Potential XML technologies for LMF (auxiliary working paper for LMF)

Gil Francopoulo INRIA Loria

29 July 2005

1 Introduction

This working paper is a study for XML specifications to be added (possibly[1]) in the LMF document as an annex. This document is not to be considered as a reference but instead as a first step to prepare the ground.

For this study only two sub-parts of LMF will be taken into account: the core model and some elements from the NLP morphological extension. This extension is to be used as an example in order to induce the process needed for the other extensions.

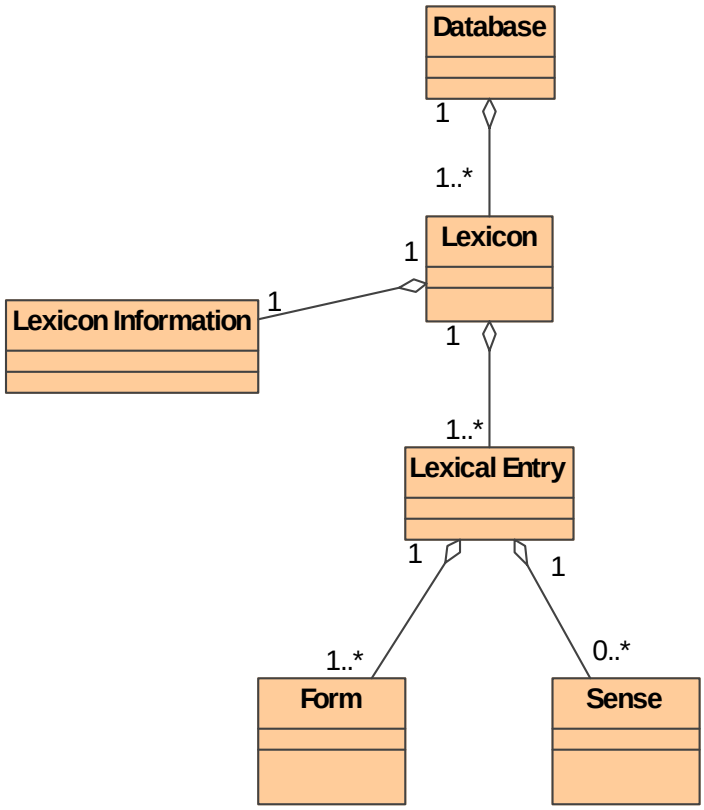
The following XML technologies are tried:

- Option-A: a conventional DTD description
- Option-B: a GMT description
- Option-C: a Relax NG schema
- Option-D: a W3C schema
- Option-E: a RDF/OWL-DL description
- Option-F: an ODD process

2 LMF according to UML class model

Let's recall that the core model is defined as follows:

