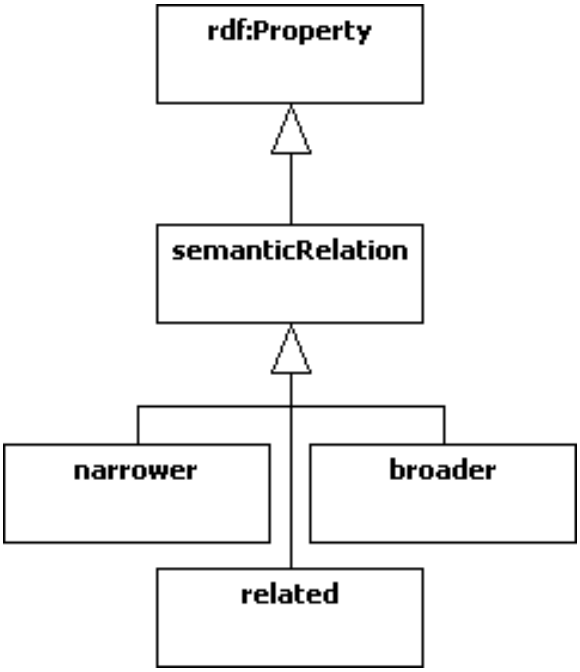

Figure 3: SKOS Core: Relations (properties)

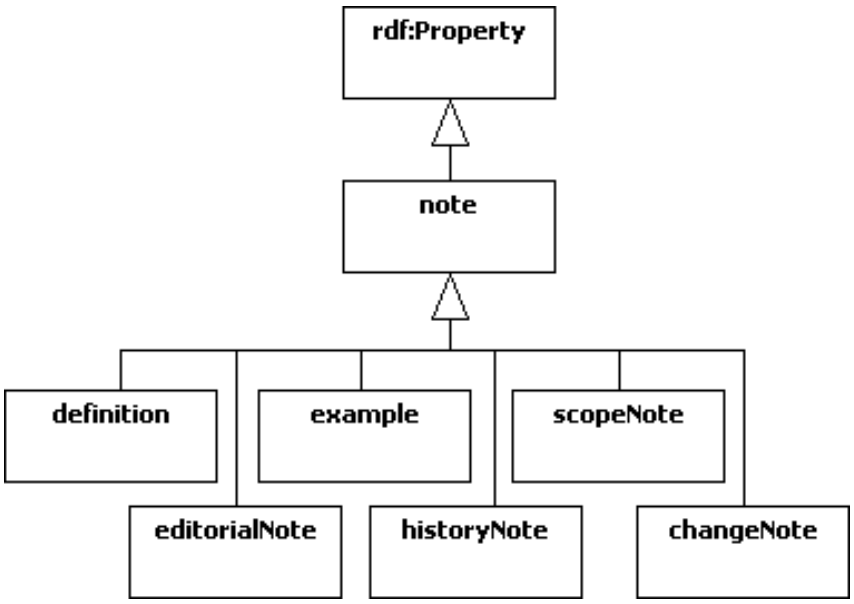

Figure 4: SKOS Core: Notes (properties)

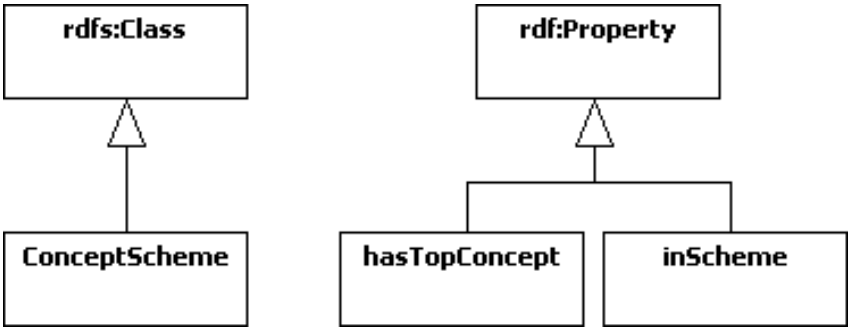

Figure 5: SKOS Core: Concept Schemes (classes and properties)

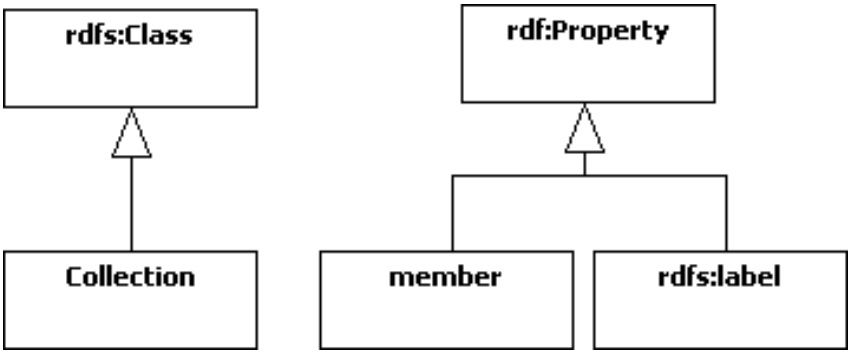

Figure 6: SKOS Core: Collections (classes and properties)

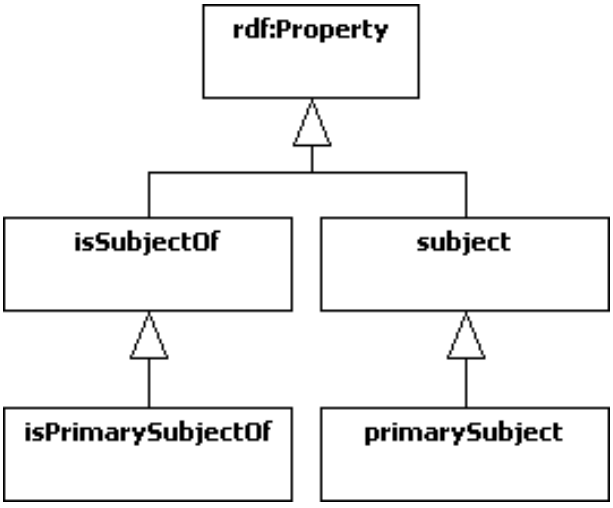

Figure 7: SKOS Core: Subject Indexing (classes and properties)

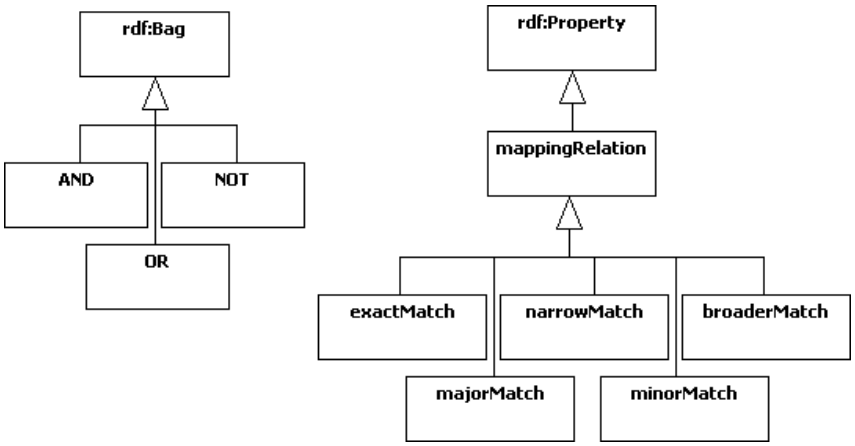

Figure 8: SKOS Mapping

3.2 Analysis of thesauri native formats for SKOS Core transformation

According to the project specifications, five thesauri have to be taken into account (Fig. 9).

