# Metashare as an ontology for the interoperability of linguistic datasets

Philipp Cimiano[1], Jorge Gracia[2], Penny Labropoulou[3], John P. McCrae[1], Víctor Rodríguez Doncel[2], and Marta Villegas[4]

[1] Cognitive Interaction Technology, Excellence Cluster, Bielefeld University, Inspiration 1, D-33619 Germany,
`{cimiano, jmccrae}@cit-ec.uni-bielefeld.de`

[2] Ontology Engineering Group, Universidad Politécnica de Madrid, Boadilla del Monte, Madrid, Spain
`{jgracia, vrodriguez}@fi.upm.es`

[3] Athena R.C./ILSP, Athens, Greece,
`penny@ilsp.athena-innovation.gr`

[4] University Pompeu Fabra, Barcelona, Spain,
`marta.villegas@upf.edu`

**Abstract. Keywords:** keywords

## 1 Introduction

[1]

## 2 Related Work

## 3 The META-SHARE Ontology

### 3.1 Original MS XSD schema [PL]

### 3.2 Purpose of the ontology [MV,JG,JPM]

(e.g., why do RDF and OWL for an already defined vocabulary?)

### 3.3 Formal modelling and mapping issues [MV, JPM, PL]

### 3.4 Interface with DCAT and other vocabularies [JPM]

### 3.5 Licensing module [VRD, PL]

Victor: I suggest the following structure:

FIRST: We describe first the Metashare schema, whose licensing information is described in an independent XML Schema file, available on git [5].

[5] https://github.com/metashare/META-SHARE/blob/master/misc/schema/v3.0/META-SHARE-LicenseMetadata.xsd

THIRD: We discuss on the needs that motivated the evolution from the previous model.

FOURTH: We list the changes that we have introduced.

Short introduction on the ODRL vocabulary. And going beyond ODRL: License Templates as an easy entry points for Semantic Web - laymans.

Example of license template, example of license. Directly in TTL. Maybe introducing a figure depecting what is metadata for resource/distrubiont/license?

## 4 META-SHARE in LingHub

LingHub is a large resource containing information about a wide range of language resources, but unlike META-SHARE it does not directly collect this information, but instead harmonizes the metadata from a wide range of sources. In this section, we will first describe how the original META-SHARE data was translated into RDF and the alignment with DCAT [?], previously described, was achieved. Furthermore, we will then consider how we have used the META-SHARE vocabulary as a base vocabulary to align terms from other resources included in LingHub.

### 4.1 LIXR mapping methodology [JPM]

When translating XML documents into RDF, one of the most common approaches is based on Extensible Stylesheet Language Transformations (XSLT) [?,?,?], which has been extended by some authors into a significant framework [?]. However, XSLT has a number of disadvantages for this task:

- The set of functions and operators suported by most processors is limited.
- Limited ability to declare new functions.
- Does not allow stream (SAX) processing of large files.
- XSLT is a one-way transformation language and it is not this possible to 'round-trip' the conversion, i.e., convert RDF to XML.
- XSLT syntax is expressed in XML and thus is very verbose and aesthetically unpleasing. For this reason, many people use alternative more compact syntaxes[6][7]

Furthermore, the META-SHARE syntax is very complex consisting of 111 complex types and 207 simple types. As such we deemed that the development of a new language for transformation and writing our converter in that language would take less development effort than writing a conversion entirely in XSLT. The mapping methodology we developed is a domain-specific langauge [?] called Lightweight Invertible XML to RDF Conversion (LIXR) and aims to improve on the situation by fixing the concerns above.

To begin with we selected the Scala programming language as the basis for LIXR as it has a proven syntactic flexibility that makes it easy to write domain-specific languages [?]. A simple example of a LIXR mapping is given below:

[6] Compact XML: https://pythonhosted.org/compactxml/

[7] Jade: http://jade-lang.com/

```
object Metashare extends eu.liderproject.lixr.Model {
  val dc = Namespace("http://purl.org/dc/elements/1.1/")
  val ms = Namespace("http://purl.org/ms-lod/MetaShare.ttl#")
  val msxml = Namespace("http://www.ilsp.gr/META-XMLSchema")

  msxml.resourceInfo --> (
    a > ms.ResourceInfo,
    handle(msxml.identificationInfo)
  )

  msxml.identificationInfo --> (
    dc.title > content(msxml.resourceName) @@ msxml.resourceName.att("lang")
  )
}
```

In this example, we first create our model extending the basic LIXR model and define namespaces as dynamic Scala objects[8]. We then make two mapping declarations for the tags `resourceInfo` and `identificationInfo`. LIXR (as XSLT) simply searches for a matching declaration at the root of the XML document to begin the transformation. Having matched the `resourceInfo` tag, the system first generates the triple that states that the base element has type `ms:resourceInfo`, and then 'handles' any children `identificationInfo` tags by searching for an appropriate rule for each one. For `identificationInfo` the system generates a triple using the `dc:title` property whose value is the content of the `resourceName` tag tagged with the language given by the attribute `lang`.

| Name | Tags | Implementation | LoC | LoC/Tag |
|---|---|---|---|---|
| TBX | 48 | Java | 2,752 | 57.33 |
| CLARIN (OLAC-DMCI) | 79 | XSLT | 404 | 5.11 |
| CLARIN (OLAC-DMCI) | 79 | XSLT (Compact Syntax) | 255 | 3.22 |
| TBX | 48 | LIXR | 197 | 4.10 |
| CLARIN (OLAC-DMCI) | 79 | LIXR | 176 | 2.23 |
| MetaShare | 730 | LIXR | 2,487 | 3.41 |

**Table 1.** Comparison of XML to RDF mapping implementations, by number of tags in XML schema, and non-empty lines of code (LoC)

To evaluate the effectiveness of our approach we compared directly with two other XML to RDF transformations, we had carried out in this project, and reimplemented them using the LIXR language. In table **??**, we see the effort to implement these using LIXR is approximately half of using XSLT and about ten times less than writing a converter from scratch.

In addition to the reduction in effort using this approach, we also note several other advantages of the LIXR approach, due to its declarative declaration

[8] This is a newer feature of Scala only supported since 2.10 (Jan 2013)

– We can easily switch to using a stream-based parse for XML (e.g., SAX) so we can process large files without having to use much memory
– A reverse mapping can be extracted that re-generates the XML from the outputted RDF
– We can extract the type, range and domain of RDF entities generated during this procedure. This export formed the initial version of the ontology described in this paper

### 4.2 Harmonizing MetaShare with other metadata sources [JPM]

## 5 Discussion

### 5.1 Applications of the MetaShare model (beyond LingHub) [MV]

### 5.2 5.2 Challenges and future outlooks [PC]

## 6 Conclusion

## References

1. Piperidis, S.: The META-SHARE language resources sharing infrastructure: Principles, challenges, solutions. In: Proceedings of the Eighth Conference on International Language Resources and Evaluation. pp. 36–42 (2012)