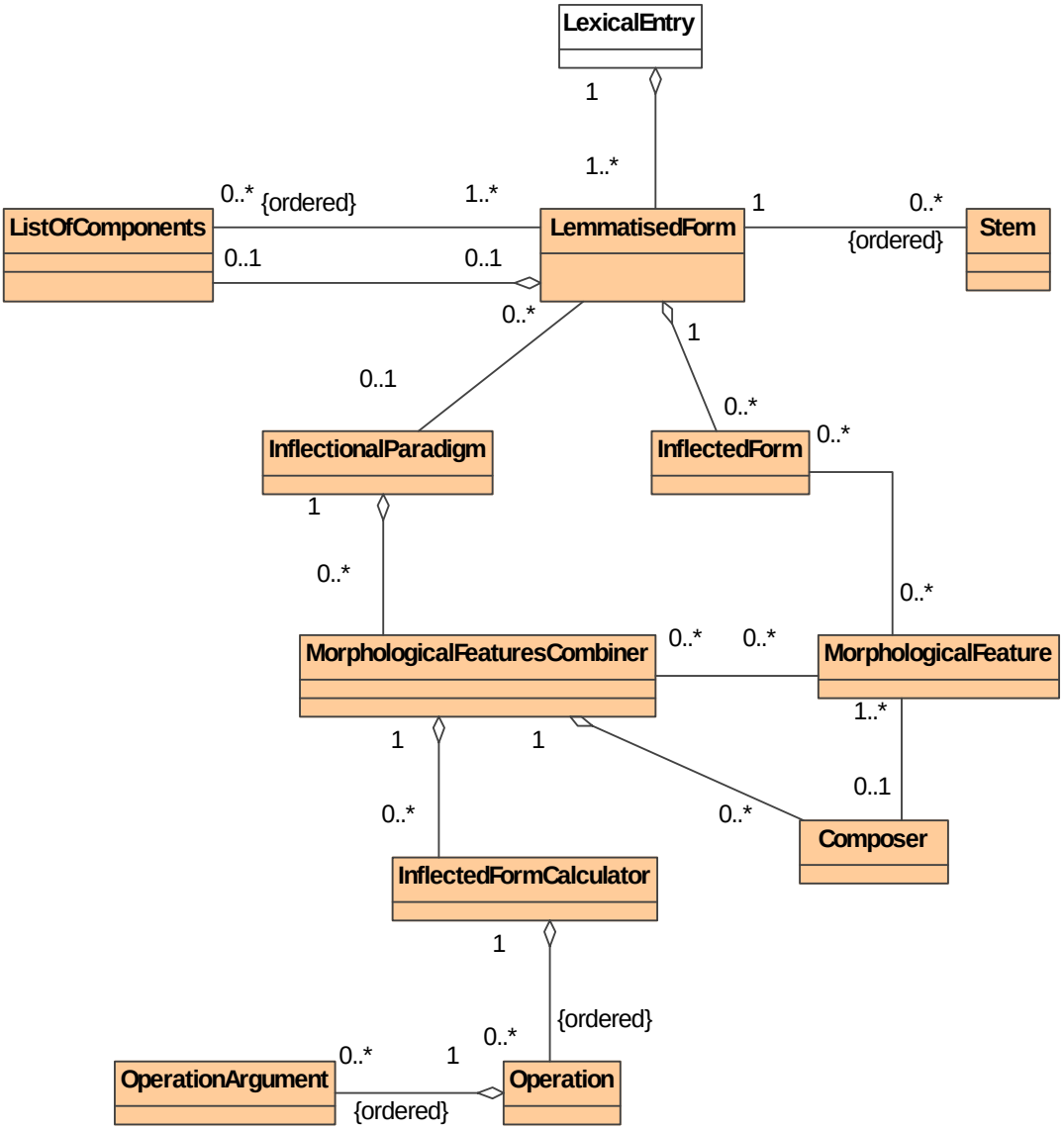
[1] This is to be decided together with my co-editor Monte George.



Classes can be adorned by attributes comprised of a pair of data categories. In the core model, all attributes are optional except the attributes /part of speech/ and /identifier/ held by the Lexical Entry. These two attributes are mandatory.

A developer can select one or several extensions that will be added to the core model.

The NLP morphological extension that is defined as follows:



We take into account only one extension in order to figure out how an extension will be added to the core model.

Within this extension, only InflectionalParadigm and ListOfComponents are taken into account. InflectionalParadigm is selected in order to study i) how to represent aggregation from another element than Lexical Entry and ii) how to represent identifier reference. ListOfComponents is selected in order to study how to represent ordered elements.

Being global to the whole lexicon InflectionParadigm is to be inserted within Lexicon.

3 Example instance

In order to run XML validation checkers, an example instance is written. This instance has the following elements: one database, one lexicon, one lexicon information (with language attribute), three lexical entries (two simple words and one MWE) and one inflectional paradigm. The example describes "heavy rain" as a MWE and "heavy" and "rain" as simple words.

Depending on the selected XML technology, the first lines of this instance could be a little bit different, but the rest, composed of the informative content should be identical.

For the purpose of this study, a certain XML style has been chosen: this is not very important. Only the main features of the various technologies are important.

The informative content is as follows:

```
<Database>
  <Lexicon>
    <LexiconInformation>
      <DatCat att="language" val="eng"/>
    </LexiconInformation>
    <LexicalEntry id="LE1" partOfSpeech="adjective">
      <LemmatisedForm>
        <WrittenForm>heavy</WrittenForm>
      </LemmatisedForm>
    </LexicalEntry>
    <LexicalEntry id="LE2" partOfSpeech="noun">
      <LemmatisedForm inflection="asTable">
        <WrittenForm>rain</WrittenForm>
      </LemmatisedForm>
    </LexicalEntry>
    <LexicalEntry id="LE3" partOfSpeech="noun">
      <LemmatisedForm>
        <WrittenForm>heavy rain</WrittenForm>
      </LemmatisedForm>
      <ListOfComponents idrefs="LE1 LE2"/>
    </LexicalEntry>
    <InflectionalParadigm id="asTable"/>
  </Lexicon>
</Database>
```

4 Option-A: Conventional DTD description

4.1 Conventions

This specification respects ISO 8779 modified for XML. This technology is the base of XML and is often called XML-1.0. The term "conventional" is used in order to qualify the fact that UML class names are implemented through XML element names. This term is used in order to distinguish this strategy from the GMT specification.