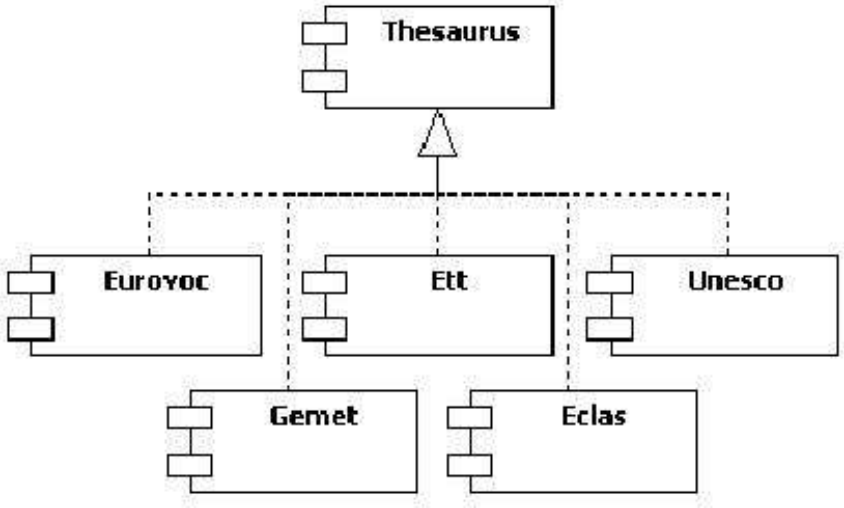

Figure 9

They have been provided in XML proprietary formats, specified by a DTD (as for EUROVOC) or by a specific documentation (as for UNESCO Thesaurus, Eclas and ETT). Only GEMET has been provided in RDF-SKOS native format and it does not need any convertion.

In order to implement the converters for SKOS Core transformation, an analysis of the native XML formats has been carried out in order to establish correspondences between native XML elements of the thesauri of interest and elements of RDF-SKOS Core.

Tabs. 1 to 4 report the tables of correspondences between elements of the source thesauri and their description in SKOS Core format (except GEMET which has been given in SKOS Core native format).

XML elements	**SKOS Core**
DOMAINE_ID	skos:Collection
THESAURUS_ID	skos:ConceptScheme
DESCRIPTEUR_ID	skos:Concept
LIBELLE	skos:prefLabel
USED_FOR	skos:altLabel
PERMUTATION (/PERM/PERM_EL)	skos:altLabel
SCOPE_NOTE	skos:scopeNote
RELATION_BT	skos:broader
RELATION_RT	skos:related

Table 1: Correspondences between EUROVOC XML native format and SKOS Core format.

XML elements	**SKOS Core**
DATABASE_UNESEN/RECORD/MT	skos:ConceptScheme
MT	skos:inScheme
RECORD	skos:Concept
Term, Terme, Término	skos:prefLabel
UF, EP, UP, USE, EMP	skos:altLabel
SN, NE, NA	skos:scopeNote
BT, TG	skos:broader
NT, TS, TE	skos:narrower
RT, TA, TR	skos:related

Table 2: Correspondences between Unesco Thesaurus XML native format and SKOS Core format.

XML elements	**SKOS Core**
eclas:dataField[@tag='150']	skos:prefLabel
eclas:dataField[@tag='450']	skos:altLabel
eclas:dataField[@tag='550'] and eclas:subfield[@code='w'] = 'g'	skos:broader
eclas:dataField[@tag='550'] and eclas:subfield[@code='w'] = 'h'	skos:narrower
eclas:dataField[@tag='680']	skos:scopeNote
eclas:record and eclas:subfield[@code='065']	skos:Concept

Table 3: Correspondences between ECLAS XML native format and SKOS Core format.

XML elements	SKOS Core
/THESAURUS/CONCEPT	skos:Concept
/THESAURUS/{ENG, DAN, DUT, EST, FIN, FRE, ITA}	skos:prefLabel
/THESAURUS/STA	skos:editorialNote
/THESAURUS/UPD	skos:changeNote
/THESAURUS/SN	skos:scopeNote
/THESAURUS/UF	skos:altLabel
/THESAURUS/BT	skos:broader
/THESAURUS/NT	skos:narrower
/THESAURUS/RT	skos:related

Table 4: Correspondences between ETT XML native format and SKOS Core format.

3.3 Thesauri SKOS Core XSLT trasformation

As discussed in D1.5 and D2.2, to deal with a uniform representation of the thesauri of interest an RDF-SKOS Core trasformation of each one has been carried out. The chosen solution makes use of XSL stylesheets along with XSLT technologies to trasform thesauri of interest from proprietary XML formats to RDF-SKOS Core format.

In Fig. 10 is sketched the generic use-case designing the procedure (`m2skos`) for thesauri transformation.

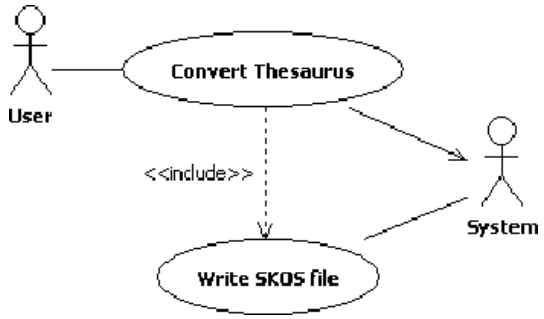

Figure 10: Use case of the `m2skos` trasformation.

In particular in Fig. 10 are represented the following entities:

- **Primary Actor**: User
- **Level**: User Goal
- **Scope**. m2skos application

Main Success Scenario:

1. User specifies a file to be converted, its format and the name of the output
2. System executes the file parsing and writes the output file in SKOS Core format

Fig. 11 outlines the scenario to be implemented for the thesauri SKOS conversion on a Windows platform (a similar scenario can be foreseen on Linux platform, the only differences are related to the language used to implement the batch script and the trasformer to be used ("xsltproc" on Windows platform).

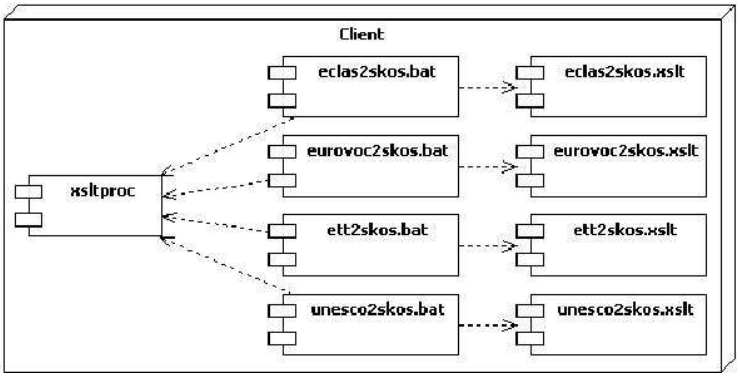

Figure 11: m2skos

4 Thesaurus mapping algorithms implementation

Thesaurus mapping algorithms have been implemented in a Java environment using the available API oriented to semantic Web standards management, as well as libraries able to handle linguistic pre-proceesing functionalities.

4.1 The RDF-SKOS standard management

RDF-SKOS standard has been managed using Jena[1], a Java framework for building Semantic Web applications. It provides API to handle RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine.

Jena is an open source library including:

- A RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- SPARQL query engine

4.2 Term pre-processing and ranking functions implementation

According to the proposed algorithms, a number of pre-processing steps were expected on thesaurus terms to increase the statistical quality of terms and to reduce the computational complexity of the problem.

[1] `http://jena.sourceforge.net/`

In particular the following pre-processing steps have been carried out:

- digit characters can be represented using a special character;
- non alphanumeric characters can be represented using a special character
- *Stemming* has been applied on pure words to reduce them to their morphological root
- use of word stoplists (*stopwords elimination*) to eliminate very common terms which do contribute to the semantics of complex terms.

In order to implement such functionalities the Apache Lucene [2] text search engine library, written in Java, has been used.

Apache Lucene is an Apache project implementing a high-performance, full-featured text search engine library written entirely in Java. This technologies has been designed and implemented to develop application that requires full-text cross-platform search. It is an open source project available for free download and it offers powerful features through a simple API:

- ranked searching – best results returned first
- many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
- fielded searching (e.g., title, author, contents)
- date-range searching
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching

Apache Lucene has been used also because it provides an API to implement some of the similarity measures as well as ranking functions implementation of the proposed methodologies to establish automatic thesaurus term matching. In particular the Apache Lucene library provides primitives able to implement

- Mapping between term *Lexical Manifestations* (Levenshtein Distance)
- Mapping between term *Lexical Contexts* (Cosine Distance)

Firstly Apache Lucene provides primitives to measure the distance between terms using the Levenshtein Distance.

