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 Bielefeld, 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] ILSP/Athena R.C., Athens, Greece,
penny@ilsp.athena-innovation.gr
[4] University Pompeu Fabra, Barcelona, Spain,
marta.villegas@upf.edu

Abstract. **Keywords:** keywords

1 Introduction

[7]

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]

When mapping an XML scheme to RDF there are naturally differences that must be accounted for, which generic mapping methodologies cannot accommodate without tending to a high degree of verbosity. The basic mapping strategy is ... [MV].

Beyond this, we made the following extensions to our mapping strategy:

- We shorten some names such as ConformanceToBestStandardsAndPractices
  JPM: Perhaps we introduce sameAs links to handle this

- Developed novel classes based on existing values, e.g., Corpus ≡ ∃resourceType.corpus
- Removing unnecessary properties such as `versionInfo`.
- Generalized elements such as `userNature`, `notAvailableThroughMetashare`
- Grouping similar elements under novel superclasses, e.g., `DiscourseAnnotation`, `genre`
- Extending existing classes with new values and including new properties (see section 3.5

3.4 Interface with DCAT and other vocabularies [JPM]

3.5 Licensing module [VRD, PL]

Skeleton

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

We discuss on the needs that motivated the evolution from the previous model. We describe (if not done before) also the precedure and methodology.

Short introduction on the ODRL vocabulary.

We describe the most important changes that we have introduced.

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 [6] 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 [6], 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 Mapping META-SHARE to RDF [JPM]

When translating XML documents into RDF, one of the most common approaches is based on Extensible Stylesheet Language Transformations (XSLT) [10, 8, 2], which has been extended by some authors into a significant framework [5]. However, XSLT has a number of disadvantages for this task:

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

[6] http://linghub.org

– 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[7][8]

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 [4] 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 [9]. A simple example of a LIXR mapping is given below:

```
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[9]. 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`.

[7] Compact XML: https://pythonhosted.org/compactxml/
[8] Jade: http://jade-lang.com/
[9] This is a newer feature of Scala only supported since 2.10 (Jan 2013)

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-trivial 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 particular these were the TBX model [1] as well as the OLCA-DMCI profile of the CLARIN metadata [10]. In table 1, 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

- 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 other resources with META-SHARE [JPM]

LingHub brings resources from a wide-range of sources and while we can use standards such as DCAT and Dublin Core to guarantee a common representation of the basic Metadata of a resource, there does not a standard for the represenation of metadata specific to linguistics. For that reason, we use the META-SHARE model as a standard for other resources to use in LingHub. In particular, we will focus on the application of META-SHARE as a model to harmonize CLARIN data. The CLARIN repository describes its resources using a small common set of metadata and a larger description defined by the Component Metadata Infrastructure [3, CMDI]. These metadata schemes are extremely diverse as shown in table 2.

[10] http://catalog.clarin.eu/ds/ComponentRegistry/rest/registry/profiles/clarin.eu:cr1:p_1288172614026/xsd

Component Root Tag	Institutes	Frequency
Song	1 (MI)	155,403
Session	1 (MPI)	128,673
OLAC-DcmiTerms	39	95,370
mods	1 (Utrecht)	64,632
DcmiTerms	2 (BeG,HI)	46,160
SongScan	1 (MI)	28,448
media-session-profile	1 (Munich)	22,405
SourceScan	1 (MI)	21,256
Source	1 (MI)	16,519
teiHeader	2 (BBAW, Copenhagen)	15,998

Table 2. The top 10 most frequent component types in CLARIN and the institutes that use them. Abbreviations: MI=Meertens Institute (KNAW), MPI=Max Planck Insitute (Nijmegen), BeG=Netherlands Institute for Sound and Vision, HI=Huygens Institute (KNAW), BBAW=Berlin-Brandenburg Academy of Sciences

5 Discussion

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

5.2 Challenges and future outlooks [PC]

6 Conclusion

References

1. Systems to manage terminology, knowledge and content – TermBase eXchange (TBX). Tech. Rep. 30042, ISO (2008)
2. Borin, L., Dannells, D., Forsberg, M., McCrae, J.P.: Representing Swedish lexical resources in RDF with lemon. In: Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference (2014)
3. Broeder, D., Windhouwer, M., Van Uytvanck, D., Goosen, T., Trippel, T.: CMDI: a component metadata infrastructure. In: Describing LRs with metadata: towards flexibility and interoperability in the documentation of LR workshop programme. p. 1 (2012)
4. Fowler, M., Parsons, R.: Domain-specific languages. Addison-Wesley Professional (2010)
5. Lange, C.: Krextor – an extensible XML → RDF extraction framework. Scripting and Development for the Semantic Web (SFSW) (449), 38 (2009)
6. Maali, F., Erickson, J., Archer, P.: Data catalog vocabulary (DCAT). W3C recommendation, The World Wide Web Consortium (2014)
7. 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)
8. Van Deursen, D., Poppe, C., Martens, G., Mannens, E., Walle, R.: XML to RDF conversion: a generic approach. In: Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS'08. International Conference on. pp. 138–144. IEEE (2008)

9. Wampler, D., Payne, A.: Programming Scala, chap. 11. O'Reilly (2008)
10. Wüstner, E., Hotzel, T., Buxmann, P.: Converting business documents: A clarification of problems and solutions using XML/XSLT. In: Advanced Issues of E-Commerce and Web-Based Information Systems, International Workshop on. pp. 61–61. IEEE Computer Society (2002)