Adornment is implemented by the means of a pair “att” / “val”. The former is for a data category name like /grammatical gender/. The latter is for a data category value like /masculine/. The mandatory attributes are implemented by the means of an attribute like /part of speech/ for Lexical Entry.

4.2 Core model XML implementation

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Database (DatCat*, Lexicon+) >
```

```
<!--        to be filled by a pair taken from the data category selection →
<!ELEMENT DatCat >
<!ATTLIST DatCat
        att             CDATA   #REQUIRED
        val             CDATA   #REQUIRED>
<!ELEMENT Lexicon (DatCat*, LexiconInformation, LexicalEntry+) >
<!ELEMENT LexiconInformation (DatCat*) >
<!ELEMENT LexicalEntry (DatCat*, Form+, Sense*) >
<!ATTLIST LexicalEntry
        id              ID      #REQUIRED
        partOfSpeech    CDATA   #REQUIRED>
<!ELEMENT Form (DatCat*) >
<!ELEMENT Sense (DatCat*)>
```

4.3 NLP Morphological extension XML implementation

A subset of the extension is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT LexicalEntry (DatCat*, LemmatisedForm+) >
<!ATTLIST LexicalEntry
        Id              ID      #REQUIRED
        partOfSpeech    CDATA   #REQUIRED>
```

4.4 unresolved issue(s)

With a DTD, in fact with XML 1.0, element redefinition is not possible. Specifications in separate files are not possible. Modularity is nonexistent.

As a consequence, extension composition is not defined and left to the user burden.

5 Option-B: Generic mapping tool

5.1 Conventions

This specification respects ISO 16642 standard that defines GMT as a generic mapping tool.

5.2 XML implementation

UML class names are translated by the use of a generic struct XML element with a specific type value.

The GMT DTD is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT struct ((feat|brack)*,struct*)>
<!ATTLIST struct    type        (DB,LI,LX,LE,FO,SE)  #REQUIRED
                    id          ID                   #IMPLIED
                    target      CDATA                #IMPLIED>
<!ELEMENT feat (#PCDATA|annot)*>
<!ATTLIST feat      type        CDATA                #REQUIRED
                    target      CDATA                #IMPLIED
```