Morever it is able to construct a vocabulary of term from a given collection of documents. According to the isomorphism established in our approach between the Thesaurus Mapping ($\mathcal{TM}$) and the Information Retrieval $\mathcal{IR}$ problems (see

[2] `http://lucene.apache.org`

Deliverable D1.5) this is obtained once we look at terms (simple or complex) (better the *semantic* of terms) as documents and a thesaurus as a collection of documents. Therefore the Apache Lucene APIs can be effectively used to privide a representation of thesaurus terms Lexical Contexts (a vector of binary or weighted terms) and to measure their distance using the Cosine Distance.

In order to implement the mapping methodology proposed associated to Lexical Network representations of the term semantics (basically graphs of terms and their relations), the JGraph library has been used.

JGraph is an open source graph visualization library written in Java. It is fully Swing compatible component, both visually and in its design architecture. It provides a range of graph drawing functionality for client-side or server-side applications. Moreover it has a powerful API enabling you to visualize, interact with, automatically layout and perform analysis of graphs.

Example applications for a graph visualization library include process diagrams, workflow WWW visualization, networks displays, mapping applications. JGraph, through it's programming API, provides the means to configure how the graph or network is displayed and the means to associate a context or metadata with those displayed elements.

Such API has been used to deal with graph in order to impement the similarity and ranking function for *Lexical Networks* mapping (Graph Edit Distance).

In Figs. 12 to 16 the design, in terms of UML diagrams, of the application to implement automatic mapping assessment have been reported.

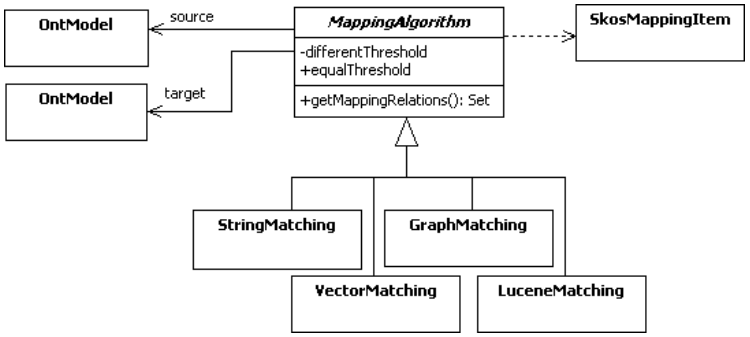

Figure 12

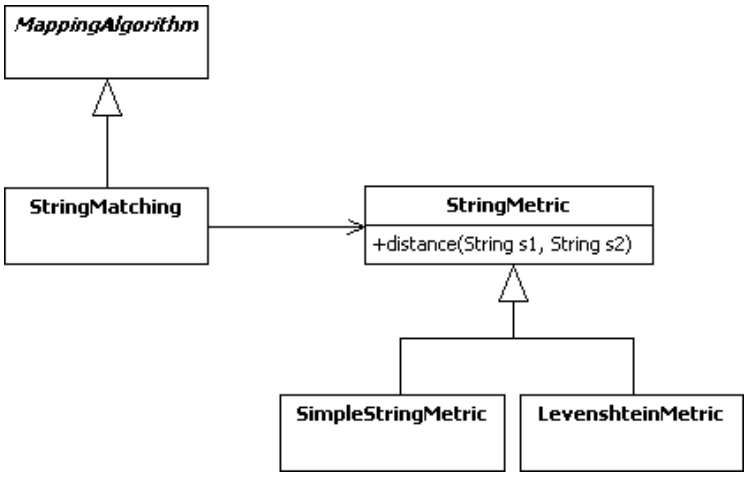

Figure 13

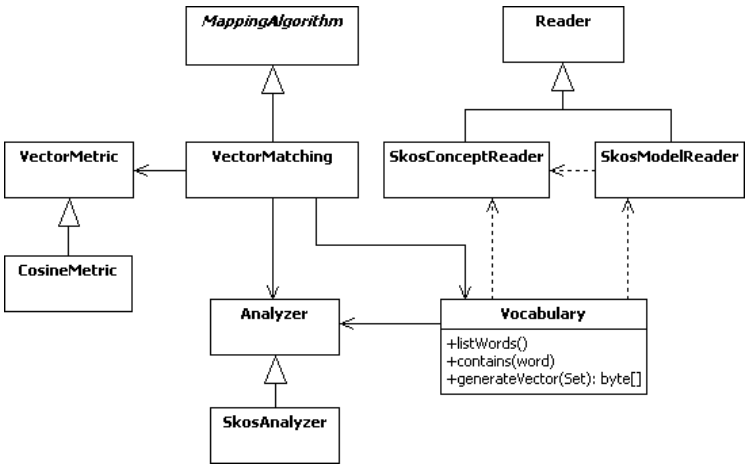

Figure 14

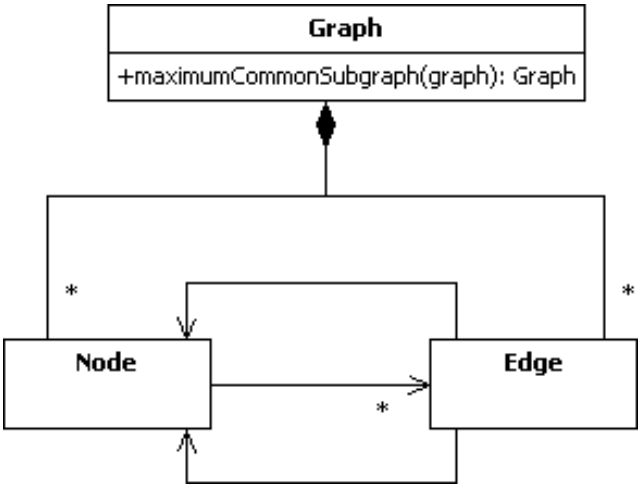

Figure 15

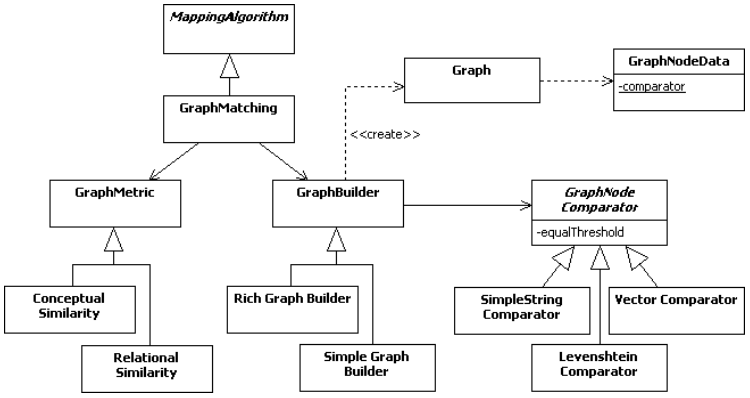

Figure 16

5 Thesaurus interoperability assessment

The interoperability approach proposed has been assessed on a "gold standard" composed by a different number of mapping relations beetween EUROVOC and the other thesauri of interest for the project. In Tab. 5 the number of relations contained in the "gold standard" of ETT and EUROVOC as pivot is reported. This preliminary "gold standard" has been used to produce a very first assessment of the algorithm and to tune the threshold values.

Such a "gold standard" mapping relations have been created by two groups of experts, each one dealing with one of the two domains ("Law" and "Employment and Working conditions") chosen to assess the interoperability approach. The experts used the THALEN (THesaurus ALigning ENvironment (see Deliverable D2.2)) application developed within the project and produced a "gold standard" data set in

Relation type	Gold-standard relations number
exactMatch	71
broadMatch	15
narrowMatch	18
Total	104

Table 5: Number of "gold standard" relations established between EUROVOC and ETT

SKOS mapping format, containing mapping relations between each couple of thesauri having EUROVOC as pivot.

The automatic mapping algorithms are under test on such a "gold standard" data set, giving results within each domains chosen for experiments, as well as for the whole data set.

The proposed approach has been applied according to the typical *divide et impera* criterion. The proposed thesaurus mapping solutions will be firstly implemented at the level of fields and microthesauri or corresponding sub-structures (such as domains, chapters or groups). The basic assumption is that the methods for mapping, previously described and proposed for this study, give better results on semantically correlated sub-domains. Each field or microthesaurus is identified by a term or group of terms which are a lexical representation of a concept and a specific domain of interest.

For a *Lexical Context* or a *Lexical Network* term representation, we have used the set of related terms in an *adjacency* relation to the given one, namely terms, as well as the associated information, connected with only one edge.

5.1 Global performances

Taking into account the isomorphism established between the Information Retrieval problem ($\mathcal{IR}$) and the Thesaurus Mapping problem ($\mathcal{TM}$) (see Deliverable D1.5), a more general view of the automatic mapping quality can be also given using the well-known $\mathcal{IR}$ measure of `Precision` and `Recall`, both for a specific mapping relation and considering average values of `Precision` and `Recall` with respect to different mapping relations.

Given two thesauri and the related "gold standard" including the following SKOS Mapping types of relations $\mathcal{C}$ = {exactMatch, broadMatch, narrowMatch}:

- let N_i, $i \in \mathcal{C}$ the number of the exactMatch, broadMacth or narrowMatch relations *established by experts* in the "gold standard";
- let N_i^t, $i \in \mathcal{C}$ the number of correct exactMatch, broadMatch or narrowMatch relations *predicted by the system*;
- let N_i^f, $i \in \mathcal{C}$ the number of wrong exactMatch, broadMatch or narrowMatch relations *predicted by the system.*

the performances of the mapping methodologies will be measured in terms of

$$Precision_i = \frac{N_i^t}{N_i^t + N_i^f} \qquad Recall_i = \frac{N_i^t}{N_i}$$

where i represents the semantic label of a SKOS Mapping relation ($i \in \mathcal{C}$).

The global performances of the mapping system may be evaluated in two different ways [2]:

- *microeveraging*: precision and recall are obtained by summing over all individual decisions:

$$Precision^{\mu} = \frac{\sum_{i}^{|\mathcal{C}|} N_i^t}{\sum_{i}^{|\mathcal{C}|}(N_i^t + N_i^f)} \qquad Recall^{\mu} = \frac{\sum_{i}^{|\mathcal{C}|} N_i^t}{\sum_{i}^{|\mathcal{C}|} N_i}$$

  where μ indicates microaveraging;

- *macroaveraging*: precision and recall are first evaluated "locally" for each SKOS Mapping relation type, and then "globally" by averaging over the results of the different relation types:

$$Precision^{M} = \frac{\sum_{i}^{|\mathcal{C}|} Precision_i}{|\mathcal{C}|} \qquad Recall^{M} = \frac{\sum_{i}^{|\mathcal{C}|} Recall_i}{|\mathcal{C}|}$$

  where M indicates macroaveraging.

The first set of experiments raised some issues affecting the validity of results and the computational complexity of the algorithms:

1. in the most part of the mapping predictions an association between a source descriptor and a target non-descritor is proposed; this has been proved as a valid automatic mapping results, but the "gold standard" has been built with a different criterion (relationship between descriptors only);

2. in particular the implementation of the algorithms dealing with graph matching have been proved time consuming;

3. algorithm results have been proved as particularly sensitive to threshold tuning, therefore a set of experiments aiming to establish optimal threshold values for automatic mapping in each of the algorithm frameworks are under execution.

Therefore, specific actions are under development to modify the algorithms and to validate the "gold standard" in order to solve these problems and produce the final results.

A XSL stylesheets for thesauri SKOS Core trasformation

In this appendix the XSL stylesheets developed for thesauri trasformation from XML native format into SKOS Core format are reported. According to Fig. 11 they represent components (`[thesaurus_name]2skos.xslt`) of the `m2skos` procedure, which use `xsltproc` as a trasformer on Windows plaform, and `[thesaurus_name]2skos.bat` as scripts to run the trasformations.

A.1 eurovoc2skos.xslt

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  version="1.0">

<xsl:output method="xml" indent="yes" />

<xsl:variable name="base-uri">http://www.ittig.it</xsl:variable>
<xsl:variable name="domain-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/domain#</xsl:text>
</xsl:variable>
<xsl:variable name="thesaurus-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/micro-thesaurus#</xsl:text>
</xsl:variable>
<xsl:variable name="concept-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/concept#</xsl:text>
</xsl:variable>

<xsl:key name="domain-id" match="/tag/tag/DOMAINES/RECORD/DOMAINE_ID" use="."/>
<xsl:key name="thesaurus-id" match="/tag/tag/THESAURUS/RECORD/THESAURUS_ID" use="."/>
<xsl:key name="concept-id" match="/tag/tag/DESCRIPTEUR/RECORD/DESCRIPTEUR_ID" use="."/>

<xsl:template match="/">
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<xsl:call-template name="print-domain" />
<xsl:call-template name="print-thesaurus" />
<xsl:call-template name="print-concepts" />

</rdf:RDF>

</xsl:template>


<xsl:template name="print-domain">

<skos:Collection>
   <rdfs:label>Eurovoc Thesaurus</rdfs:label>

<xsl:for-each
select="/tag/tag/DOMAINES/RECORD/DOMAINE_ID[generate-id()=generate-id(key('domain-id',.))]">
```

```
<xsl:variable name="dom-id1" select="."/>
     <skos:member rdf:resource="{$domain-base-uri}{$dom-id1}"/>
</xsl:for-each>

     </skos:Collection>

<xsl:for-each
select="/tag/tag/DOMAINES/RECORD/DOMAINE_ID[generate-id()=generate-id(key('domain-id',.))]">
<xsl:variable name="dom-id2" select="."/>
<skos:Collection rdf:about="{$domain-base-uri}{$dom-id2}">

<xsl:for-each select="/tag/tag/DOMAINES">
<xsl:variable name="lng1" select="@LNG" />
<xsl:for-each select="./RECORD" >
<xsl:variable name="dom-id3" select="./DOMAINE_ID"/>
<xsl:variable name="label1" select="./LIBELLE" />

<xsl:if test="$dom-id2 = $dom-id3">
<rdfs:label xml:lang="{$lng1}" >
<xsl:value-of select="$label1" />
</rdfs:label>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each
select="/tag/tag/THESAURUS/RECORD/THESAURUS_ID[generate-id()=generate-id(key('thesaurus-id',.))]">
<xsl:variable name="thes-id1" select="."/>
<xsl:if test="$dom-id2 = substring( $thes-id1, 1, 2 )">
     <skos:member rdf:resource="{$thesaurus-base-uri}{$thes-id1}"/>
     </xsl:if>
</xsl:for-each>

     </skos:Collection>
</xsl:for-each>

</xsl:template>



<xsl:template name="print-thesaurus">
<xsl:for-each
 select="/tag/tag/THESAURUS/RECORD/THESAURUS_ID[generate-id()=generate-id(key('thesaurus-id',))]">
<xsl:variable name="thes-id2" select="."/>
<skos:ConceptScheme rdf:about="{$thesaurus-base-uri}{$thes-id2}">
<xsl:for-each select="/tag/tag/THESAURUS">
<xsl:variable name="lng2" select="@LNG" />
<xsl:for-each select="./RECORD" >
<xsl:variable name="thes-id3" select="./THESAURUS_ID"/>
<xsl:variable name="label2" select="./LIBELLE" />

<xsl:if test="$thes-id2 = $thes-id3">
<rdfs:label xml:lang="{$lng2}" >
<xsl:value-of select="$label2" />
</rdfs:label>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DESCRIPTEUR_THESAURUS/RECORD">
<xsl:variable name="thes-id4" select="./THESAURUS_ID"/>
<xsl:variable name="cnp-id1" select="./DESCRIPTEUR_ID" />
<xsl:if test="$thes-id2 = $thes-id4 and ./TOPTERM='O'">
<skos:hasTopConcept rdf:resource="{$concept-base-uri}{$cnp-id1}"/>
</xsl:if>
```