```
                        source      CDATA               #IMPLIED>
<!ELEMENT brack (feat+,(feat|brack)*)>
<!ATTLIST brack         source      CDATA               #IMPLIED>
<!ELEMENT annot (#PCDATA)>
<!ATTLIST annot         type        CDATA               #REQUIRED
                        target      CDATA               #REQUIRED>
```

5.3 unresolved issue(s)

With a DTD, in fact with XML 1.0, element redefinition is not possible. Specifications in separate files are not possible. Modularity is nonexistent.

As a consequence, extension composition is not defined and left to the user burden.

6 Option-C: Relax NG XML schema

6.1 Conventions

Relax NG is a roughly a sub-set of the W3C recommendation dedicated to XML schemas. Only some specific features are not covered by the W3C schemas. Relax NG is the only part of these recommendations that has been published by ISO. The ISO name is ISO/IEC 19757-2. Relax NG is one of the projects of ISO-DSDL. Relax NG is well defined and less complex than the W3C schemas.

Relax NG is described at http://relaxng.org and in "Eric van der Vlist: Relax NG, O'Reilly 2003".

6.2 XML implementation

Composition is implemented through external grammar inclusion. Some elements are redefined using the RELAX NG combine feature. UML generalization being impossible within RELAX NG, Adornment is implemented by the means of the pair define/reference. Two files are presented below. This schema has been verified using JING (http://www.thaiopensource.com/relaxng/jing.html) from James Clarke.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ##########################################BEGINNING OF FILE lmf.rxg-->
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
      datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
<start>
            <ref name="Database"/>
</start>
<!--#################################################################database -->
<define name="Database">
<element name="Database">
            <ref name="Adornment"/>
            <oneOrMore>
            <ref name="Lexicon"/>
            </oneOrMore>
</element>
</define>
```

```
<!--#############################################################lexicon -->
<define name="Lexicon">
<element name="Lexicon">
        <ref name="LexiconContent"/>
</element>
</define>
<define name="LexiconContent">
        <ref name="Adornment"/>
        <ref name="LexiconInformation"/>
        <oneOrMore>
        <ref name="LexicalEntry"/>
        </oneOrMore>
</define>
<!--#################################################lexicon information-->
<define name="LexiconInformation">
<element name="LexiconInformation">
        <ref name="Adornment"/>
</element>
</define>
<!--#########################################################lexical entry -->
<define name="LexicalEntry">
<element name="LexicalEntry">
         <ref name="LexicalEntryContent"/>
</element>
</define>
<define name="LexicalEntryContent">
        <!--must be able to be referenced-->
        <attribute name="id"><data type="ID"/></attribute>
        <attribute name="partOfSpeech"/>
        <ref name="Adornment"/>
        <zeroOrMore>
                <ref name="Sense"/>
        </zeroOrMore>
</define>
<!--#####################################################################Sense-->
<define name="Sense">
<element name="Sense">
         <ref name="SenseContent"/>
</element>
</define>
<define name="SenseContent">
        <ref name="Adornment"/>
</define>
<!--for data categories=pairs taken from the data category selection -->
<define name="Adornment">
        <zeroOrMore>
        <element name="DatCat">
```

```
                <attribute name="att"/>
                <attribute name="val"/>
                </element>
                </zeroOrMore>
</define>
<include href="NLPMorphology.rxg"/>
<!--location for other extensions -->
</grammar>
<!--#####################################END OF FILE-->
<!--#####################BEGINNING OF FILE NLPMorphology.rxg-->
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
        datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
<!--##########################################redef with addition -->
<define name="Lexicon" combine="choice">
<element name="Lexicon">
                <ref name="LexiconContent"/>
                <zeroOrMore>
                <ref name="InflectionalParadigm"/>
                </zeroOrMore>
</element>
</define>
<!--##########################################redef with addition -->
<define name="LexicalEntry" combine="choice">
<element name="LexicalEntry">
                <ref name="LexicalEntryContent"/>
                <oneOrMore>
                <ref name="LemmatisedForm"/>
                </oneOrMore>
                <optional>
                <ref name="ListOfComponents"/>
                </optional>
</element>
</define>
<!--#################################################Lemmatised Form-->
<define name="LemmatisedForm">
<element name="LemmatisedForm">
                <ref name="Adornment"/>
        <element name="WrittenForm"><text/></element>
        <optional>
                <!--Because idref types are forbidden in elements-->
                <attribute name="inflection"><data type="IDREF"/></attribute>
                </optional>
        <optional>
                <ref name="ListOfComponents"/>
        </optional>
</element>
</define>
```

```
<!--#################################################ListOfComponents-->
<define name="ListOfComponents">
<element name="ListOfComponents">
        <attribute name="idrefs"><data type="IDREFS"/></attribute>
</element>
</define>
<!--#################################################InflectionalParadigm-->
<define name="InflectionalParadigm">
<element name="InflectionalParadigm">
<!--must be able to be referenced, so an XML identifier is mandatory -->
        <attribute name="id"><data type="ID"/></attribute>
        <ref name="Adornment"/>
</element>
</define>
</grammar>
<!-- ########################################END OF FILE -->
```

7 Option-D: W3C schema

7.1 Conventions

W3C schema specifications can be found at www.w3.org/XML/Schema. W3C schemas are rather complex and subject to various debates among the XML community. The W3C specification is so complex and so messy that a validator implementing the full specification does not seem to exist.

W3C schemas are described in "Eric van der Vlist: XML schema, O'Reilly 2002".

7.2 XML implementation

Adornment is implemented thru schema generalization by the means of the XML <xs:extension> tag.