```
</xsl:for-each>

     </skos:ConceptScheme>
</xsl:for-each>
</xsl:template>



<xsl:template name="print-concepts">
<xsl:for-each
select="/tag/tag/DESCRIPTEUR/RECORD/DESCRIPTEUR_ID[generate-id()=generate-id(key('concept-id',.))]">
<xsl:variable name="cnp-id2" select="."/>
<skos:Concept rdf:about="{$concept-base-uri}{$cnp-id2}">

<xsl:for-each select="/tag/tag/DESCRIPTEUR">
<xsl:variable name="lng3" select="@LNG" />
<xsl:for-each select="./RECORD" >
<xsl:variable name="cnp-id3" select="./DESCRIPTEUR_ID"/>
<xsl:variable name="label3" select="./LIBELLE" />

<xsl:if test="$cnp-id2 = $cnp-id3">
<skos:prefLabel xml:lang="{$lng3}" >
<xsl:value-of select="$label3" />
</skos:prefLabel>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DESCRIPTEUR_THESAURUS/RECORD">
<xsl:variable name="thes-id5" select="./THESAURUS_ID"/>
<xsl:variable name="cnp-id4" select="./DESCRIPTEUR_ID" />
<xsl:if test="$cnp-id2 = $cnp-id4">
<skos:inScheme rdf:resource="{$thesaurus-base-uri}{$thes-id5}"/>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/tag/tag/USED_FOR">
<xsl:variable name="lng4" select="@LNG" />
<xsl:for-each select="./RECORD" >
<xsl:variable name="cnp-id5" select="./DESCRIPTEUR_ID"/>

<xsl:if test="$cnp-id2 = $cnp-id5">
<xsl:for-each select="./UF/UF_EL">
<skos:altLabel xml:lang="{$lng4}" >
<xsl:value-of select="." />
</skos:altLabel>
</xsl:for-each>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each select="/tag/tag/PERMUTATION">
<xsl:variable name="lng5" select="@LNG" />
<xsl:for-each select="./RECORD" >
<xsl:variable name="cnp-id6" select="./DESCRIPTEUR_ID"/>

<xsl:if test="$cnp-id2 = $cnp-id6">
<xsl:for-each select="./PERM/PERM_EL">
<skos:altLabel xml:lang="{$lng5}" >
<xsl:value-of select="." />
</skos:altLabel>
</xsl:for-each>
</xsl:if>
</xsl:for-each>
</xsl:for-each>
```

```
<xsl:for-each select="/tag/tag/SCOPE_NOTE">
<xsl:variable name="lng6" select="@LNG" />
<xsl:for-each select="./RECORD" >
<xsl:variable name="cnp-id7" select="./DESCRIPTEUR_ID"/>

<xsl:if test="$cnp-id2 = $cnp-id7">
<xsl:for-each select="./SN">
<skos:scopeNote xml:lang="{$lng6}" >
<xsl:value-of select="." />
</skos:scopeNote>
</xsl:for-each>
</xsl:if>
</xsl:for-each>
</xsl:for-each>

<xsl:for-each select="/tag/tag/RELATIONS_BT/RECORD">
<xsl:variable name="cnp-id8" select="./SOURCE_ID"/>
<xsl:variable name="cnp-id9" select="./CIBLE_ID"/>

<xsl:if test="$cnp-id2 = $cnp-id8">
<skos:broader rdf:resource="{$concept-base-uri}{$cnp-id9}"/>
</xsl:if>

<xsl:if test="$cnp-id2 = $cnp-id9">
<skos:narrower rdf:resource="{$concept-base-uri}{$cnp-id8}"/>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/tag/tag/RELATIONS_RT/RECORD">
<xsl:variable name="cnp-id10" select="./DESCRIPTEUR1_ID"/>
<xsl:variable name="cnp-id11" select="./DESCRIPTEUR2_ID"/>

<xsl:if test="$cnp-id2 = $cnp-id10">
<skos:related rdf:resource="{$concept-base-uri}{$cnp-id11}"/>
</xsl:if>

<xsl:if test="$cnp-id2 = $cnp-id11">
<skos:related rdf:resource="{$concept-base-uri}{$cnp-id10}"/>
</xsl:if>
</xsl:for-each>

     </skos:Concept>
</xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

A.2 unesco2skos.xslt

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  version="1.0">

<xsl:output method="xml" indent="yes" />
```

```
<xsl:variable name="base-uri">http://www.ittig.it</xsl:variable>
<xsl:variable name="domain-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/domain#</xsl:text>
</xsl:variable>
<xsl:variable name="thesaurus-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/micro-thesaurus#</xsl:text>
</xsl:variable>
<xsl:variable name="concept-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/concept#</xsl:text>
</xsl:variable>


<xsl:key name="then-id" match="/tag/tag/DATABASE_UNESEN/RECORD/MT" use="."/>
<xsl:key name="thfr-id" match="/tag/tag/DATABASE_UNESFR/RECORD/MT" use="."/>
<xsl:key name="thsp-id" match="/tag/tag/DATABASE_UNESSP/RECORD/MT" use="."/>
<xsl:key name="thru-id" match="/tag/tag/DATABASE_UNESRU/RECORD/MT" use="."/>


<xsl:template match="/">
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<xsl:call-template name="print-thesaurus" />
<xsl:call-template name="print-concepts" />
</rdf:RDF>

</xsl:template>



<xsl:template name="print-thesaurus">
<skos:Collection>
   <rdfs:label>Unesco Thesaurus</rdfs:label>

   <skos:member rdf:resource="{$thesaurus-base-uri}t1"/>
   <skos:member rdf:resource="{$thesaurus-base-uri}t2"/>
   <skos:member rdf:resource="{$thesaurus-base-uri}t3"/>
   <skos:member rdf:resource="{$thesaurus-base-uri}t4"/>
   <skos:member rdf:resource="{$thesaurus-base-uri}t5"/>
   <skos:member rdf:resource="{$thesaurus-base-uri}t6"/>
   <skos:member rdf:resource="{$thesaurus-base-uri}t7"/>
     </skos:Collection>

<skos:Collection rdf:about="{$thesaurus-base-uri}t1">
<rdfs:label xml:lang="EN">1. Education</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '1'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
</skos:Collection>
<skos:Collection rdf:about="{$thesaurus-base-uri}t2">
<rdfs:label xml:lang="EN">2. Science</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '2'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
```

```
</skos:Collection>
<skos:Collection rdf:about="{$thesaurus-base-uri}t3">
<rdfs:label xml:lang="EN">3. Culture</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '3'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
</skos:Collection>
<skos:Collection rdf:about="{$thesaurus-base-uri}t4">
<rdfs:label xml:lang="EN">4. Social and human sciences</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '4'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
</skos:Collection>
<skos:Collection rdf:about="{$thesaurus-base-uri}t5">
<rdfs:label xml:lang="EN">5. Information and communication</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '5'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
</skos:Collection>
<skos:Collection rdf:about="{$thesaurus-base-uri}t6">
<rdfs:label xml:lang="EN">6. Politics, law and economics</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '6'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
</skos:Collection>
<skos:Collection rdf:about="{$thesaurus-base-uri}t7">
<rdfs:label xml:lang="EN">7. Countries and country groupings</rdfs:label>
<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
     <xsl:if test="substring(., 1, 1 ) = '7'">
     <skos:member rdf:resource="{$thesaurus-base-uri}t{substring(., 1, 4 )}"/>
     </xsl:if>
</xsl:for-each>
</skos:Collection>


<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
<xsl:sort select="." />
<xsl:variable name="thes-id1" select="."/>

<xsl:if test="string-length( $thes-id1 ) > 0">
<skos:ConceptScheme rdf:about="{$thesaurus-base-uri}t{substring($thes-id1, 1, 4 )}">
<xsl:call-template name="print-thesaurus-labels">
<xsl:with-param name="thes-id2" select="$thes-id1"/>
</xsl:call-template>

<xsl:call-template name="print-thesaurus-top-terms">
<xsl:with-param name="thes-id3" select="$thes-id1"/>
</xsl:call-template>
</skos:ConceptScheme>
     </xsl:if>
</xsl:for-each>
```

```
</xsl:template>


<xsl:template name="print-thesaurus-labels">
<xsl:param name="thes-id2" />

<xsl:for-each
select="/tag/tag/DATABASE_UNESEN/RECORD/MT[generate-id()=generate-id(key('then-id',.))]">
<xsl:if test="substring( $thes-id2, 1, 4 ) = substring( ., 1, 4 )">
<rdfs:label xml:lang="EN" >
<xsl:value-of select="." />
</rdfs:label>
</xsl:if>
</xsl:for-each>

<xsl:for-each
select="/tag/tag/DATABASE_UNESFR/RECORD/MT[generate-id()=generate-id(key('thfr-id',.))]">
<xsl:if test="substring( $thes-id2, 1, 4 ) = substring( ., 1, 4 )">
<rdfs:label xml:lang="FR" >
<xsl:value-of select="." />
</rdfs:label>
</xsl:if>
</xsl:for-each>

<xsl:for-each
select="/tag/tag/DATABASE_UNESSP/RECORD/MT[generate-id()=generate-id(key('thsp-id',.))]">
<xsl:if test="substring( $thes-id2, 1, 4 ) = substring( ., 1, 4 )">
<rdfs:label xml:lang="ES" >
<xsl:value-of select="." />
</rdfs:label>
</xsl:if>
</xsl:for-each>

<xsl:for-each
select="/tag/tag/DATABASE_UNESRU/RECORD/MT[generate-id()=generate-id(key('thru-id',.))]">
<xsl:if test="substring( $thes-id2, 1, 4 ) = substring( ., 1, 4 )">
<rdfs:label xml:lang="RU" >
<xsl:value-of select="." />
</rdfs:label>
</xsl:if>
</xsl:for-each>
</xsl:template>

<xsl:template name="print-thesaurus-top-terms">
<xsl:param name="thes-id3" />

<xsl:for-each select="/tag/tag/DATABASE_UNESEN/RECORD">
<xsl:if
test="substring( $thes-id3, 1, 4 ) = substring( ./MT, 1, 4 ) and not( ./BT ) and not( ./USE )">
<skos:hasTopConcept rdf:resource="{$concept-base-uri}{./Term}"/>
</xsl:if>
</xsl:for-each>
</xsl:template>


<xsl:template name="print-concepts">
<xsl:for-each select="/tag/tag/DATABASE_UNESEN/RECORD">
<xsl:variable name="cnp-lbl" select="./Term" />
<xsl:variable name="cnp-lbl-fr" select="./FR" />
<xsl:variable name="cnp-lbl-sp" select="./SP" />

<xsl:if test="not( ./USE )">
<skos:Concept rdf:about="{$concept-base-uri}{$cnp-lbl}">
<xsl:if test="./MT">
```

```
<skos:inScheme rdf:resource="{$thesaurus-base-uri}t{substring( ./MT, 1, 4 )}"/>
</xsl:if>


<skos:prefLabel xml:lang="EN" >
<xsl:value-of select="./Term" />
</skos:prefLabel>
<skos:prefLabel xml:lang="FR" >
<xsl:value-of select="./FR" />
</skos:prefLabel>
<skos:prefLabel xml:lang="ES" >
<xsl:value-of select="./SP" />
</skos:prefLabel>


<xsl:for-each select="./UF">
<skos:altLabel xml:lang="EN" >
<xsl:value-of select="." />
</skos:altLabel>
</xsl:for-each>


<xsl:for-each select="./SN">
<skos:scopeNote xml:lang="EN" >
<xsl:value-of select="." />
</skos:scopeNote>
</xsl:for-each>


<xsl:for-each select="./BT">
<skos:broader rdf:resource="{$concept-base-uri}{.}"/>
</xsl:for-each>
<xsl:for-each select="./NT">
<skos:narrower rdf:resource="{$concept-base-uri}{.}"/>
</xsl:for-each>
<xsl:for-each select="./RT">
<skos:related rdf:resource="{$concept-base-uri}{.}"/>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DATABASE_UNESEN/RECORD">
<xsl:if test="$cnp-lbl = ./USE">
<skos:altLabel xml:lang="EN" >
<xsl:value-of select="./Term" />
</skos:altLabel>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DATABASE_UNESFR/RECORD">
<xsl:if test="$cnp-lbl-fr = ./EMP">
<skos:altLabel xml:lang="FR" >
<xsl:value-of select="./Terme" />
</skos:altLabel>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DATABASE_UNESSP/RECORD">
<xsl:if test="$cnp-lbl-sp = ./EMP">
<skos:altLabel xml:lang="ES" >
<xsl:value-of select="./Termino" />
</skos:altLabel>
</xsl:if>
</xsl:for-each>


<xsl:for-each select="/tag/tag/DATABASE_UNESRU/RECORD">
```

```
<xsl:if test="$cnp-lbl = ./EN and ./USE ">
<skos:altLabel xml:lang="RU" >
<xsl:value-of select="./Term" />
</skos:altLabel>
</xsl:if>
</xsl:for-each>


<xsl:for-each select="/tag/tag/DATABASE_UNESFR/RECORD">
<xsl:if test="$cnp-lbl = ./EN">
<xsl:for-each select="./EP">
<skos:altLabel xml:lang="FR" >
<xsl:value-of select="." />
</skos:altLabel>
</xsl:for-each>


<xsl:for-each select="./NE">
<skos:scopeNote xml:lang="FR" >
<xsl:value-of select="." />
</skos:scopeNote>
</xsl:for-each>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DATABASE_UNESSP/RECORD">
<xsl:if test="$cnp-lbl = ./EN">
<xsl:for-each select="./UP">
<skos:altLabel xml:lang="ES" >
<xsl:value-of select="." />
</skos:altLabel>
</xsl:for-each>


<xsl:for-each select="./NA">
<skos:scopeNote xml:lang="ES" >
<xsl:value-of select="." />
</skos:scopeNote>
</xsl:for-each>
</xsl:if>
</xsl:for-each>

<xsl:for-each select="/tag/tag/DATABASE_UNESRU/RECORD">
<xsl:if test="$cnp-lbl = ./EN and not( ./USE )">
<xsl:for-each select="./UF">
<skos:altLabel xml:lang="RU" >
<xsl:value-of select="." />
</skos:altLabel>
</xsl:for-each>


<xsl:for-each select="./SN">
<skos:scopeNote xml:lang="RU" >
<xsl:value-of select="." />
</skos:scopeNote>
</xsl:for-each>
</xsl:if>
</xsl:for-each>
</skos:Concept>
</xsl:if>
</xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

A.3 eclas2skos.xslt

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:eclas="http://www.loc.gov/MARC21/slim"
  version="1.0">

<xsl:output method="xml" indent="yes" />

<xsl:variable name="base-uri">http://www.ittig.it</xsl:variable>
<xsl:variable name="domain-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/domain#</xsl:text>
</xsl:variable>
<xsl:variable name="concept-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/concept#</xsl:text>
</xsl:variable>




<xsl:template match="/">
   <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:skos="http://www.w3.org/2004/02/skos/core#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

      <xsl:call-template name="print-thesaurus" />
      <xsl:call-template name="print-concepts" />
      </rdf:RDF>
</xsl:template>




<xsl:template name="print-thesaurus">
   <skos:Collection>
      <rdfs:label>Eclas Thesaurus</rdfs:label>

      <xsl:for-each
select="/eclas:collection/eclas:record/eclas:datafield[@tag='150'] |
/eclas:collection/eclas:record/eclas:datafield[@tag='151']">

         <xsl:if
  test="./eclas:subfield[ @code='9' ] = 'eng'and
              substring( ./eclas:subfield[ @code='a'], 4, 1 ) = ' '
  and not (../eclas:datafield[@tag='065'])">
            <skos:member
            rdf:resource="{$domain-base-uri}{substring(./eclas:subfield[ @code='a'],1,3)}"/>
         </xsl:if>

      </xsl:for-each>
   </skos:Collection>
```

```
   <xsl:for-each select="/eclas:collection/eclas:record/eclas:datafield[@tag='150'] |
        /eclas:collection/eclas:record/eclas:datafield[@tag='151']">

      <xsl:if test="./eclas:subfield[ @code='9' ] = 'eng'and
             substring( ./eclas:subfield[ @code='a'], 4, 1 ) = ' '
and not (../eclas:datafield[@tag='065'])">
         <xsl:variable name="domain-id" select="substring(./eclas:subfield[ @code='a'],1,3)" />
         <skos:Collection rdf:about="{$domain-base-uri}{$domain-id}">