This schema has been verified using Sun Multi-schema Validator (aka MSV) but the validation produces errors for the reasons explained in the Problem section just below. As said in the MSV document, the validator implements only a sub-set of the W3C recommendation.

```
<?xml version="1.0"?>
<!--##############################BEGINNING OF FILE lmf.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!--#################################################-->
<xs:element name="Database">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="Adorned">
        <xs:all minOccurs="1">
          <xs:element ref="Lexicon"/>
        </xs:all>
      </xs:extension>
    </xs:complexContent>
```

```
  </xs:complexType>
</xs:element>
<!--#################################################-->
<xs:element name="Lexicon">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="Adorned">
        <xs:sequence>
          <xs:element ref="LexiconInformation"/>
          <xs:all minOccurs="1">
            <xs:element ref="LexicalEntry"/>
          </xs:all>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<!--#################################################-->
<xs:element name="LexiconInformation">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="Adorned"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<!--#################################################-->
<xs:element name="LexicalEntry">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="LexicalEntryType">
<!--        <xs:sequence minOccurs="1">
          <xs:element ref="Form"/>
        </xs:sequence> -->
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:complexType name="LexicalEntryType">
  <xs:complexContent>
      <xs:extension base="Adorned">
        <xs:attribute ref="id"/>
        <xs:attribute ref="partOfSpeech"/>
      </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!--#################################################-->
<xs:element name="Form">
```

```
  <xs:complexType name="Form">
  </xs:complexType>
</xs:element>
<!--#################################################-->
<xs:complexType name="Adorned">
  <xs:complexContent>
    <xs:sequence minOccurs="0">
      <xs:element ref="DatCat"/>
    </xs:sequence>
  </xs:complexContent>
</xs:complexType>
<!--#################################################-->
<xs:element name="DatCat">
  <xs:complexType>
      <xs:attribute ref="att"/>
      <xs:attribute ref="val"/>
  </xs:complexType>
</xs:element>
<!--#################################################-->
<xs:element name="name"                                  type="xs:token"/>
<xs:attribute name="id"                                  type="xs:ID"/>
<xs:attribute name="partOfSpeech"            type="xs:token"/>
<xs:attribute name="att"                         type="xs:token"/>
<xs:attribute name="val"                         type="xs:token"/>
<!--#################################################-->
<xs:redefine schemaLocation="NLPMorphology.xsd"/>
</xs:schema>
<!--##############################END OF FILE -->
<?xml version="1.0"?>
<!--########################BEGINNING OF FILE NLPMorphology.xsd -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="LexicalEntry">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="LexicalEntryType">
        <xs:sequence minOccurs="1">
          <xs:element ref="LemmatisedForm"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<!--#########################################################-->
<xs:element name="LemmatisedForm">
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="Form">
```

```
      <xs:sequence>
        <xs:element ref="WrittenForm"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
<!--#########################################################-->
<xs:element name="WrittenForm"                type="xs:string"/>
<!--#########################################################-->
</xs:schema>
```

7.3 Problem

Let's recall that the LMF is defined by a core model and some included extensions. With W3C schema it is not possible to keep such a structure. It's not possible to redefine within an included file an element that is defined in the top level file.

8 Option-E: RDF/OWL-DL description

8.1 Conventions

RDF/OWL-DL is based on W3C schema recommendations. RDF/OWL-DL is itself a W3C recommendation. RDF/OWL-DL is rather popular among the Semantic web community.

RDF/OWL-DL is described at www.w3.org/TR/owl-guide. RDF/OWL-DL is described in "Grigoris Antoniou & Frank van Harmelen: A semantic web primer, MIT press 2004" or in "Heiner Stuckenschmidt & Frank van Harmelen: Information sharing on the semantic web, Springer Verlag 2005".