    <xsl:for-each select="../eclas:datafield[@tag='150'] | ../eclas:datafield[@tag='151']">

               <xsl:call-template name="print-label">
                  <xsl:with-param name="label-text" select="./eclas:subfield[ @code='a']" />
                  <xsl:with-param name="label-lng" select="./eclas:subfield[ @code='9']" />
                  <xsl:with-param name="label-type" select="'rdfsLabel'" />
               </xsl:call-template>

    </xsl:for-each>

            <xsl:for-each select="/eclas:collection/eclas:record/eclas:datafield[@tag='150'] |
                 /eclas:collection/eclas:record/eclas:datafield[@tag='151']">
               <xsl:if test="./eclas:subfield[ @code='9' ] = 'eng'and
                    substring( ./eclas:subfield[ @code='a'], 7, 1 ) = ' '  and not
                    (../eclas:datafield[@tag='065']) and
                    substring(./eclas:subfield[ @code='a'],1,3) = $domain-id">
                  <skos:member
                   rdf:resource="{$domain-base-uri}{substring(./eclas:subfield[ @code='a'],1,6)}"/>
               </xsl:if>
            </xsl:for-each>


 </skos:Collection>
      </xsl:if>

   </xsl:for-each>



   <xsl:for-each
        select="/eclas:collection/eclas:record/eclas:datafield[@tag='150'] |
        /eclas:collection/eclas:record/eclas:datafield[@tag='151']">

      <xsl:if test="./eclas:subfield[ @code='9' ] = 'eng' and
             substring( ./eclas:subfield[ @code='a'], 4, 1 ) = '.' and
             substring( ./eclas:subfield[ @code='a'], 7, 1 ) = ' '  and
             not (../eclas:datafield[@tag='065'])">
         <xsl:variable name="domain-id" select="substring(./eclas:subfield[ @code='a'],1,6)" />
         <skos:Collection rdf:about="{$domain-base-uri}{$domain-id}">

    <xsl:for-each select="../eclas:datafield[@tag='150'] | ../eclas:datafield[@tag='151']">

               <xsl:call-template name="print-label">
                  <xsl:with-param name="label-text" select="./eclas:subfield[ @code='a']" />
                  <xsl:with-param name="label-lng" select="./eclas:subfield[ @code='9']" />
                  <xsl:with-param name="label-type" select="'rdfsLabel'" />
               </xsl:call-template>

    </xsl:for-each>

            <xsl:for-each select="/eclas:collection/eclas:record/eclas:datafield[@tag='150'] |
                 /eclas:collection/eclas:record/eclas:datafield[@tag='151']">
               <xsl:if test="./eclas:subfield[ @code='9' ] = 'eng' and
                       substring( ./eclas:subfield[ @code='a'], 10, 1 ) = ' '  and
                       not (../eclas:datafield[@tag='065']) and
                       substring(./eclas:subfield[ @code='a'],1,6) = $domain-id">
```

```
                <skos:member
                 rdf:resource="{$domain-base-uri}{substring(./eclas:subfield[ @code='a'],1,9)}"/>
              </xsl:if>
            </xsl:for-each>

 </skos:Collection>
      </xsl:if>

   </xsl:for-each>

   <xsl:for-each
        select="/eclas:collection/eclas:record/eclas:datafield[@tag='150'] |
        /eclas:collection/eclas:record/eclas:datafield[@tag='151']">

      <xsl:if test="./eclas:subfield[ @code='9' ] = 'eng' and
           substring( ./eclas:subfield[ @code='a'], 4, 1 ) = '.' and
           substring( ./eclas:subfield[ @code='a'], 7, 1 ) = '.' and
           substring( ./eclas:subfield[ @code='a'], 10, 1 ) = ' '  and
           not (../eclas:datafield[@tag='065'])">
         <xsl:variable
              name="domain-id" select="substring(./eclas:subfield[ @code='a'],1,9)" />
         <skos:Collection rdf:about="{$domain-base-uri}{$domain-id}">

    <xsl:for-each select="../eclas:datafield[@tag='150'] | ../eclas:datafield[@tag='151']">

               <xsl:call-template name="print-label">
                  <xsl:with-param name="label-text" select="./eclas:subfield[ @code='a']" />
                  <xsl:with-param name="label-lng" select="./eclas:subfield[ @code='9']" />
                  <xsl:with-param name="label-type" select="'rdfsLabel'" />
               </xsl:call-template>

    </xsl:for-each>

            <xsl:for-each
            select="/eclas:collection/eclas:record/eclas:datafield[ @tag='065']/
            eclas:subfield[ @code='a']">
               <xsl:if test="substring(.,1,9) = $domain-id">
                  <skos:member rdf:resource="{$concept-base-uri}{.}"/>
               </xsl:if>
            </xsl:for-each>

 </skos:Collection>
      </xsl:if>

   </xsl:for-each>
</xsl:template>




<xsl:template name="print-concepts">

   <xsl:for-each
    select="/eclas:collection/eclas:record/eclas:datafield[ @tag='065']/eclas:subfield[ @code='a']">
      <skos:Concept rdf:about="{$concept-base-uri}{.}">

         <xsl:for-each
              select="../../eclas:datafield[@tag='150' ] | ../../eclas:datafield[@tag='151' ]">
               <xsl:call-template name="print-label">
                  <xsl:with-param name="label-text" select="./eclas:subfield[ @code='a']" />
                  <xsl:with-param name="label-lng" select="./eclas:subfield[ @code='9']" />
                  <xsl:with-param name="label-type" select="'skosPrefLabel'" />
               </xsl:call-template>
         </xsl:for-each>
```

```
<xsl:for-each
     select="../../eclas:datafield[@tag='450' ] | ../../eclas:datafield[@tag='451' ]">
      <xsl:call-template name="print-label">
         <xsl:with-param name="label-text" select="./eclas:subfield[ @code='a']" />
         <xsl:with-param name="label-lng" select="./eclas:subfield[ @code='9']" />
         <xsl:with-param name="label-type" select="'skosAltLabel'" />
      </xsl:call-template>
</xsl:for-each>

<xsl:for-each select="../../eclas:datafield[@tag='680' ]">
      <xsl:call-template name="print-label">
         <xsl:with-param name="label-text" select="./eclas:subfield[ @code='i']" />
         <xsl:with-param name="label-lng" select="./eclas:subfield[ @code='9']" />
         <xsl:with-param name="label-type" select="'skosScopeNote'" />
      </xsl:call-template>
</xsl:for-each>

<!--
<xsl:for-each select="../../eclas:datafield[@tag='550'] | ../../eclas:datafield[@tag='551']">
   <xsl:if test="./eclas:subfield[@code='w']='g'">
      <xsl:choose>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)=' '">
            <skos:broader
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,3)}"/>
         </xsl:when>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)='.' and
             substring(./eclas:subfield[@code='a'],7,1)=' '">
            <skos:broader
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,6)}"/>
         </xsl:when>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)='.' and
          substring(./eclas:subfield[@code='a'],7,1)='.' and
          substring(./eclas:subfield[@code='a'],10,1)=' '">
            <skos:broader
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,9)}"/>
         </xsl:when>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)='.' and
          substring(./eclas:subfield[@code='a'],7,1)='.' and
          substring(./eclas:subfield[@code='a'],10,1)='.'">
            <skos:broader
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,14)}"/>
         </xsl:when>
      </xsl:choose>
   </xsl:if>
</xsl:for-each>

<xsl:for-each
 select="../../eclas:datafield[ @tag='550'] | ../../eclas:datafield[@tag='551']">
   <xsl:if test="./eclas:subfield[@code='w']='h'">
      <xsl:choose>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)=' '">
            <skos:narrower
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,3)}"/>
         </xsl:when>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)='.' and
          substring(./eclas:subfield[@code='a'],7,1)=' '">
            <skos:narrower
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,6)}"/>
         </xsl:when>
         <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)='.' and
             substring(./eclas:subfield[@code='a'],7,1)='.' and
             substring(./eclas:subfield[@code='a'],10,1)=' '">
            <skos:narrower
             rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,9)}"/>
```