8.2 XML implementation

Adornment is implemented thru the tag: "rdfs:subClassOf".

```
<?xml version="1.0" encoding="UTF-8"?>
<!--#########################BEGINNING OF FILE CoreModel.rdf →
<rdf:RDF    xmlns:rdf=  "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
            xmlns:owl= "http://www.w3.org/2002/07/owl#">
<!-##############################################top level and generic  stuff →
<owl:TransitiveProperty rdf:ID="has"/>
<owl:Class rdf:ID="Adorned">
            <owl:intersectionOf  rdf:parseType="Collection">
            <!-- sub-class of the most generic class→
            <owl:Class rdf:about="owl:Thing"/>
            <owl:Restriction>
                        <owl:onProperty rdf:resource="#has"/>
                        <owl:someValuesForm rdf:resource="#Adornment"/>
            </owl:Restriction>
            </owl:intersectionOf>
</owl:Class>
```

```
<owl:Class rdf:ID="Adornment">
        <rdf:Bag>
        <owl:onProperty rdf:resource="#AdornmentAtt"/>
        <owl:onProperty rdf:resource="#AdornmentVal"/>
        </rdf:Bag>
</owl:Class>
<owl:DatatypeProperty rdf:ID="#AdronmentAtt">
        <rdf:range          rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="#AdronmentVal">
        <rdf:range          rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!-##################################################database →
<owl:Class rdf:ID="Database">
        <rdfs:subClassOf rdf:resource="#Adorned"/>
        <rdf:subClassOf>
                <owl:Restriction>
                <owl:onProperty rdf:resource="#has"/>
                <owl:minCardinality rdf:datatype="#xsd:nonNegativeInteger">1
                </owl:minCardinality>
                <owl:allValuesFrom rdf:resource="#Lexicon"/>
                </owl:Restriction>
        </rdf:subClassOf>
</owl:Class>
<!-##################################################lexicon →
<owl:Class rdf:ID="Lexicon">
        <rdfs:subClassOf rdf:resource="#Adorned"/>
        <rdf:subClassOf>
                <owl:Restriction>
                <owl:onProperty rdf:resource="#has"/>
                <owl:allValuesFrom rdf:resource="#LexiconInformation"/>
                </owl:Restriction>
        </rdf:subClassOf>
        <rdf:subClassOf>
                <owl:Restriction>
                <owl:onProperty rdf:resource="#has"/>
                <owl:minCardinality rdf:datatype="#xsd:nonNegativeInteger">1
                </owl:minCardinality>
                <owl:allValuesFrom rdf:resource="#LexicalEntry"/>
                </owl:Restriction>
        </rdf:subClassOf>
        <owl:imports rdf:resource="NLPMorphologicalExtensionConstants.rdf"/>
        <!—calls to other constants for extensions are to be located here →
</owl:Class>
<!--##################################################lexicon information→
<owl:Class rdf:ID="LexiconInformation">
        <rdfs:subClassOf rdf:resource="#Adorned"/>
```

```
</owl:Class>
<!--#################################################lexical entry →
<owl:Class rdf:ID="LexicalEntry">
        <rdfs:subClassOf rdf:resource="#Adorned"/>
        <rdf:Bag>
        <owl:onProperty rdf:resource="#LexicalEntryId"/>
        <owl:onProperty rdf:resource="#PartOfSpeech"/>
        </rdf:Bag>
        <rdf:subClassOf>
                <owl:Restriction>
                <owl:onProperty rdf:resource="#has"/>
                <owl:minCardinality rdf:datatype="#xsd:nonNegativeInteger">0
                </owl:minCardinality>
                <owl:allValuesFrom rdf:resource="#Sense"/>
                </owl:Restriction>
        </rdf:subClassOf>
        <owl:imports rdf:resource="NLPMorphologicalExtensionContent.rdf"/>
        <!—calls to other contents from extensions are to be located here →
</owl:Class>
<owl:DatatypeProperty rdf:ID="#LexicalEntryId">
        <rdf:range              rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="#PartOfSpeech">
        <rdf:range              rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<!--#############################################Sense→
<owl:Class rdf:ID="Sense">
        <rdfs:subClassOf rdf:resource="#Adorned"/>
</owl:Class>
</rdf:RDF>
<!--#####################END OF FILE →
<!--#####################BEGINNING OF FILE NLPMorphologicalExtensionContent.rng→
<!-- we use about in order to extend parent definition when the class is already defined→
<?xml version="1.0" encoding="UTF-8"?>
<owl:Class rdf:ID="LemmatisedForm">
        <rdfs:subClassOf rdf:resource="#Adorned"/>

        <!-- to be completed
</owl:Class>
<!--#####################END OF FILE →
```

8.3 Problem

The problem is the same as for the W3C schema technology: it does not seem possible to redefine any element in included files.

9 Option-F: ODD process

ODD stands for "One document does it all" and is an on-going process lead by the TEI and the Oxford University Computing Services Department. ODD is used within the TEI-P5 project in order to generate Relax NG schemas.

ODD is described at http://www.tei-c.org.uk/P5/Guidelines/TD.html

=> TO BE COMPLETED AND COMPARED

10 Conclusion

Options A and B (aka DTD and GMT) do not permit any sort of modularization. Due to the fact that our architecture is based on a core model and additional extensions, these options are not suited for our purpose.

Options D and E (aka W3C schemas and RDF/OWL-DL) do not allow any redefinition of core model elements.

Option C (aka Relax NG) do not permit any generalization. This doesn't seem to be a problem as far as Form is not defined within core model.

Conventional DTD and Relax NG are the only XML technologies that are ISO standards. **Until now, Relax NG is the best candidate for LMF.**

Before taking a final decision, the following actions must be done in sequence:

- Talk about the strategy concerning such and such technical decision.
- Full writing of the various extensions, in order to be sure that all LMF features are able to be represented.