```
                </xsl:when>
                <xsl:when test="substring(./eclas:subfield[@code='a'],4,1)='.' and
                   substring(./eclas:subfield[@code='a'],7,1)='.' and
                   substring(./eclas:subfield[@code='a'],10,1)='.'">
                   <skos:narrower
                    rdf:resource="{$concept-base-uri}{substring(./eclas:subfield[@code='a'],1,14)}"/>
                </xsl:when>
             </xsl:choose>
          </xsl:if>
       </xsl:for-each>
       -->
    </skos:Concept>
  </xsl:for-each>

</xsl:template>



<xsl:template name="print-label">
  <xsl:param name="label-text" />
  <xsl:param name="label-lng" />
  <xsl:param name="label-type" />

  <xsl:variable name="decoded-lng">
    <xsl:choose>
      <xsl:when test="$label-lng ='eng'">
        <xsl:text>EN</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='fre'">
        <xsl:text>FR</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='dan'">
        <xsl:text>DA</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='swe'">
        <xsl:text>SE</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='fin'">
        <xsl:text>FI</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='ita'">
        <xsl:text>IT</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='por'">
        <xsl:text>PT</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='cze'">
        <xsl:text>CS</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='dut'">
        <xsl:text>NL</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='bul'">
        <xsl:text>BG</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='ger'">
        <xsl:text>DE</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='gre'">
        <xsl:text>EL</xsl:text>
      </xsl:when>
      <xsl:when test="$label-lng ='hun'">
        <xsl:text>HU</xsl:text>
```

```
         </xsl:when>
         <xsl:when test="$label-lng ='lav'">
            <xsl:text>LV</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='lit'">
            <xsl:text>LT</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='pol'">
            <xsl:text>PL</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='slo'">
            <xsl:text>SI</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='slv'">
            <xsl:text>SK</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='spa'">
            <xsl:text>ES</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='rum'">
            <xsl:text>RO</xsl:text>
         </xsl:when>
         <xsl:when test="$label-lng ='scr'">
            <xsl:text>HR</xsl:text>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>EN</xsl:text>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:variable>

   <xsl:choose>
     <xsl:when test="$label-type='rdfsLabel'">
        <rdfs:label xml:lang="{$decoded-lng}">
           <xsl:value-of select="$label-text"/>
        </rdfs:label>
     </xsl:when>
     <xsl:when test="$label-type='skosPrefLabel'">
        <skos:prefLabel xml:lang="{$decoded-lng}">
           <xsl:value-of select="$label-text"/>
        </skos:prefLabel>
     </xsl:when>
     <xsl:when test="$label-type='skosAltLabel'">
        <skos:altLabel xml:lang="{$decoded-lng}">
           <xsl:value-of select="$label-text"/>
        </skos:altLabel>
     </xsl:when>
     <xsl:when test="$label-type='skosScopeNote'">
        <skos:scopeNote xml:lang="{$decoded-lng}">
           <xsl:value-of select="$label-text"/>
        </skos:scopeNote>
     </xsl:when>
  </xsl:choose>

</xsl:template>


</xsl:stylesheet>
```

A.4 ett2skos.xslt

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  version="1.0"
>

<xsl:output method="xml" indent="yes" />

<xsl:variable name="base-uri">http://www.ittig.it</xsl:variable>
<xsl:variable name="domain-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/domain#</xsl:text>
</xsl:variable>
<xsl:variable name="thesaurus-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/micro-thesaurus#</xsl:text>
</xsl:variable>
<xsl:variable name="concept-base-uri">
   <xsl:value-of select="$base-uri" />
   <xsl:text>/concept#</xsl:text>
</xsl:variable>


<xsl:key name="domain-id" match="/THESAURUS/CONCEPT/FD" use="."/>

<xsl:template match="/">
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:skos="http://www.w3.org/2004/02/skos/core#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<xsl:call-template name="print-domain" />
<xsl:call-template name="print-concepts" />
</rdf:RDF>

</xsl:template>


<xsl:template name="print-domain">
<skos:Collection>
   <rdfs:label>ETT Thesaurus</rdfs:label>

<xsl:for-each select="/THESAURUS/CONCEPT/FD[generate-id()=generate-id(key('domain-id',.))]">
<xsl:variable name="dom-id1" select="generate-id(.)"/>
     <skos:member rdf:resource="{$domain-base-uri}{$dom-id1}"/>
</xsl:for-each>
     </skos:Collection>

<xsl:for-each select="/THESAURUS/CONCEPT/FD[generate-id()=generate-id(key('domain-id',.))]">
<xsl:variable name="dom-id2" select="generate-id(.)"/>
<xsl:variable name="dom-lb1" select="." />

<skos:Collection rdf:about="{$domain-base-uri}{$dom-id2}">

<rdfs:label xml:lang="EN" >
<xsl:value-of select="$dom-lb1" />
</rdfs:label>

<xsl:for-each select="/THESAURUS/CONCEPT">
<xsl:if test="(./DESCRIPTOR ) and not (./BT )">
<xsl:variable name="dom-lb2" select="./FD" />

<xsl:if test="$dom-lb1 = $dom-lb2">
```

```
     <skos:member rdf:resource="{$concept-base-uri}d{translate(./DESCRIPTOR,' ','-')}"/>
     </xsl:if>
     </xsl:if>
</xsl:for-each>

     </skos:Collection>
</xsl:for-each>

</xsl:template>



<xsl:template name="print-concepts">
<xsl:for-each select="/THESAURUS/CONCEPT">
<xsl:choose>
<xsl:when test="./DESCRIPTOR">
<skos:Concept rdf:about="{$concept-base-uri}d{translate(./DESCRIPTOR,' ','-')}">
<xsl:apply-templates/>
</skos:Concept>
</xsl:when>

<xsl:otherwise>
<skos:Concept rdf:about="{$concept-base-uri}nd{translate(./NON-DESCRIPTOR,' ','-')}">
<xsl:apply-templates/>
</skos:Concept>
</xsl:otherwise>
</xsl:choose>

</xsl:for-each>

</xsl:template>



<xsl:template match="ENG">
<skos:prefLabel xml:lang="EN" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>

<xsl:template match="DAN">
<skos:prefLabel xml:lang="DA" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>

<xsl:template match="DUT">
<skos:prefLabel xml:lang="NL" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>

<xsl:template match="EST">
<skos:prefLabel xml:lang="ET" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>

<xsl:template match="FIN">
<skos:prefLabel xml:lang="FI" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>
```

```
<xsl:template match="FRE">
<skos:prefLabel xml:lang="FR" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>

<xsl:template match="ITA">
<skos:prefLabel xml:lang="IT" >
<xsl:value-of select="." />
</skos:prefLabel>
</xsl:template>



<xsl:template match="STA">
<skos:editorialNote>
<xsl:value-of select="." />
</skos:editorialNote>
</xsl:template>

<xsl:template match="UPD">
<skos:changeNote>
<xsl:value-of select="." />
</skos:changeNote>
</xsl:template>



<xsl:template match="SN">
<skos:scopeNote>
<xsl:value-of select="." />
</skos:scopeNote>
</xsl:template>


<xsl:template match="UF">
<skos:altLabel>
<xsl:value-of select="." />
</skos:altLabel>
</xsl:template>



<xsl:template match="BT">
<skos:broader rdf:resource="{$concept-base-uri}d{translate(.,' ','-')}"/>
</xsl:template>

<xsl:template match="NT">
<skos:narrower rdf:resource="{$concept-base-uri}d{translate(.,' ','-')}"/>
</xsl:template>

<xsl:template match="RT">
<skos:related rdf:resource="{$concept-base-uri}d{translate(.,' ','-')}"/>
</xsl:template>


<xsl:template match="text()|@*" />


</xsl:stylesheet>
```

References

[1] Wen-Tao Cai, Sheng-Rui Wang, and Qing-Shan Jiang. Address extraction: a graph matching and ontology-based approach to conceptual information retrieval. In *Proceedings of the Third Intemational Conference on Machine Learning and Cybernetics*, pages 1571–1576, 2004.

[2] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.